

博士論文

身体図式の逐次学習機能を有する
知能ロボットシステムの研究

令和3年12月3日提出
指導教員 稲葉 雅幸 教授

東京大学大学院 情報理工学系研究科
知能機械情報学専攻
48-197506 河原塚 健人

目 次

第1章 序論	13
1.1 本研究の背景	15
1.2 本研究の目的	16
1.3 本論文の構成	17
第2章 身体図式の逐次学習機能を有する智能ロボットシステム	21
2.1 智能ロボットシステム	24
2.1.1 開発する智能ロボットシステムの要件	24
2.1.2 智能ロボットシステムの先行研究と本研究の位置づけ	24
2.2 身体図式の逐次学習機能	28
2.2.1 身体図式とは	28
2.2.2 開発する身体図式の要件	29
2.2.3 身体図式とその他モデルの比較	30
2.2.4 身体図式学習に関する先行研究	30
2.3 柔軟性と冗長性を有するロボットの智能システム	36
2.3.1 ロボットにおける柔軟性と冗長性	36
2.3.2 柔軟性と冗長性に基づくロボットの分類	38
2.3.3 柔軟性と冗長性を有するロボットの智能システム設計	42
2.3.4 身体図式学習	44
2.4 身体図式の逐次学習機能に基づく環境適応能力の向上	45
2.4.1 身体-道具-動作環境の相関複雑性の分類と攻略	45
2.4.2 身体-道具-動作環境の時間的変容の分類と攻略	46
2.5 筋骨格構造・身体図式学習に関する基礎知識	48
2.5.1 筋骨格ヒューマノイドの基本構造と特徴	48
2.5.2 筋骨格構造の基本的な制御と状態推定	54
2.5.3 身体図式学習に用いるニューラルネットワークの基礎	60
2.6 本章のまとめ	66
第3章 身体図式の逐次学習システム	69
3.1 身体図式の逐次学習システム - 一般化多感覚相関モデルの学習と利用	72
3.1.1 ネットワーク構成	72

3.1.2	ネットワークの基礎的利用法	76
3.1.3	データ収集	77
3.1.4	ネットワーク学習	78
3.1.5	ネットワーク更新	79
3.1.6	順伝播と誤差逆伝播の繰り返しによる最適化計算	79
3.1.7	状態推定	80
3.1.8	制御	81
3.1.9	シミュレーション	82
3.1.10	異常検知	82
3.1.11	一般化多感覚相関モデルの特徴	83
3.1.12	プログラム構成	84
3.2	身体図式ネットワークの入出力自動決定	85
3.2.1	概要	85
3.2.2	身体図式ネットワークの入出力自動決定	86
3.3	本章のまとめ	90
第4章	身体図式学習に向けたハードウェア構成	93
4.1	軸駆動型ロボット PR2 のハードウェア構成	96
4.2	台車型ロボット Fetch のハードウェア構成	97
4.3	低剛性樹脂製ロボット KXR のハードウェア構成	98
4.4	柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成要件	99
4.4.1	学習制御プラットフォームの構成要件	101
4.4.2	開発する構成要素	101
4.5	柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成要素	102
4.5.1	筋モジュール	102
4.5.2	関節モジュール	104
4.5.3	筋経由点ユニット	105
4.5.4	非線形弾性ユニット	106
4.5.5	冗長センサを備えた柔軟ハンド	111
4.5.6	可動眼球ユニット	112

4.5.7	全周触覚を備えた足	112
4.5.8	人体模倣型両耳ユニット	113
4.5.9	臀部接触センサ	114
4.6	柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成	116
4.6.1	MusashiLarm	116
4.6.2	Musashi	119
4.6.3	MusashiOLegs	121
4.6.4	TWIMP	122
4.6.5	Musashi-W	130
4.6.6	Kengoro	131
4.7	筋破断を補償する冗長性最大化に基づく筋配置最適化	134
4.7.1	概要と先行研究	134
4.7.2	筋破断を補償する冗長性最大化に基づく筋配置最適化手法	135
4.7.3	実験	139
4.7.4	議論	144
4.8	本章のまとめ	146
第5章	身体図式学習を支える反射制御	147
5.1	反射制御の要件	150
5.1.1	柔軟性と冗長性の欠点補完	150
5.1.2	人間の反射制御とロボットへの適用	151
5.1.3	反射制御システムの概要	153
5.2	筋温度制御	155
5.2.1	概要と先行研究	155
5.2.2	筋温度制御システム	156
5.2.3	実験	162
5.3	筋弛緩制御	171
5.3.1	概要と先行研究	171
5.3.2	筋弛緩制御システム	172
5.3.3	実験	175

5.4	速度最大化反射戦略	182
5.4.1	概要と先行研究	182
5.4.2	冗長腱駆動構造の問題点	182
5.4.3	速度最大化反射システム	183
5.4.4	実験	186
5.4.5	議論	190
5.5	拮抗筋抑制制御	194
5.5.1	概要と先行研究	194
5.5.2	拮抗筋抑制制御システム	194
5.5.3	拮抗筋抑制制御の特徴	196
5.5.4	実験	198
5.6	伸長反射制御	207
5.6.1	概要と先行研究	207
5.6.2	伸長反射システム	207
5.6.3	実験	210
5.6.4	議論	215
5.7	本章のまとめ	218
第 6 章	身体図式学習の基礎実験	221
6.1	身体図式ネットワークの入出力自動決定実験	224
6.1.1	静的身体図式の入出力自動決定実験	224
6.1.2	動的身体図式の入出力自動決定実験	226
6.2	静的道具操作学習 - 身体-道具間関係変化への適応	231
6.2.1	概要と先行研究	231
6.2.2	逐次的な把持状態変化を考慮した適応的道具先端操作学習	233
6.2.3	実験	236
6.2.4	議論	242
6.3	静的筋骨格身体動作学習 - 関節角度-筋張力-筋長間関係への適用	247
6.3.1	概要と先行研究	247
6.3.2	Musculoskeletal AutoEncoder	248

6.3.3	実験	254
6.3.4	Musculoskeletal AutoEncoder の特殊系に関する考察	259
6.4	動的模倣動作学習 - 動作スタイルを考慮可能な模倣学習	271
6.4.1	概要と先行研究	271
6.4.2	動作スタイルに追加制約を考慮可能な模倣学習	273
6.4.3	実験	276
6.4.4	議論	283
6.5	動的筋骨格ハンド学習制御 - 関節角度-筋張力-筋長-接触力関係への適用	287
6.5.1	概要と先行研究	287
6.5.2	筋骨格ハンド	288
6.5.3	Hand Dynamics Network	289
6.5.4	実験	294
6.6	本章のまとめ	304
第7章	身体図式学習の応用実験	307
7.1	身体図式ネットワーク入出力の減少 - 筋破断を考慮した適応的身体図式学習	310
7.1.1	概要と先行研究	310
7.1.2	筋破断を考慮した適応的身体図式学習	313
7.1.3	実験	322
7.1.4	議論	332
7.2	身体図式ネットワーク入出力の増加 - 筋追加を考慮した適応的身体図式学習	335
7.2.1	概要と先行研究	335
7.2.2	筋追加を考慮したハードウェア設計	337
7.2.3	筋追加を考慮した適応的身体図式学習	338
7.2.4	実験	340
7.2.5	議論	344
7.3	確率を含む静的身体図式学習 - 危険回避学習	347
7.3.1	概要と先行研究	347
7.3.2	危険回避のオンライン学習	348
7.3.3	実験	352

7.3.4	議論	354
7.4	平均分散表現を含む動的体図式学習 - 環境適応型台車制御学習	358
7.4.1	概要と先行研究	358
7.4.2	SPNPB による分散最小化を含む環境適応型制御	359
7.4.3	実験	362
7.4.4	議論	369
7.5	体図式の自動分割 - 筋骨格ヒューマノイドにおける筋分割	370
7.5.1	概要と先行研究	370
7.5.2	機能的接続と空間的接続を利用したセンサ・アクチュエータの自動分割	371
7.5.3	筋骨格ヒューマノイドにおける筋分割	376
7.5.4	実験	377
7.5.5	議論	382
7.6	動的体図式の展開による高速制御計算 - ペダル操作学習	387
7.6.1	概要と先行研究	387
7.6.2	筋骨格ヒューマノイドにおける自動運転	388
7.6.3	タスク特化型自己身体制御獲得	390
7.6.4	実験	393
7.6.5	議論	397
7.7	動的体図式の展開による高速制御計算 - 把持安定化学習	399
7.7.1	概要と先行研究	399
7.7.2	把持安定化制御	401
7.7.3	実験	406
7.7.4	議論	408
7.8	明示的パラメータを用いた動的体図式学習 - 筋温度制御	413
7.9	本章のまとめ	415
第 8 章	体図式の逐次学習機能を有する知能ロボットシステムによる環境適応能力の向上	419
8.1	体図式の逐次学習機能を有する知能ロボットシステムによる環境適応能力の向上	422
8.1.1	体図式の逐次学習機能を有する知能ロボットシステム	422
8.1.2	体図式の逐次学習機能による相関複雑性の攻略	425

8.1.3	身体図式の逐次学習機能による時間的変容の攻略	426
8.2	静的身体図式を用いた動作修正手法	429
8.2.1	概要と先行研究	429
8.2.2	筋長補償制御による動作修正	429
8.2.3	実験	433
8.2.4	議論	439
8.3	筋骨格ヒューマノイドによる自動運転実験	441
8.3.1	概要と先行研究	441
8.3.2	筋骨格ハードウェアと自動運転	442
8.3.3	筋骨格ソフトウェアと自動運転	443
8.3.4	実験	448
8.4	低剛性樹脂製ヒューマノイドによる道具操作実験	452
8.4.1	概要と先行研究	452
8.4.2	低剛性樹脂製ヒューマノイドによる全身道具操作	453
8.4.3	実験	456
8.5	筋骨格ヒューマノイドによる柔軟布操作実験	464
8.5.1	概要と先行研究	464
8.5.2	可変剛性と素材変化を考慮した動的柔軟布操作	465
8.5.3	実験	470
8.5.4	議論	478
8.6	全身筋骨格ヒューマノイドのバランス制御実験	483
8.6.1	概要と先行研究	483
8.6.2	動的身体図式学習を用いた筋骨格ヒューマノイドのバランス制御	484
8.6.3	実験	487
8.6.4	実機実験	491
8.6.5	議論	493
8.7	本章のまとめ	496
第9章	結論	499
9.1	本研究の総括	501

9.2 結論	504
9.3 今後の展望	504
謝辞	507
発表文献	511
参考文献	527

第1章

序論

1.1 本研究の背景

これまで、産業用ロボットアームから始まり、軸数や軸配置を変えながら様々なロボットが開発されてきた。その発展とともにロボットの活躍の場も変化し、産業用から生活環境 [1]、介護現場 [2]、災害現場 [3] 等の様々な場所への応用が期待されている。2015 年には、DARPA Robotics Challenge (DRC) [4] が開催された。一台のロボットで移動、ドア開け、バルブ回し、不整地走行、階段昇降等のタスクをこなすことが求められた。その結果として、何も示し合わせることなく、全てのチームが 2 本の手と 2 本の足を持つヒューマノイドロボットを採用した [5, 6, 7, 8]。これは、人間と同様の環境中を移動・認識・操作していくためには、人間と同様の身体構造が必要であることを間接的に示唆した。一方、ロボットはその認識誤差や操作誤差、認識の難しい常に変化し得る環境に上手く対応できず、ほとんどのチームが動作を完遂することなく幕を閉じた。これらの失敗から、様々な方向の研究が進められている。無論、より正確な認識や行動計画、力制御等が多く開発されている一方、人間や生物のような、ハードウェアとして柔軟で冗長な身体に着目した研究も盛んに行われている。この一つの潮流として、ソフトロボティクス [9, 10] がある。ここでは、軸駆動型だけでなく、腱駆動型・空気圧型・誘電エラストマーアクチュエータ型等の様々なロボットが開発され、徐々に成果が実を結びつつある。このように、現在のロボットの形態は一つではなく、様々なアクチュエーション、センシングが含まれ、これらにより、様々な道具や環境に対応できるようなロボットが望まれている。

一方で人間の知能について考えてみる。人間はロボットのように予めモデリングされているわけではなく、自律的に経験から自身の身体モデルを構築していく。これにより、身体・道具・動作環境の間の相関複雑性やそれらの時間的変容を捉え、適応することで生きている。本論文ではこれを環境適応能力(ここで言う環境とは身体・道具・動作環境の全てを含む)と呼んでいる。しかし、これまでのロボットはこの環境適応能力に乏しいため、柔軟身体や柔軟物体、環境や道具の変化等を扱うことは難しい。

環境適応能力を有する知能ロボットシステムに足りない点は何であろうか。本研究では、これを主に (1) モデル化困難性の攻略, (2) 逐次的モデル変化の攻略, (3) 一般化であると考えている。(1) はソフトロボットを始めとした柔軟な身体においてよく問題視され、これを解決する手法が盛んに開発されている。一方、モデル化困難性はソフトロボットに限った話ではない。通常の産業用軸駆動型ロボットでも、ぴったり 1 m 手を動かせと言われて動かせるわけではなく、どんなに小さくても必ずその動作に誤差、つまりモデル化のできていない要素がある。ましてや、環境に接触して動作する場合や道具を持って行動する場合は、その環境や道具の特性を完璧にモデル化するのは不可能なため、大きく誤差

が出やすい。また、身体の様々な感覚間の冗長な関係性をモデル化することも同様に難しい。ある方向に手を動かした時にその手は目にどう映るか、どのような音が鳴るのか、手を動かした方向に環境があったときどのように接触力が測定されるのか、どのような反力を受けるのか、これらの関係性を実世界かつ身体が柔軟であっても推論できる機能は現在のどのロボットにも備わっていない。(2)は、(1)を攻略しモデル化を正確に行っても、その対象とするロボットの身体や道具、環境は逐次的に変化し、構築したモデルも変化していくという点である。例えばソフトロボットのような、ゴムやバネで構成された柔軟身体は経年劣化を起こしやすい。また、初期化ごとに微妙に身体のセンサ値遷移が異なることも多い。把持して用いる道具は毎回異なるであろうし、ロボットが動作する環境も常に一定ではない。つまり、知能ロボットにおいてモデルは逐次的に更新されるべきものである。最後に(3)は、現在のように様々なアクチュエーション・センシングが含まれるロボット全てに適用可能な一般化が必要であるという点である。知能ロボットシステムは、アクチュエータ・センサの、ある種類の組み合わせに対してしか適用できない手法であってはならない。ロボットごとに異なるセンサ・アクチュエータの関係性を人間のように自律的に経験から学習・理解する必要がある。

これら(1)–(3)を解決することで、様々なロボットの環境適応能力を格段に向上させることができる。これまで、モデル化困難ゆえに考えることの無かった身体-道具-動作環境の様々な要素をモデル化することで、柔軟な身体や道具、摩擦の異なる床等を扱うことができるようになる。また、そのモデルの逐次的な変化を捉えることができれば、様々な身体状態や道具、環境に適応した環境適応行動が可能となる。最後に、センサ・アクチュエータの相関関係の学習を自律的に行うことで、様々なロボットに適用可能な一般化された知能ロボットシステムの構築が可能となる。

1.2 本研究の目的

本研究では、柔軟性と冗長性から生まれる身体-道具-動作環境の相関複雑性と時間的変容に対応可能な、人間のような環境適応能力の高いシステムを、ロボットの知能として実現する方法を明らかにする。そのために、人間の身体・反射・学習能力に着想を得た、柔軟性と冗長性の利点を多く持つロボットハードウェアの構成法、柔軟性と冗長性による欠点を補うような低レイヤの反射制御、長い周期で柔軟性と冗長性の利点を増強し欠点を補う身体図式学習を開発する。これにより、様々なロボットのタスクにおける環境適応能力が向上することを実験的に評価する。

本研究のコンセプトをよりわかりやすく図に示したものがFig. 1.1である。本研究は人間の知能の模倣、その中でも高度な自律性に着目する。人間はこの自律性により、身体-道具-動作環境間の相関複雑性と時間的変容に適応可能であり、環境適応能力が高い。一方これまでのロボットはこの環境適応

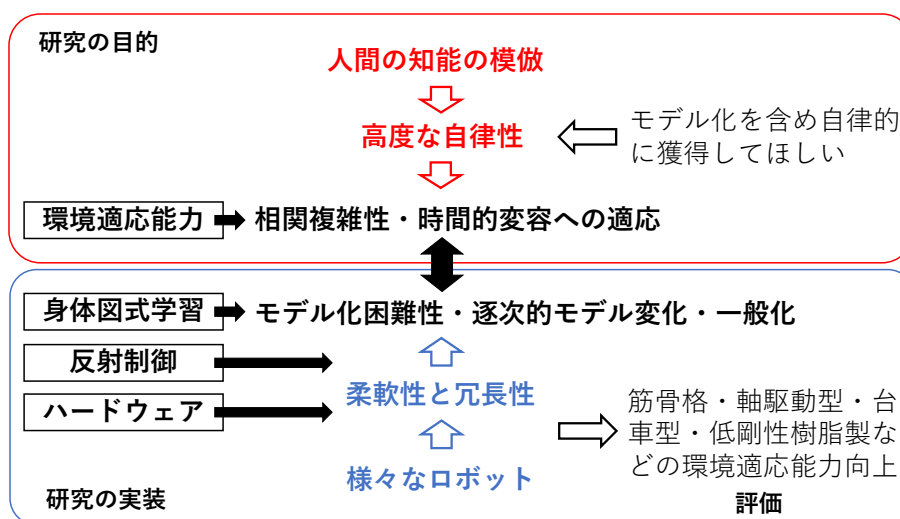


Fig. 1.1: The concept of this study.

能力に乏しく、これを解決する知能ロボットシステムを開発する。これまで様々な形態のロボットが開発されてきたが、それらは全て柔軟性と冗長性を持ち、これはモデル化困難性と逐次的モデル変化を生む。また、様々なロボットに適用可能なシステムとして、一般性を持つ必要がある。ゆえに、これらに対応可能な身体図式の逐次学習機能を開発し、これに基づく知能ロボットシステムにより、筋骨格型・軸駆動型・台車型・低剛性樹脂製等の様々なロボットの環境適応能力向上を実現する。一方で、身体図式学習システムだけでは、ロボット身体の環境適応能力を最大限引き出すことはできず、また、柔軟性と冗長性による制御的な問題点を解決することができない。柔軟性と冗長性を最大化するロボットの身体構成法、また、短い周期で柔軟性と冗長性の制御の問題点を解決する低レイヤの反射制御が必要である。このハードウェア構成・反射制御・身体図式学習機能を合わせた知能ロボットシステムにより、様々なロボットの環境適応能力を向上させることを本研究の目的とする。

1.3 本論文の構成

本論文は全9章から構成される。以下に各章の概要を述べる。また、それらの関係性について Fig. 1.2 に図示する。

第1章「序論」では、本研究の背景と目的、及び本論文の構成について述べた。ロボットの身体におけるモデル化困難性と逐次的モデル変化を攻略する知能ロボットシステムの開発を目指し、柔軟性と冗長性を有する筋骨格ヒューマノイドを含む様々なロボット・部位・タスクへの本手法の適用、ロボットの環境適応能力向上を目指す。

第2章「身体図式の逐次学習機能を有する知能ロボットシステム」では、本研究で提案する身体図式の要件とその他身体モデルとの比較、また柔軟性と冗長性に基づくロボットの分類と本手法の適用について議論する。また、身体設計・反射制御・身体図式学習について概要と先行研究を述べ、本学習機能を有する知能ロボットシステムによる環境適応能力向上について議論する。

第3章「身体図式の逐次学習システム」では、本研究の中核である身体図式の逐次学習機能の定式化と基礎システム、本ネットワークの入出力の自動決定による自律的モデル化について詳細を述べる。

第4章「身体図式学習に向けたハードウェア構成」では、身体図式学習に用いる軸駆動型・台車型・低剛性樹脂製軸駆動型・筋骨格型ロボットの詳細について述べる。特に、柔軟性と冗長性の利点を有する筋骨格ヒューマノイドの学習制御プラットフォーム設計について、構成・再構成の容易化戦略、冗長なセンサ・アクチュエータ配置のためのモジュール化戦略、冗長性最大化のための設計最適化等について述べる。

第5章「身体図式学習を支える反射制御」では、身体図式学習を補助する反射制御について述べる。長期的学習を可能にするモータコア温度制御から、筋骨格ヒューマノイドの柔軟性と冗長性の欠点を補う拮抗筋抑制制御・筋弛緩制御・関節速度最大化反射戦略・伸長反射制御についても詳細に述べる。

第6章「身体図式学習の基礎実験」では、身体図式学習を適用した基礎実験について述べる。具体的には、筋骨格ヒューマノイドの静的身体制御、筋骨格ハンドの動的制御、軸駆動型・筋骨格型における静的道具操作と動的模倣学習に適用することでその有効性を示す。

第7章「身体図式学習の応用実験」では、身体図式学習の応用実験について述べる。主に、身体図式におけるネットワーク入出力の減少と増加を伴う身体変化、身体図式の部位ごとにおける自動分割について述べる。また、これを応用した動作実現として、身体間干渉による危険回避学習、高速ペダル操作学習、分散最小化台車制御等についても述べる。

第8章「身体図式の逐次学習機能を有する知能ロボットシステムによる環境適応能力の向上」では、本研究で開発した手法を統合した、筋骨格ヒューマノイドによる運転操作実現と柔軟布操作実現、低剛性樹脂製ヒューマノイドによる道具操作実現等を行い、多様なロボットにおける身体図式学習による環境適応能力の向上についてまとめる。

第9章「結論」では、本研究の結論をまとめ、今後の課題と展望について述べる。

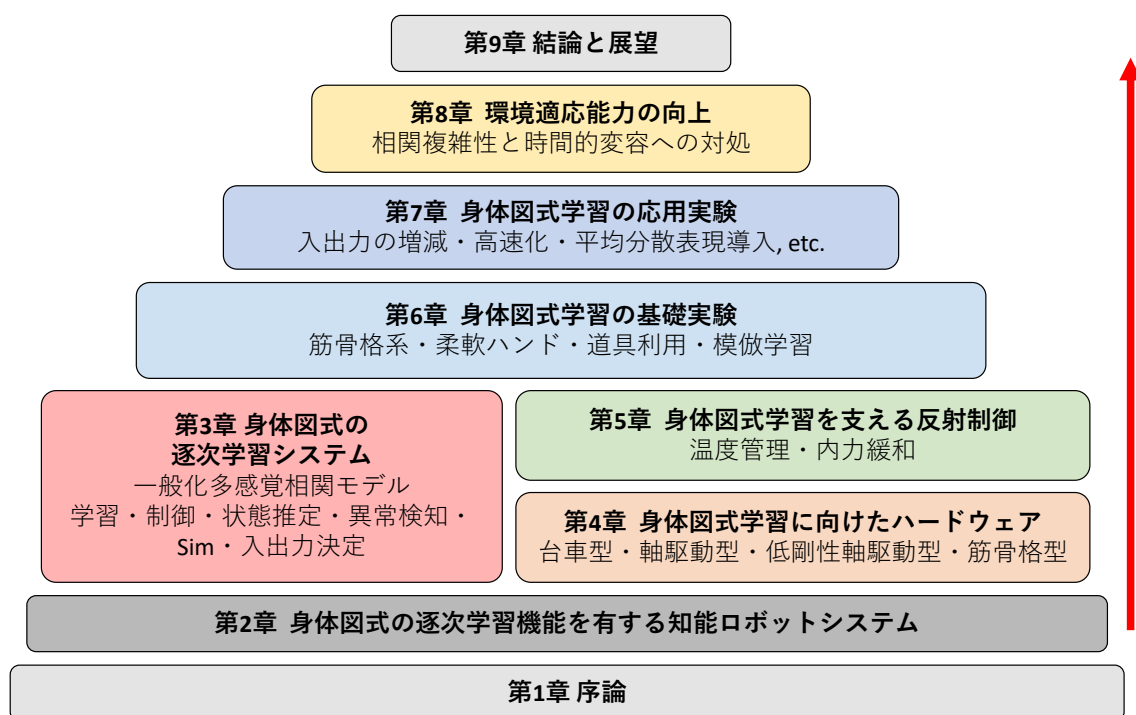


Fig. 1.2: Chapter structure of this study.

第2章

身体図式の逐次学習機能を有する 知能ロボットシステム

本章の概要を Fig. 2.1 に示す。本章ではまず、2.1 節において、これまでの知能ロボットシステムと其中における本研究の位置づけについて述べ、本研究は古典的知能システムや反射型知能システム、学習型知能ロボットシステムのハイブリッド型であり、このうち実機学習が現実的なロボット身体に近い部分を身体図式として学習し、制御や認識、状態推定に統一的に利用することで、身体-道具-動作環境の相関複雑性と時間的変容を攻略するものであることを述べる。次に、2.2 節において、この身体図式は感覚と運動の相関関係を身体構造をもって表現するものであり、心理学や脳科学における身体図式をロボットに転移した、多感覚性、汎用性、自律獲得性、変化適応性において優れる概念であることを述べる。次に、2.3 節において、全てのロボットは時間的・空間的柔軟性と制御的・感覚的冗長性を持ち、モデル化困難性と逐次的モデル変化という問題を持つこと、これらの問題を解決する身体図式学習だけではなく、柔軟性と冗長性の利点を活かすハードウェアと、その欠点を速い周期で補う反射制御が必要であることを述べる。次に、2.4 節において、本研究の知能ロボットシステムによる身体-道具-動作環境間関係の相関複雑性と時間的変容の攻略、これによる環境適応能力の向上について述べる。最後に、2.5 節では、本研究で扱う筋骨格ヒューマノイドやニューラルネットワークに関する基礎

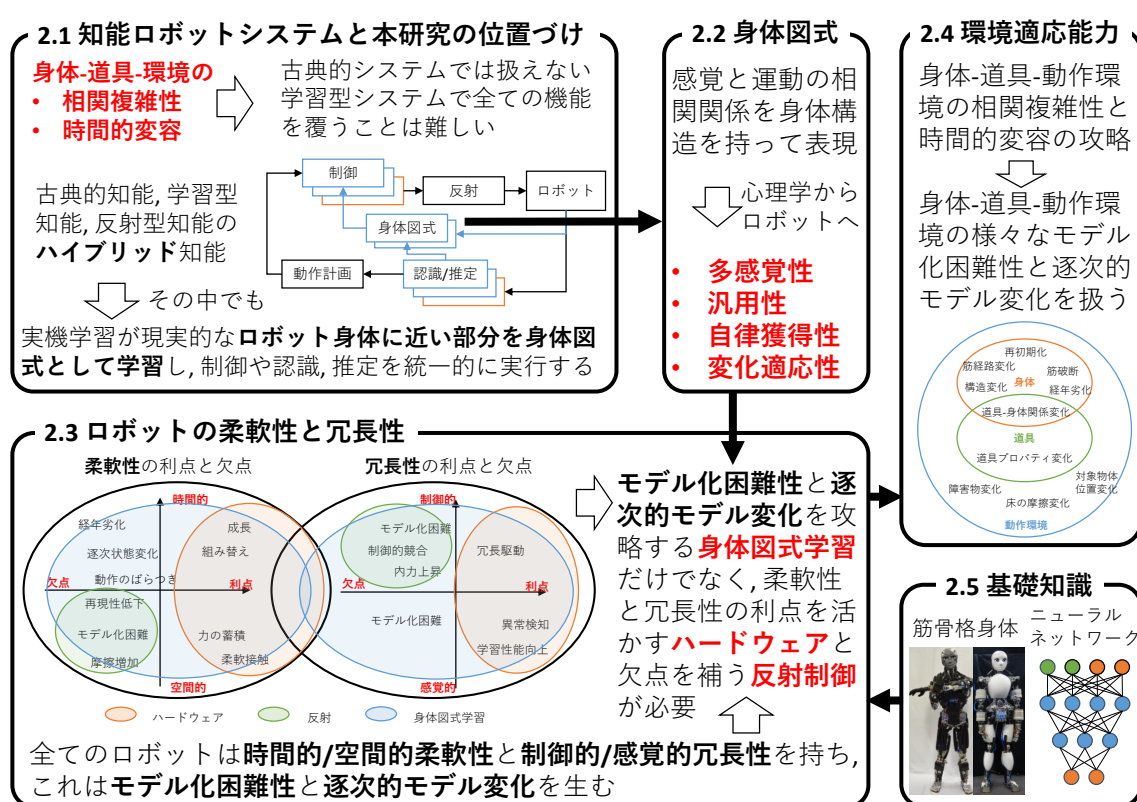


Fig. 2.1: The overview of this chapter.

知識についてまとめる。

2.1 知能ロボットシステム

2.1.1 開発する知能ロボットシステムの要件

まず、本研究における知能ロボットシステムを、“人間のように自律的に経験から学習を行うシステム”と定義する。人間は自律的な学習システムにより、その複雑な身体を自由自在に動かし、怪我や成長、未知の道具や環境に適応しながら生きている。これは言い換えると、**身体-道具-動作環境**の間の**相関複雑性**と**時間的変容**に対処していると言える。ここでいう**身体-道具-動作環境**間の相関とは、体をどう動かすと、どう視界や手の位置、物体の位置等が変化し、接触力や筋張力、筋温度等が変化していくのかという相関関係を意味する。この相関関係は一般的に非常に複雑であり、人間はこれを自律的な学習により徐々に獲得することができる。つまり、**相関複雑性**に対処することが可能である。また、ここでいう**身体-道具-動作環境**間の**時間的変容**とは、成長や怪我等による身体の変化、操作する道具や動作する環境の変化、身体と道具の関係性や身体と環境の関係性の変化を表す。人間はこの変化に対して逐次的な学習により適応し続けることが可能である。つまり、**時間的変容**に対処することが可能である。

本研究ではこの**身体-道具-動作環境**の相関複雑性と時間的変容への対応能力を、**環境適応能力**と呼ぶ。ここでいう環境とは、**身体-道具-動作環境**を表している。本研究では、人間のような自律的な学習により、**身体-道具-動作環境**の相関複雑性と時間的変容に対応する環境適応能力を向上可能な知能ロボットシステムの開発を行う。

2.1.2 知能ロボットシステムの先行研究と本研究の位置づけ

ここでは、より広い意味での知能ロボットシステムについて、その先行研究を古典的知能ロボットシステム、反射型知能ロボットシステム、学習型知能ロボットシステムに分けて紹介する。それらの簡単なアーキテクチャを図で表現したものを Fig. 2.2 に示す。

古典的知能ロボットシステム

[11] では、ロボットにおける言語の階層性について、ロボットアームを用いた作業を例に、コマンドレベル・原始的動作レベル・構造的動作レベル・対象物状態レベル・作業目標レベルの5つの階層構造に分けて整理している。対象物状態レベルで動作を記述する古典的知能ロボットシステムとして、

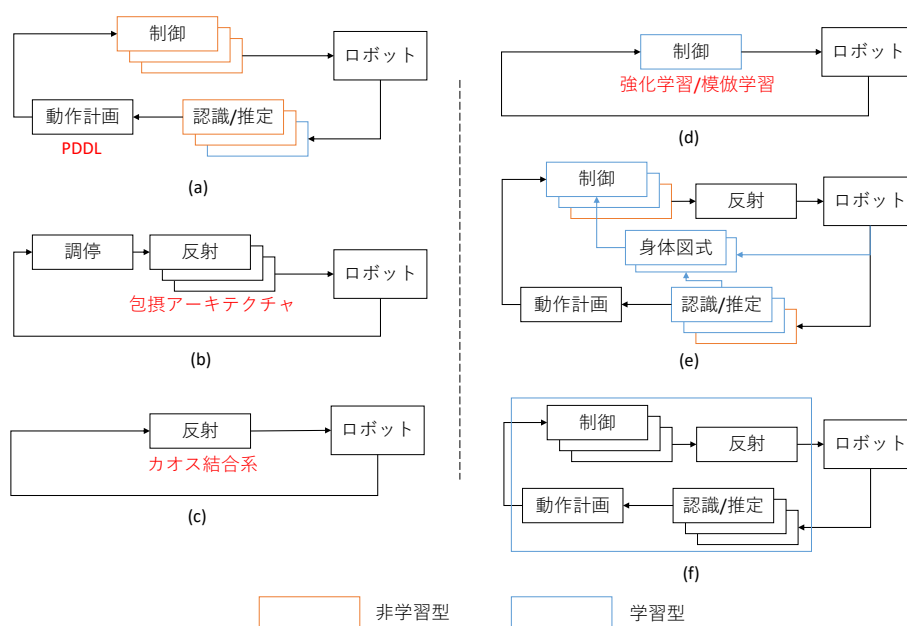


Fig. 2.2: The comparison of architectures of the intelligent robot systems.

シンボリックな状態記述言語を用いた行動計画法が挙げられる (Fig. 2.2 の (a)). この代表的な例が, STRIPS 型行動計画器 [12] である. これは, 初期状態 (Init), 目標状態 (Goal), 行動 (Actions) を定義したインスタンスを用意し, これを用いて初期状態を目標状態へと導く動作を計画する. 行動に含まれる各オペレータには事前条件や効果, 事後条件等が記述される. この STRIPS 型行動計画器を実現する言語の一つが PDDL [13] である. この PDDL には様々なバージョンがあり, STRIPS 型行動計画器の時点では対応していなかった様々な概念が取り込まれている.

コマンドレベル・原始的動作レベル・構造的動作レベルで持って Action が人間により記述され, 対象物状態レベルでの動作記述が, それらの Action の遷移を決定し呼び出す形と言える. 一方で, 人間が記述する部分が多く, 記述の難しい柔軟なロボットや道具, 環境変化適応等は扱わない場合がほとんどである.

反射型智能ロボットシステム

反射とは, 特定の感覚刺激に対する無意識の反応を示す. 反射型智能ロボットシステムとして最もよく知られるものに包摂アーキテクチャ [14] がある (Fig. 2.2 の (b)). これは, 複雑で知的な振る舞いをする行動を単純な振る舞いのモジュールに分け, 階層構造を構築する. それぞれのモジュールは全てのセンサ値を参照でき, アクチュエータに指令を送ることができる. 上位の階層ほど抽象的な目的

を持ち、それぞれの層は下位の層の目的を包摂する。上位のモジュールが下位のモジュールを上書き・抑制することで行動の調停が行われる。このような感覚と運動を結ぶモジュール群が並列に実行される仕組みは、並列行動アーキテクチャと呼ばれる。これを拡張した手法として、視覚を導入しデータの流れを制御することでより高い目的志向性を持った sonja システム [15] も開発されている。

この他にも、カオス結合系 [16] を使った知能ロボットシステムが例として挙げられる (Fig. 2.2 の (c))。カオス結合系とは、複数のカオス写像をある結合場によって結合すると、初期値鋭敏性により各要素がばらけようとする働きと、結合場によって収束しようとする働きの間に競合が起き、多様な部分的協調パターンが発現する系である。この結合場を身体ダイナミクスとすることで、身体の多自由度における様々な協調パターンが創発される。初めはランダムな動きであるが、数秒すると秩序的な動作が創発し、障害物等の状況変化が生じると動きが一旦乱れ、また新たな秩序を形成する。また、低レイヤにおける自律システムとして、Central Pattern Generator (CPG) [17] が知られている。これは、除脳ネコに代表されるような身体の神経振動子と呼ばれるリズム発生機構による周期的な運動パターンの生成系である。なお、外部からの入力なしに動作する自律系であるため、厳密には反射とは異なる。神経系振動子への入力信号を調整することで、その分岐現象により様々な歩容が自律的に生成されることが示されており、環境変化に適応的な知能ロボットシステムが開発されている。この CPG の仕組みと Hebb 学習の仕組み [18] を組み込んだ胎児からの発達シミュレーションも研究されている [19]。

これらの反射型システムは、実世界で即応的に動作する適応性のあるシステムが構築できる。また、主に行動創発の文脈で取り上げられることが多い。一方で、カオス結合系や CPG は目的志向性の高いシステムを実現することが難しく、包摂アーキテクチャは階層が増えると各層のモジュールやその調停の設計が困難であるという問題がある。

学習型知能ロボットシステム

学習型の知能というと、現在最も盛んに研究されているのは、強化学習 [20, 21] であろう (Fig. 2.2 の (d))。これは、報酬として現在状態に対して何らかの評価値を与え、これを予測できるようにベルマン方程式からネットワークを更新していく。また、模倣学習 [22, 23] は人間のデモンストレーションを模倣するように学習することで、適応的な動作が可能になる。予測モデル学習 [24, 25] は制御入力に関する感覚の遷移を表現する予測モデルを学習し、これを元にモデル予測制御を行う。これらの模倣学習や予測モデル学習は教師あり学習に分類される。[26] では、大脳皮質は教師なし学習を、大脳基底核は強化学習を、小脳は教師あり学習を行っているとして、学習アルゴリズムによる脳の機能分化について論じている。

この他にも、予測符号化 [27] や自由エネルギー原理 [28] に基づく知能システムの研究がなされている。予測符号化は予測モデル学習と大きくは変わらないが、予測誤差を最小化するように情報を再現することが脳機能の本質であるとする神経科学方面からの主張である。自由エネルギー原理はこの予測モデルを使い、感覚入力の予測にくさを最小化するように内部モデル及び行動を最適化し続けるという法則である。なお、これらを実機のロボットに適用しタスクをこなすような例はほとんどない。

強化学習や模倣学習、予測モデル学習は、学習という仕組みによって複雑な身体やモデル化の難しい問題にも柔軟に適用することが可能である。これらは、原始的動作レベル・構造的動作レベル・対象物状態レベル・作業目標レベルなど様々なレイヤに利用することができ、レイヤが上がるほどその学習は難しくなる。究極的にはこれらの学習によって作業目標レベルでの記述ができるはずであるが、現実はその簡単ではなく、実機における試行回数の限界やモデルの表現力等から非常に困難である。

本研究の位置づけ

このように、様々な知能ロボットシステムが開発されてきたが、全てが同じレイヤで議論されているわけではない。PDDL は対象物状態レベルであるし、包摂アーキテクチャはコマンドレベルや原始動作レベルでの反射制御の積み重ねによって対象物状態レベルや作業目標レベルの動作を暗黙的に実現している。学習型システムはこれらのどの階層にも適用できるが、抽象的な指令を実行するには限界がある。そして、本研究はこれらの研究と競合するものではない。身体・道具・動作環境の相関複雑性と時間的変容は古典的知能ロボットシステムでは扱うことができず、一方で学習型知能ロボットシステムとして End-to-End に全ての作業を扱うことは難しい。また、反射型知能ロボットシステムも有用な場面は多い一方で、これにより全てを置き換えることはできない。よって本知能ロボットシステムを、古典的知能ロボットシステム、反射型知能ロボットシステム、学習型知能ロボットシステムのハイブリッド知能システムと考える。その中でも、実機学習が現実的な、ロボット身体に近い部分を身体図式という形で学習する。予測モデル学習や模倣学習の入出力を一般化し、多感覚間の相関表現を取り入れ、この身体図式の制御・状態推定・異常検知・シミュレーション等への利用を統一的に表現し、相関複雑性と時間的変容に対応可能としたものである。一部では一般的なコマンドレベル・原始的動作レベル・構造的動作レベルでの動作記述を行いながら、これらで人間が直接記述することの難しい、後に説明する柔軟性と冗長性を扱う部分について、本身体図式学習を導入していく。図でまとめると、Fig. 2.2 の (e) になる。一般的な動作計画や認識と推定、制御、反射等のコンポーネントがあるが、この中で制御や認識、推定については学習型手法と古典的手法の両者を用いる。本研究は、その中でも学習

の部分について身体図式学習という形で一般化を施していると捉えることができる。

この身体図式学習は、End-to-End に動作が生成されるわけではなく、学習した身体図式の利用方法にユーザの思考を反映できる点で、先に述べた学習型知能ロボットシステムに比べ、機能を調整しやすい形となっている。主に、損失関数の形式や入力 of 更新率、バッチ数、エポック数等をユーザが自由に変更することが可能である。全てのパラメータを人間が管理する古典的知能ロボットシステムにその機能の調整能力では劣るが、これまで扱うことの難しかった身体-道具-動作環境の相関複雑性と時間的変容に対応可能な環境適応能力が得られる点、ある程度の機能調整が可能かつそれらを統一的に様々なロボットやタスクに適用できる点で、本研究はその他の知能ロボットシステムに比べ優れている。

一方で、知能ロボットシステムの究極形としては、(f) が望ましい。人間は一つの脳というネットワークシステムにおいて、認識や動作計画、制御、反射等が分化していき、全体システムを形作る。しかし、現状脳と同程度の表現力を有するネットワーク構築が難しいこと、このような機能の分化や実機における学習システムの構築が困難であることから、本研究では (e) の形を採用している。

2.2 身体図式の逐次学習機能

2.2.1 身体図式とは

人間における知能システムは、心理学や神経科学の分野では身体図式 (**Body Schema**) や身体像 (**Body Image**) という言葉で表現される [29, 30]。身体図式は、現在の姿勢状態や各身体部位の位置関係、ある動作を行うためにどのように身体を動かせば良いのか等を表す、感覚と運動の関係を身体という構造を持って表現した自己身体認知の暗黙的な枠組みである。一方、これと区別されて議論される概念に身体像 (**Body Image**) がある。これは自己の身体を対象とする認知であり、世界座標から見た意識的な自己認知と言える。身体図式が無意識的な、意識の志向対象とはならない自己認知なのに対して、身体像は意識の志向対象となる。また、身体図式は自己の身体を中心として展開され、外部の物体との相対的な位置関係を持たないのに対して、身体像は外部から自己の身体を対象化した存在であり、外部の物体と並列に置かれた存在である。身体図式は道具によって拡張され、視覚的認知可能な外部物体が自己の体性感覚に取り込まれる。そして、身体故障や成長に伴い、時間とともに変化していくものである。

[31] では身体図式について基礎的な7つの性質を提示している。それは“空間知覚性”、“モジュラ性”、“動作更新性”、“変化適応性”、“多感覚性”、“時空間一貫性”、“他者間共通性”の7つである。空間知覚性は身体図式が身体部位を空間内の体積を持った物体としてその位置や配置を表現していること、

モジュラー性は身体図式が身体部位ごとにモジュール化されていること、動作更新性は身体図式が常に体の動きに従ってそれを認識し更新されること、変化適応性は怪我や成長に伴って身体図式が変化すること、多感覚性は身体図式が多様な感覚情報を統合していること、時空間一貫性は身体図式が時間的かつ空間的に一貫した構造を提供すること、他者間共通性は共通の身体図式がそれぞれの人間に共有されていることを表す。

2.2.2 開発する身体図式の要件

本研究では、感覚入力や制御入力間の関係性を表現し、体をどう動かすと視界や接触、音や温度がどう変化するという相関を記述するモデルを身体図式と呼び、これを逐次的に学習することでロボットの環境適応能力を向上させることを考える。この身体図式を核として、ロボットの制御や状態推定、異常検知やシミュレーションを行う。本研究の身体図式は心理学や神経科学における身体図式の一部の特徴を知能ロボットシステムの観点から抽出したものである。特に、空間知覚性、他者間共通性は取り扱わない。空間知覚性については有用な場面があるものの、本研究の感覚入力や制御入力の相関表現には必要ないため扱っていない。動作更新性、時空間一貫性、モジュラー性の特徴は本研究の身体図式も有している一方で、これらはロボットのモデル化としては当たり前であるため、明示的には取り扱わない。よって、多感覚性と変化適応性のみを明示的に考慮する。一方で、これまでの議論から、この身体図式は身体-道具-動作環境に関する相関複雑性と時間的変容に対応しなければならない。つまり、この身体図式のモデル化という観点では、そのモデル化困難性と逐次的モデル変化に対応しなければならないということを意味する。また、様々な身体部位や道具、タスク等を扱うことが可能な一般性にも対応する必要がある。これらの議論から、この身体図式が持たなければならない4つの特徴を以下に整理する。

- 多感覚性 - 多様な感覚間の相関を表現可能
- 汎用性 - 制御から状態推定、異常検知、シミュレーション等のロボットの基本要素を構成可能
- 自律獲得性 - 自律的な学習によりネットワーク構造を含めたモデル獲得が可能
- 変化適応性 - モデルの逐次更新により逐次的なモデル変化に対処可能

多感覚性は単純に多感覚を扱えば良いという話ではない。多数の感覚の相関関係を表現することで、あるセンサが見えない状態や、ある制御入力を使いたくない状態等にも対応できるようにする必要がある。また、自律獲得性も単に学習によりモデルを獲得するというものであってはならない。そのモデルの構造自体までもロボットが自律的に獲得することで、一般性に対応する必要がある。これらの特徴を持った身体図式をモデル化し、自律的に学習させ、環境適応能力を向上させることが本研究の目

的である。

2.2.3 身体図式とその他モデルの比較

ここでは、身体図式、一般的なロボットモデル、一般的な学習モデルを比較し、本研究の位置づけを明確にする。

まず、通常のロボットのモデルといえばロボットのリンク長や関節配置・慣性パラメータ等を含むモデルであり、人間が直接明示的な値を記述する。そのため、様々な用途に利用可能な汎用性は持ち合わせている一方で、自律獲得性や変化適応性を持たない。軸駆動型や腱駆動型等ある程度の種類のロボットには適用可能な一方で、柔軟身体や柔軟道具など、明示的パラメータ化の難しいロボットはそもそも扱うことが難しい。もちろんパラメータ同定 [32] の分野も発展している一方で、人間が明示的に考えることのできるパラメータの範疇でしかモデルを適応させることはできない。モデルとして接触センサや視覚、6軸力センサ等を含めることが可能という意味である程度の多感覚性を持つ一方で、聴覚やひずみゲージ等、扱うことができない感覚入力も多い。同時に、モデルは非常に安定しており、パラメータも明示的に指定されているため、解釈性に優れている。

次に、一般的な学習モデルについて簡単に述べる。これらは例えば強化学習や模倣学習、教師あり学習等が相当する。これらはもちろんある程度の自律獲得性は有しているが、そのネットワーク構造、つまり感覚と制御の入出力関係は人間が決定しているため、本研究における自律獲得性として十分ではない。また、変化適応性は基本的に持ち合わせていない。一度オフラインで学習した後はモデルを固定して利用することがほとんどである。多感覚性については、視覚や内部感覚だけでなく、触覚や聴覚を使った機械学習モデルも徐々に構築されてきている [33] が、その相関を表現できるものではない。また、状態推定や制御器、異常検知器等は個別に開発されており、一つのモデルにより様々なコンポーネントを実行する汎用性はない。

これらに対して、本研究の身体図式は心理学における身体図式の考え方をロボット側に転移した概念と言え、その多感覚性、汎用性、自律獲得性、変化適応性において優れた概念として、本研究ではこれを学習する機構を開発する。

2.2.4 身体図式学習に関する先行研究

これまで行われてきたロボットの身体動作学習について、Fig. 2.4 にまとめる。一般的に、学習方法は教師なし学習、教師あり学習、強化学習の3つに大別される。なお、本研究では強化学習と進化的ア

身体図式		ロボットモデル		一般的学習モデル	
感覚や制御入力間の相関を表現したモデル		リンク長や関節配置, 慣性等を含むモデル		強化学習・模倣学習など	
・ 多感覚性	○	・ 多感覚性	△	・ 多感覚性	△
・ 汎用性	○	・ 汎用性	○	・ 汎用性	×
・ 自律獲得性	○	・ 自律獲得性	×	・ 自律獲得性	△
・ 変化適応性	○	・ 変化適応性	×	・ 変化適応性	×
(心理学や神経科学)					
+ 空間知覚性・動作更新性		+ 明示性・安定性			
・ モジュール性・時空間一貫性					
・ 他者間共通性					

Fig. 2.3: The characteristics of body schema, robot model, and general learning model.

ルゴリズムを合わせて, 報酬学習という区別をしている. また, 教師あり学習・自己教師あり学習の区別については議論の余地があるが, 本研究では, 人間の明確なアノテーションが必要な場合を純粋な教師あり学習, それが必要ない場合を自己教師あり学習としている.

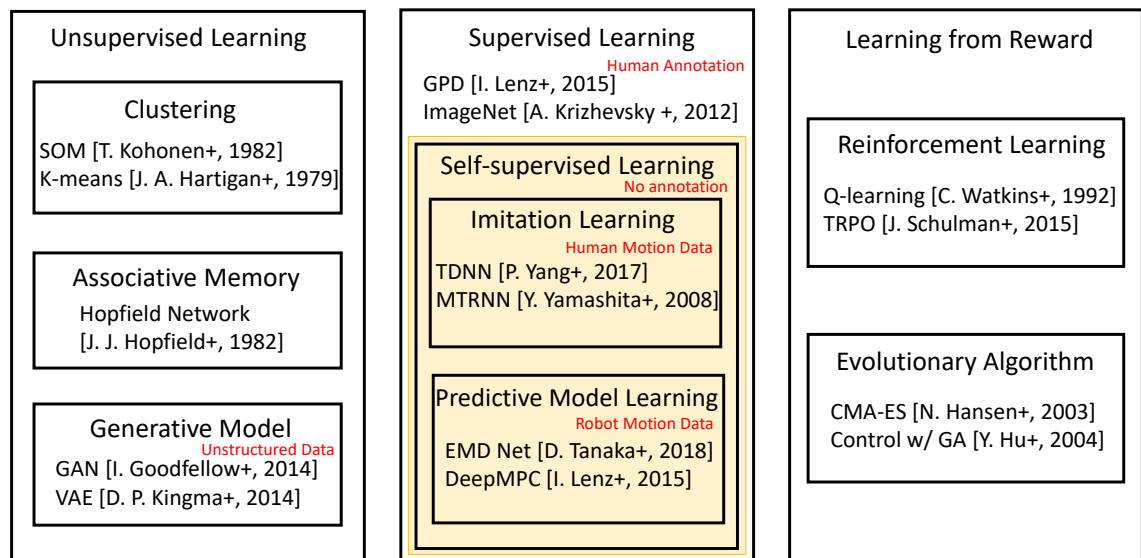


Fig. 2.4: Studies on body motion control learning of robots.

まず, 教師あり学習について述べる. 教師あり学習というと, ImageNet のような画像認識タスク [34] が最もよく見られるが, 身体動作学習に限れば, 例えば Grasping Point Detection (GPD) が良く知られている [35]. これは物体画像に対して, どの位置を把持するべきかをアノテーションすることで, 未知の物体画像に対して, どの位置に手を持ってくるべきかが分かるという手法である. しかし, これは人間

のアノテーションが必要であり、自律的な学習により相関複雑性と時間的変容に対応する身体図式概念とは程遠い。一方、教師あり学習の中にも自己教師あり学習という枠組みがある。これは、アノテーションを必要としない教師あり学習を表す。例えば、先の GPD のような把持学習の文脈では、S. Levine らによる把持成功率学習 [36] が挙げられる。ロボットが箱の中の物体を適当に把持し、その際に把持が成功したかどうかを自動で検知、物体を戻すことによって、ロボット自身が連続でデータを収集することができる。ここで、自己教師あり学習に基づくロボットの身体動作学習を、模倣学習と予測モデル学習の2つに大別する。まず、模倣学習というのは、人間の行った動作データから、ロボットがそれを模倣するように動作学習する手法である。例として、MTRNN (Multiple Timescales Recurrent Neural Network) [22] や TDNN (Time Delay Neural Network) [23] 等が存在する。これらは基本的に、現在のロボット状態や視覚状態 s_t とロボットの制御入力 u_t から、次時刻の s_{t+1} と u_{t+1} を出力する。また、この模倣学習を逆強化学習による報酬関数の推定と強化学習による動作学習により行った手法も存在する [37]。次に、予測モデル学習は、ロボットの行った動作データから、自身の身体や対象物体状態に関する予測モデルを学習する手法である。例として、DeepMPC [24] や EMD-Net (Encode-Manipulate-Decode Network) [25] 等が存在する。これらは基本的に、 s_t と u_t を入力として、 s_{t+1} を出力するようなネットワークを構成する。このネットワークを用いて、 s_{t+1} を指令状態に近づけるように動作入力 u_t を決定している。把持成功率学習 [36] も、一種の予測モデル学習と言える。

次に、報酬学習としては、強化学習・進化的アルゴリズムがよく用いられる。ラベルについては、教師あり学習のように動作に関する正解データが直接与えられるわけではなく、報酬として現在状態に対して何らかの評価値が与えられる仕組みを持つ。まず、強化学習は、環境中を動作するロボットの行動価値関数を、報酬関数を元に更新していき、その方策を徐々に改善していく手法である。報酬関数と初期方策を決め、その方策に従ってロボットが動作した際の様々な状態・行動に対する行動価値関数をベルマン方程式から更新していく。代表的なものに Q 学習 [20] や TRPO (Trust Region Policy Optimization) [21] が挙げられる。次に、進化的アルゴリズムは突然変異や交叉、自然淘汰といった進化の仕組みを元に開発されたアルゴリズムである。報酬関数(コスト関数)を定義し、これを元にそれぞれの個体(解)の生存を決定、また、個体を進化させることで動作を最適化することができる。遺伝的アルゴリズム (GA) を使った行動計画 [38] や共分散行列適応進化戦略 CMA-ES [39] によるブラックボックス最適化が例である。

最後に、教師なし学習について述べる。教師なし学習は、教師あり学習に比べて構造化されておらず、ラベルもない単一のモーダルに対して行われる学習を表す。これらにはクラスタリング、連想記憶、生成モデル等が含まれる。クラスタリングはその名の通りデータの分類を表し、自己組織化マップ [40]

や K-means [41] が挙げられる。連想記憶は一部の情報を手がかりとして、全体の情報を読みだすような機構を言い、Hopfield Network [42] が代表的な例である。生成モデルはそのデータの生成過程を暗黙的にモデル化したものであり、VAE (Variational AutoEncoder) [43] や GAN (Generative Adversarial Network) [44] 等が代表的な例である。一方、これらの教師なし学習をロボットの身体動作学習に直接使うことは難しく、事前学習や画像の圧縮等に用いられる場合が多い。

ここで、身体図式学習について考える。まず、人間のように自分自身で動きを学んでいく自律獲得性を考えると、純粋な教師あり学習はあり得ない。また、柔軟な身体や道具はシミュレーションが難しいため、実機のみでの学習が必須である。そのため、強化学習のような報酬学習は試行回数の観点から現実的ではない。教師なし学習は一部利用可能であるが、身体図式学習自体を行うことは難しい。そこで本研究では、自己教師あり学習について主に考えていく。その中でも、ロボットの動作データのみから学習可能な予測モデル学習を軸に、この身体図式学習について考える。

様々なモデルの内部表現

ここでは、身体図式の内部表現の候補について考える。主要なものに、ニューラルネットワーク、ガウス過程、関数近似、データテーブル、決定木等が存在する。

関数近似には線形近似や多項式近似等様々な近似方法が考えられるが、データ入力次元が増えるごとに関数は複雑になり、リアルタイム性は失われていくという問題がある。また、出力次元に関しては、それぞれの相関を考慮できない。データテーブルは最もシンプルな形であるが、得られたデータ数が増えるごとに、検索の計算量が線形に増加していく。ガウス過程は平均と分散を考慮できる点で関数近似と異なるものの、出力次元に関して相関を考慮できない、入力次元によって計算量が爆発する等の点は関数近似と変わらない。決定木は解釈性が高いものの、微分可能でないため、相関関係における勾配に基づいた入出力の更新等は難しい。

これらに対して、ニューラルネットワークは入出力それぞれの値の相関を表現できる点、それらに関する暗黙的な内部表現を獲得できる点で有用である。また、入出力の増加によって計算量が爆発することはなく、画像等の多次元入力も利用可能である。微分可能であるため、勾配に基づいて入出力を変化させることも可能である。一方、ガウス過程や関数近似等に比べ、必要なデータ数が増えてしまうという問題があるが、これは Batch Normalization [45] や Adam [46] 等の様々な更新則によって改善されつつある。本研究では、このニューラルネットワークを中心としたモデル化により身体図式を構築している。

なお、本研究では扱わないが、ニューラルネットワークの亜種である Transformer [47] や Reservoir

Computing [48] 等の技術を利用してモデル化を行うことも可能である。

多感覚性

多様な感覚の関係性を表現可能な多感覚性の先行研究について述べる。これまで身体の内部感覚のみを用いた手法 [49] が一般的であったが、深層学習の台頭により、視覚 [50] や聴覚 [51], 接触覚 [52] を使った学習手法が一般的になりつつある。また、接触覚と視覚を合わせて利用する手法も多い [33]。一方で、これらの手法は基本的に内部感覚や視覚、触覚等を一度に入力して End-to-End に用いる方法がほとんどである。そのため、例えば視界が悪く視覚が使えない場合に他の感覚でこれを補うような、多感覚性の利点を真に活かした手法は少ない。本研究は相関性を表現可能なマスク表現により様々な感覚の組み合わせで学習を進行するため、この多感覚性を最大限に利用していると言える。

汎用性

一つのネットワークを様々な用途に利用可能な汎用性について述べる。これまでに紹介した論文は基本的には制御や認識、シミュレーションや異常検知等、一つの目的を達成するために考えられてきた。強化学習や模倣学習はその代表例である。一方で、運動方程式型 [24] のネットワークはその性質から非常に汎用性が高い。しかし、これを陽に用いた研究はなく、これを使って制御や状態推定、異常検知、シミュレーション等を汎用的な計算手順で統一的に扱う手法が必要である。

自律獲得性

自律的な身体図式の獲得により、身体-道具-動作環境の相関複雑性に対応する先行研究について述べる。古くから柔軟リンクを持つロボットにおいてその制御誤差や振動を抑制する手法は開発されてきた [53, 54, 55]。また、機械学習の台頭によって学習型の手法も様々な開発されている [56, 57, 58]。この他にも、柔軟物体操作 [59] や柔軟アーム操作 [60] に関する研究も行われている。一方、これらの手法は問題の構造に多くの仮定を置いており、統一的に筋骨格構造や柔軟道具、柔軟物体等に利用できるようなものではない。模倣学習 [61] や強化学習 [62] を使った試みも多い一方で、実機のみでロボットが経験から学習していくことが可能な手法はほとんど存在しない。

また、様々な身体-道具-動作環境、タスクの組み合わせに対して適用可能であるという一般性、つまりネットワーク構造を含めた自律獲得性に関する先行研究を述べる。この一般性には、ニューラルネットワークの構造、その中でもネットワークの入出力に用いられる感覚入力や制御入力の与え方に一般

性を持たせ、これを自動決定する構造が必要となる。最も一般的に行われているネットワーク構造決定は、主にニューラルネットワークのパラメータ数の探索であろう。N. Murata らは、赤池情報量基準 (AIC) [63] を一般化した Network Information Criterion (NIC) を提案し、これに基づくニューラルネットワークの隠れ層のユニット数最適化について議論している [64]。近年は計算機性能の向上により、深層学習を用いた Neural Architecture Search (NAS) という分野で活発に議論が成されている。RNN により構成される探索ネットワークを使った強化学習との併用による NAS [65] を始めとして、セルの探索と転移学習を利用したより効率的な ENAS [66]、離散空間探索を連続的な空間探索に緩和することで探索時間を短縮した DARTS [67] 等が開発されてきている。一方、これらはニューラルネットワークのパラメータ探索であり、基本的にネットワークの入出力は固定されている。ゆえに、本研究で行うようなネットワークの入出力までも自動決定する手法に関する研究はほとんどないが、その一端を示す研究を紹介する。[68] はニューラルネットワークではないが、相互情報量をもとにセンサ・アクチュエータ間の関係性を抜き出し、それぞれに対して Locally Weighted Regression をかけることでヤコビアンを抽出、制御を試みている。[69] はニューラルネットワークにおける出力値と入力値の関係性を相互情報量をもとに抜き出し重要な入力のみを利用する手法、自己組織化マップにより入力の次元を削減する方法を提案している。

変化適応性

変化適応性に関する先行研究は様々であるが、逐次的なモデルの変化に関する情報を直接モデルに取り入れることはほとんどなく、大抵は Zero Moment Point のようにシンプルなモデル化を行い、外乱・環境変化に対して最大限安定化可能な制御を行う場合が多い [70]。学習制御の場合、強化学習 [71] や教師あり学習 [36] においては、様々な環境において大量のデータを収集することで適応的な動作を生み出す。つまり、動作の目的が一意に決まっている場合は様々な環境状態でデータを取ることで適応的な動作が可能となる。一方、動作の目的が一意に決まっておらず、そのモデルを使って様々な動作を行おうとした場合は、単に大量のデータを収集しても効果を得るのは難しい。時間的変容による暗黙的なダイナミクスの変化を何らかの形で変数として埋め込む必要があり、この一つの方法が Parametric Bias [72] である。本来は模倣学習において利用される仕組みであるが、これを予測モデル学習に埋め込むことも可能である。中時間変化や長時間変化については、一度学習し直して、その後問題が発生した際に学習し直すという形がほとんどであり [73]、オンラインで常に学習していくという例はほとんどない。なお、モデル化が容易である場合は、適応制御や二元制御 [74] といった制御分野の手法が有用である。二元制御は適応制御において、不確定性を考慮した目標状態への制御と、パラメータ推定を推

進する探索信号の導入という二元性を持った制御である。また、フィードバック誤差学習 [75] や、その発展系である複数モデルを競合させるアーキテクチャMOSAIC を利用した制御 [76] も存在する。これらはフィードバック制御系を予め組んでおき、そのフィードバック制御における誤差を無くすように逆モデルを学習するという制御である。MOSAIC の場合はこの逆モデルを複数用意することで、時間的変容に対処するものである。一方で、これらの適応制御系は予め推定するパラメータを設定できるようにある程度のモデル化が必要であるし、フィードバック誤差学習は制御対象に対してフィードバック制御が予め構築されていなければならない。そのため、本研究で扱うような、多感覚かつフィードバック系さえもモデル化困難な系に対しては利用できない。

2.3 柔軟性と冗長性を有するロボットの知能システム

2.3.1 ロボットにおける柔軟性と冗長性

本研究ではロボットをその柔軟性と冗長性から分類していく。この柔軟性と冗長性は、身体図式学習におけるモデル化困難性と逐次的モデル変化に密接に関わっている。この柔軟性を時間的柔軟性と空間的柔軟性の軸に、冗長性を制御的冗長性と感覚的冗長性の軸に分けて議論する。ここで、時間的柔軟性とは身体-道具-動作環境に関するモデルの変化、つまり身体図式の時間的な変化を表す。空間的柔軟性とは身体-道具-動作環境の剛体近似が難しいことを表す。感覚的冗長性とは、身体図式におけるセンサ値の集合 \mathbf{x} がそれよりも小さな次元の潜在変数 \mathbf{z} によって表現可能であることを表す。制御的冗長性とは、操作したい対象に対する制御入力次元の冗長性を表す。ロボットにおける柔軟性と冗長性の利点と欠点の分類を Fig. 2.5 に示す。

まず、時間的柔軟性の利点は成長や組み替えが容易であることであろう。もし学習により正しく扱うことができれば、足りないロボットの関節軸を一つ増やしたり、関節のモータをより強いものに変更したり、リンク長を変更したりすることが可能である。筋骨格型のロボットであれば、筋を後から付け足したり減らしたりといった操作は容易であり、一つの魅力的な特徴と言える。一方、時間的柔軟性の欠点は経年劣化や逐次状態変化、動作のばらつき等が挙げられる。これは、ゴムやバネ等を多用する柔軟な身体に特徴的であり、経年劣化等によって身体状態は常に変化しているといつて良い。軸駆動のように全身が完全に金属で構成される場合でも、劣悪な使用によりモータは変化し、金属には多少のたわみが生まれる。ロボットが扱う道具や動作する環境も常に変化し、これによっても身体図式は変化していく。また、筋骨格ヒューマノイドに特徴的だが、基本的に関節角度を測定することができないため、筋長の原点の初期化に再現性がなく、これによっても身体状態は容易に変わることが想定さ

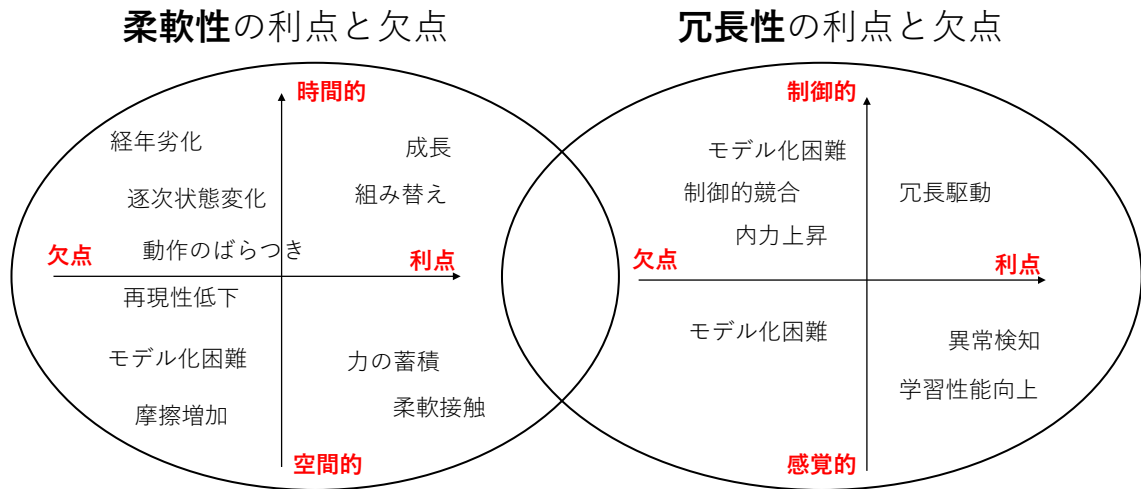


Fig. 2.5: Classification of advantages and disadvantages of robots in terms of the flexibility and redundancy.

れる。

次に、空間的柔軟性の利点は、環境接触に対する柔軟性が主であろう。この柔軟性を利用することでより素早い動作が可能になったり、これを溜めて解放することでより強い力を発揮することが可能になったりする。ハードウェアとして柔らかい身体を持つことで、安全で様々な環境に適応可能な身体構造となる。一方、空間的柔軟性の欠点は、柔軟性による再現性低下・摩擦の増加、そしてモデル化の困難性が挙げられる。剛な身体に比べて圧倒的に柔軟身体のモデル化は難しい。筋骨格型の場合には柔軟な筋と骨格の間に摩擦が発生し、ヒステリシスが生まれるため、動作の再現性が取りにくいという問題もある。また、たとえ剛であったとしても、扱う道具や動作する環境によっては剛体近似が難しく、シミュレーションと実機が大きく異なる場合も多い。

次に、制御的冗長性の利点は主に冗長駆動にある。軸駆動型では、手先の空間に対して関節が冗長に配置されている。また筋骨格型では、関節が冗長で多数の筋により駆動されるため、筋が一本破断しても動作を継続することが可能である。加えて、筋骨格型においては空間的柔軟性と制御的冗長性を合わせることで、筋骨格ヒューマノイド特有の変剛性制御が可能となり、身体剛性を状況に応じて変化させることで環境適応行動が可能である。一方、制御的冗長性の欠点は、制御的競合やこれによる内力上昇、モデル化困難性である。特に筋骨格型のように関節に対して筋が閉ループを組むような構造の場合は、意図しない制御的競合によって主動筋と拮抗筋が互いに引っ張り合い大きく内力が上昇する。

最後に、感覚的冗長性の利点は主にマルチモーダルなセンサの利用による学習性能の向上や異常検知等であろう。筋長・筋張力・筋温度・関節角度・関節トルク・接触力・視覚・聴覚・平衡感覚等多様な感覚を合わせて使用し、様々なセンサから暗黙的にその関係性を学習することでより精度の高い

学習が可能となる。また、これらを利用して異常検知や身体状態の同定をすることも可能である。一方、感覚的冗長性の欠点は、従来のようなモデルベースの手法ではモデル化が難しいという点である。様々な感覚の自明ではない相関関係を学習により獲得していく必要がある。

2.3.2 柔軟性と冗長性に基づくロボットの分類

これまでに開発されてきた様々なロボットを、その柔軟性と冗長性の観点から Fig. 2.6 に分類する。ロボットの身体について制御的・感覚的冗長性から分類を行い、身体-道具-動作環境について時間的・空間的柔軟性それぞれの観点で分類を行う。

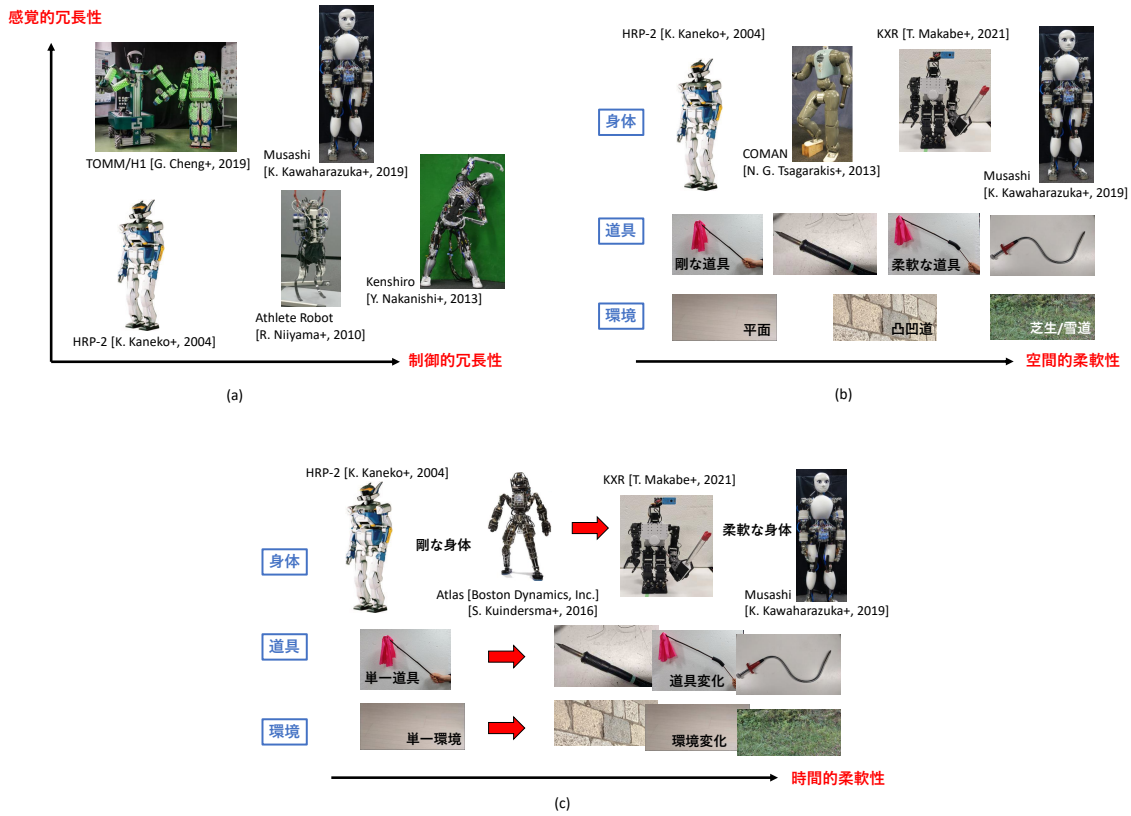


Fig. 2.6: Classification of the robots, tools and surrounding environment in terms of (a) sensor and control redundancy, (b) spatial flexibility, and (c) temporal flexibility.

冗長性に関する分類

まずは (a) 制御的・感覚的冗長性に関する分類について述べる。

最も一般的なロボットは、電気モータによって直接関節が駆動される電気軸駆動型であろう。この代表として、HRP2 [77, 3] や ASIMO [78, 79], JAXON [8] 等が挙げられる。これらはセンサとしては関節角度センサや視覚センサ、手先足先の 6 軸センサや慣性センサなどを持つ。近年は電流センサを使い関節トルクをある程度測定する方法や、関節ごとに関節トルクセンサを埋め込んだロボットも開発されている。一般的には、電気モータの先に遊星ギアやハーモニックドライブギアが接続されており、その減速比は 100 以上になる。そのため、バックドライバビリティは低く、環境からの外力に対して柔軟に対応できないという問題がある。一方、ギアの摩擦や粘弾性を補償するような制御も発展してきている [80]。これらのロボットは、その他ロボットに比べれば少ないものの、ある程度の制御的冗長性と感覚的冗長性を持ち合わせている。そして、これらの電気軸駆動型の感覚的冗長性を向上させたようなロボットも開発されてきている。例えば、全身に皮膚センサを持った TOMM や H1 などのロボット [81] がその例である。一つのロボットに 1000 近くの接触センサや近接センサ・慣性センサや温度センサを持ち、圧倒的な感覚的冗長性を持つ。

これらに対して、制御的冗長性を向上させたロボットとして、腱駆動型ロボットが存在する。腱駆動型ロボットは、これまでの関節を直接動かす軸駆動型とは違い、人間のように筋肉を模した腱/ワイヤによって駆動されるロボットである。まずこの中でも、空気圧人工筋肉によって駆動される、空圧型拮抗腱駆動について説明する。この代表例は Athlete Robot [82] や Pneumat-PB [83], buEnwa [84] であろう。これらは人間の筋配置を参考にして人工筋肉が配置されており、走行や跳躍、バランス制御等の研究がなされている。空気で駆動されるため非常に柔軟な身体である一方、制御性が悪いため、位置制御は難しい。また、スタンドアローンで動作するためには重くて大きな空圧ポンプを搭載する必要があるという問題がある。次に、電気モータとプーリにより腱/ワイヤを駆動する電気拮抗腱駆動について述べる。この代表例は ECCE [85] や Kengoro [86] が挙げられる。電気モータがギアを介してプーリに接続し、このプーリによりワイヤを駆動する。電気モータを使用しているため、空圧に比べると制御性は良い一方、ギアを介するため、ワイヤ自体に多少の柔軟性はあるものの、全体としての柔軟性は落ちてしまう。これを解決するため、ワイヤの先端に非線形弾性要素を取り付けた電気拮抗腱駆動が存在する（この代表例が Musashi [87] である）。筋は一つの関節に対して 2 本以上が作用するため、この非線形弾性要素により可変剛性制御が可能となる。人間と同様の筋骨格構造を持つため、様々な生物規範型の利点を持つ一方、非常に複雑な身体であるため、これまで制御は困難であった。センサとしては、筋長センサや筋温度センサ、筋張力センサ、関節センサ、慣性センサ、視覚センサ、張力センサ、接触力センサ等を持ち、高い感覚的冗長性を持ち合わせることが多い。本研究ではこの電気拮抗腱駆動型のヒューマノイドを“筋骨格ヒューマノイド”と呼ぶ。上記の筋骨格ヒューマノイドは関節に対する

筋のモーメントアームが一定ではなく、人間のように複雑に変化する構成を取っている。一方、関節をプーリとして、モーメントアームを一定にすることでモデル化を容易にした拮抗腱駆動型 [88, 89] も存在するが、本研究ではこれらは直接扱わない。

この他にも、制御的冗長性や感覚的冗長性は HRP-2 とほとんど同じであるが、アクチュエーション方法の異なる様々なロボットが存在する。まず、電気軸駆動型において、減速比の大きな遊星ギアやハーモニックドライブギア等を取り除いた、ダイレクトドライブ (DD) が挙げられる。この最たるものが MIT Cheetah [90] であろう。高トルク密度な電気モータと、減速比 7.67 の遊星ギアを用いることで、身体の柔らかさとスピードを実現している。この設計指針をヒューマノイドに適用したものが MIT Humanoid [91] である。減速比 6-12 の低減速比なギアを使用し、宙返り等のアクロバティックな動作が可能であるとしている。一方、減速比が低いため、現状小型のヒューマノイドしか作ることはできず、重量物の運搬等持続的負荷のかかる動作は難しい。

次に、電気軸駆動型において、モータの終端に弾性要素をつけた Series Elastic Actuator (SEA) が挙げられる。この代表例として、COMAN [92] や Baxter [93] が挙げられる。モータとリンクの間にねじりばねを配置することで、高い減速比でも柔らかい接触を可能とする。一方、常に同じ柔らかさであるため、環境やタスクによっては、柔らかすぎる・硬すぎるといった問題が起こる。これに対して、Variable Stiffness Actuator (VSA) も開発されており、これを搭載した DLR アームも存在する [94]。

次に、油圧シリンダや油圧モータを使った油圧駆動ロボットが挙げられる。この代表例は Atlas [95] や Hydra [96] であろう。これらはダイレクトドライブ並の柔らかさを保ったまま、高トルクを発揮することが可能である。Atlas は様々な不整地を二足歩行で踏破することが可能であり、その性能の高さが窺える。

空間的柔軟性に関する分類

次に (b) 空間的柔軟性に関する分類について述べる。これは身体-道具-動作環境のそれぞれについて分類している。動作環境については、平面のような完全に剛体近似ができる環境から、凸凹道、芝生や雪道のような剛体近似の難しい複雑な環境が存在する。道具についても、はたきやはさみ、ドリルのような剛な道具が存在する一方で、ゴムやバネ等で構成されたホースや紐のような道具も存在する。身体についても同様で、前述の HRP2 のような剛体近似の容易なモデル化しやすいロボットから、COMAN のようなバネを含むロボット [92]、KXR のような低剛性で樹脂製のリンク自体が曲がってしまうようなロボット [97]、Musashi のような全身が非線形弾性要素により柔軟なロボットまで様々である。

時間的柔軟性に関する分類

最後に (c) 時間的柔軟性に関する分類について述べる. これも身体-道具-動作環境のそれぞれについて分類している. 動作環境については, 単一の環境で動くようなロボットから, 複数の環境で, 環境変化を伴いながら動作するロボットも存在する. これは, 台車型や二足歩行型のような移動ロボットには必ずついてまわる特徴である. 道具についても, 単一の道具だけでなく, その曲がり方や長さ, 重さなどの異なる様々な道具を扱う場合がある. 身体については, 全身が金属で構成されるような HRP-2 や Atlas のようなロボットは経年劣化等の問題は少ない一方で, KXR や Musashi のような柔軟身体を持つロボットは経年劣化が起こりやすい. また, モジュール化によって組み換えが容易なロボットは, その身体構造を積極的に変更していく場合も存在する.

柔軟性と冗長性に関するまとめ

これまでの議論から, Fig. 2.7 に示すように, 全てのロボットは必ず制御的冗長性・感覚的冗長性・時間的柔軟性・空間的柔軟性を持つ. 通常の軸駆動型ロボットは関節の内部感覚から視覚, 6 軸センサ, 慣性センサ等を持ち, これはロボットの感覚的冗長性につながる. また, 扱う道具や動作環境は常に変化するため時間的柔軟性を持ち, それらの道具や環境は剛体近似が難しいという空間的柔軟性を生む. 関節の駆動は手先空間に対して冗長性を持つため, 制御的冗長性も持つ. そして, この軸駆動型ロボットにゴムやバネ等の柔軟要素が入った場合, 特に経年劣化等による時間的柔軟性, 非剛体性を表す空間的柔軟性が増していく. さらに複雑な筋骨格ロボットは視覚や接触覚等のセンサが増え感覚的冗長性が増し, 身体変化により時間的柔軟性, 非線形弾性により空間的柔軟性, 腱駆動により制御的冗長性が増す. つまり, この感覚的・制御的冗長性と時間的・空間的柔軟性は全てのロボットが持っており, これは身体図式のモデル化困難性と逐次的モデル変化を生む. よって, 身体図式の逐次学習機能を有する智能ロボットシステムを開発することで, ロボットのモデル化困難性と逐次的モデル変化が攻略され, 環境適応性が向上すると考えられる.

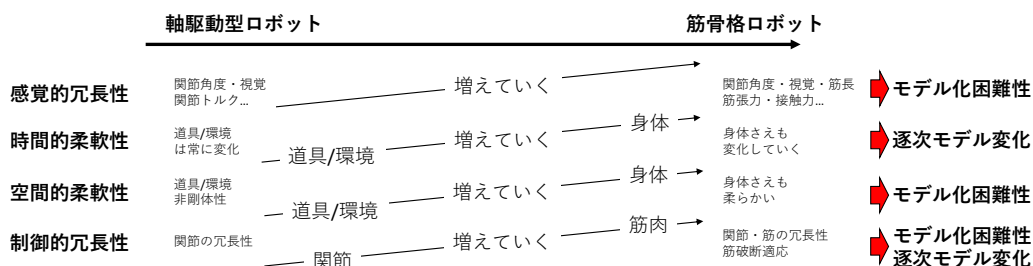


Fig. 2.7: The summarization of the flexibility and redundancy of robots.

2.3.3 柔軟性と冗長性を有するロボットの知能システム設計

これまでの議論をもとに、柔軟性と冗長性を有するロボットの知能システム設計について考える。

Fig. 2.8 に先の柔軟性と冗長性に関する利点と欠点の観点から必要なシステムの要素を示す。

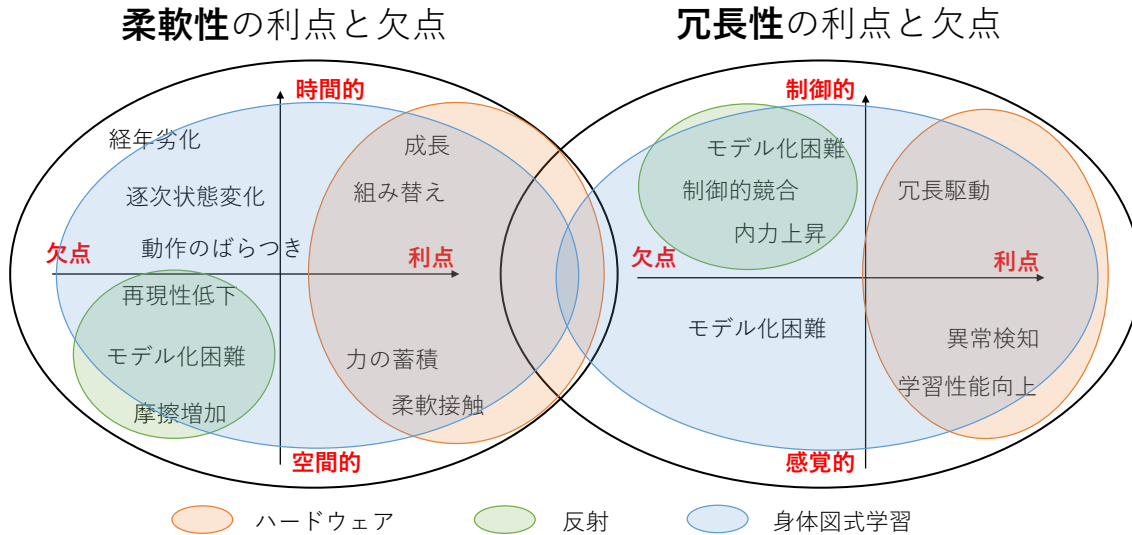


Fig. 2.8: Requirements necessary to enhance the advantages of robots in terms of the flexibility and redundancy.

本研究の主題は身体図式学習であるが、これを実機において実行するにあたって、この他にも以下の2つが必要である。

- 柔軟性と冗長性の利点を増強するハードウェア
- 空間的柔軟性と制御的冗長性の欠点を高速な周期で補完する反射制御

身体図式は柔軟性と冗長性の利点を伸ばし欠点を補完するものである一方、学習という遅い周期で実行される機能であり、モデル化誤差による問題や制御的競合による内力上昇等への素早い適応は難しい。そのため、これら空間的柔軟性と制御的冗長性の欠点を高速な周期で補完する反射制御が必要である。また、この柔軟性と冗長性の利点を活かすためには、それを実行することが可能なだけのハードウェアが必要である。以下に、このハードウェア・反射制御について詳細にまとめる。

ハードウェア

柔軟性と冗長性の利点を増強するハードウェアの開発に必要な点を以下にまとめる。なお、本研究では様々な身体構造のロボットを網羅するために、通常の電気軸駆動型や低剛性樹脂製の柔軟な軸駆

動型, 車輪型, 筋骨格型を扱う. この中で, 軸駆動型や車輪型はこれまでの研究からハードウェアの基本構成が固定されており, 既存のロボットを用いるため大きくは議論しない. 一方で, 筋骨格型はまだ普及しておらず, そのハードウェア構成も開発段階であるため, 主に筋骨格型についてそのハードウェア構成に必要な点を考える.

- 時間的柔軟性

開発するハードウェアは, 成長・組み替えが容易である必要がある. 特に筋骨格ヒューマノイドは筋モジュールや筋経路点, 骨格や関節までもモジュール化し, 容易に構造を替え, 様々な身体構造でロボットを動作させることが出来るような環境を整える. その利点を活かし, 手足のある全身筋骨格ヒューマノイドだけではなく, 足に車輪や倒立振子がついたような特殊構造も開発し, その利点を示していく.

- 空間的柔軟性

筋骨格構造において, 力の蓄積や柔軟接触ができるような非線形弾性要素を開発する必要がある. これまでの非線形弾性要素は金属とバネで構成されるため硬く, 環境接触には向かなかったが, ワイヤとゴムのみで構成可能な非線形弾性要素を構築することで, 柔軟接触行動を促進する.

- 制御的冗長性

冗長駆動が可能となるよう, 多数の筋が配置可能なモジュール化を考えなければならない. そのため, 筋の小型化や骨格と筋の一体化を目指す. また, 骨格や筋同士の接続を単一のアタッチメントによって可能にし, 筋を増やしていける構造を開発する.

- 感覚的冗長性

視覚や聴覚, 接触覚等の多数の感覚器を開発し, これを搭載する. より多くの感覚器を搭載することで, 後の身体図式学習に利用する.

反射制御

空間的柔軟性と制御的冗長性の欠点を補完する反射制御に必要な点を以下にまとめる. 関節や筋にかかる負荷を軽減する反射制御を開発する一方で, 筋骨格型のような関節と筋が閉ループをなす構造の場合は内力が大きな問題になるため, これら筋骨格構造に対する反射制御を重点的に開発する.

- 空間的柔軟性

再現性低下や摩擦増加, モデル化困難性により, 突発的に起こってしまう危険を回避する反射制御が必要である. 意図しない筋張力や筋長変化に対して適応的かつ高速に反応しなければならない.

- 制御的冗長性

制御入力が増長であることにより、制御的競合・身体の内力上昇が起こりやすいため、これを回避する反射制御が必要である。主に、主動筋に対して拮抗筋を緩める人体の相反性神経支配を参考にした反射制御が重要になる。また、モータの温度上昇を防ぐための発揮力制限制御等も低レイヤで走らせる必要がある。

2.3.4 身体図式学習

前述の柔軟性と増長の利点を活かし欠点を補うことが、2.2.2 節で述べた身体図式が持つべき4つの特徴と一致することを以下に示す。

- 時間的柔軟性による身体状態の時間変化・成長への対応 (+ 制御的増長の利点である増長駆動系による筋破断対応) - 変化適応性
- 空間的柔軟性と感覚的増長性によるモデル化困難性への対応 (+ モデル誤差と制御的増長性による内力上昇等の対応) - 多感覚性・自律獲得性
- 空間的柔軟性と感覚的増長の利点である力の蓄積や可変剛性制御, 異常検知等への活用 - 汎用性

詳細については第6章で述べるが、本研究で開発した、身体図式の具体的特徴を以下にまとめる (Fig. 2.9).

- ニューラルネットワークによりモデル化困難な身体図式を表現
- マスク表現を使い様々な感覚・制御入力の相関関係を表現
- 自己教師あり学習による自律的な実機学習
- オンライン学習・Parametric Bias 等を使った逐次的な身体図式変化への適応
- 順伝播と逆伝播の繰り返し計算による多彩な指令状態実現・身体状態推定・異常検知
- ネットワーク入出力とマスク変数の自動決定による様々な身体・道具・環境・タスクへの適用

Fig. 2.9 の左図は、ある身体状態の統合表現 z が存在し、これらと様々な感覚や制御入力 $x_{\{1,2,3,4\}}$ が互いに関係しあっている様を示す。これは展開すれば、 $x_{\{1,2,3,4\}}$ の一部 (例えば (x_1, x_2, x_3) や (x_2, x_4)) から、 z が求まる、また、 z から x 全てが求まる、ということを表す。その入力関係を、マスクである m が表現している。また、Parametric Bias p はこの身体図式の状態を表現した項であり、これを変化させることでモデルの変容をシミュレートすることができる。つまり、ネットワークの重み W または Parametric Bias p の更新により、逐次的なモデル変化への対応が可能となる。この身体図式は実機におけるデータ x から自己教師あり学習をすることができる。また、ネットワークの出力を次時刻の x とすることで、静的だけでなく、動的動作における身体図式を獲得することができる。ニューラルネットワークの微分可能性を活用し、一部のセンサデータから z を更新してこれをもとに他のセンサの状

x : redundant sensors

z : latent space of current body state

m : data mask for input sensors

p : parametric bias of implicit body state

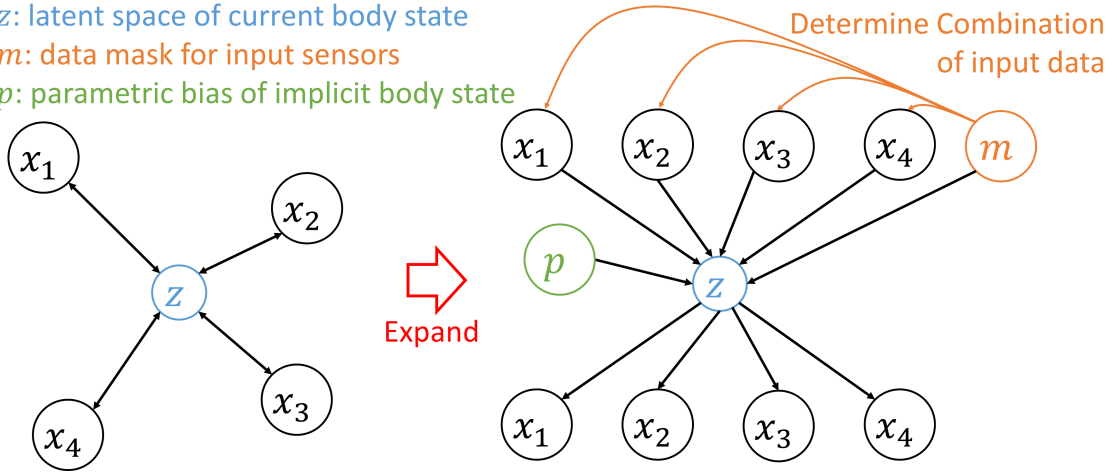


Fig. 2.9: Body schema considering the flexibility and redundancy of robots.

状態推定を行うこと, あるセンサの指令値から必要な制御入力を計算すること, 互いのセンサデータの推論誤差から異常検知を行うことが可能になる. また, 収集されたデータからこのネットワークの入出力, また, 実行可能なマスク m の集合である M を自動で決定する. これにより, 人間が身体図式のネットワーク構造を考えずとも, 得られたデータのみからこれを決定し, ロボットを制御することが可能になる. 本研究では, それぞれの部位やタスクに関して構成された身体図式が, 制御や状態推定, 異常検知, シミュレーション等に利用されると同時に, そのネットワークを切り替えながら用いることで所望のタスクを実現する.

2.4 身体図式の逐次学習機能に基づく環境適応能力の向上

開発した身体図式の逐次学習機能により, 本来の目的である智能ロボットシステムにおける身体-道具-動作環境の相関複雑性と時間的変容に対応する環境適応能力を向上させる. ここでは, この相関複雑性と時間的変容についてそれぞれ問題点とその攻略法を述べる.

2.4.1 身体-道具-動作環境の相関複雑性の分類と攻略

身体-道具-動作環境に関するモデル化困難性 (相関複雑性) を Fig. 2.10 に列挙する. ロボットの身体について考えると, 柔軟要素が含まれないほど, また, より小さなシステムであるほどモデル化は容易である. モータのようなアクチュエータ単体の電気的モデルや熱モデルは古くから確立されている.

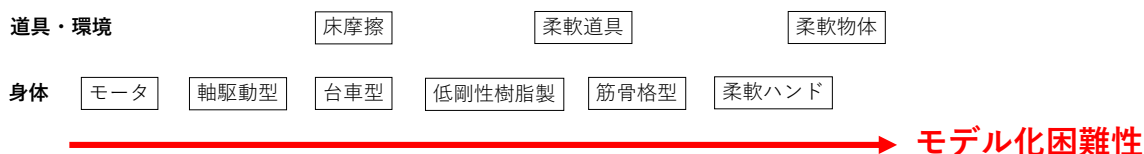


Fig. 2.10: The classification of the correlation complexity, in terms of the body, tool, and surrounding environment.

これらを合わせたような電気軸駆動型 [77, 8] の身体も比較的扱いやすい。台車型身体は基本的には柔軟性のない小さくシンプルなシステムである一方、常に環境とのインタラクションが入り、床の摩擦等の動作環境のモデル化は難しい [98]。また、車輪の滑りが確率的に起こるような場合も存在する。先の軸駆動型身体は基本的に剛体リンクによって身体が構成されるが、一方でその安さや軽さを重視した低剛性な樹脂製の軸駆動型身体も存在する [97]。これらは通常の軸駆動型ロボットに比べ、その関節やリンクのたわみから、モデル化は難しくなる。よりモデル化の難しい身体として、筋骨格型身体が挙げられる [86, 87]。ワイヤや非線形弾性要素の伸び、関節角度に応じて変化するモーメントアーム等をモデル化することは難しい。さらに、柔軟ハンドは筋骨格構造と同様なモデル化困難性に加え、扱う物体のダイナミクスも考える必要があるため、モデル化はより一層困難を極める。

身体だけではなく、道具や環境の相関複雑性についても考える。前述のように、床の摩擦等の動作環境、把持物体等はモデル化の難しい例の一つである。加えて、変形するような柔軟な道具や、布のような柔軟物体のモデル化は非常に困難であり、学習制御が必須となる。

本研究ではこれらの相関複雑性をニューラルネットワークの自己教師あり学習により獲得する。マスク表現を用いることでセンサ間の相関関係を直接学習することができる。また、ニューラルネットワークの入出力構造自体まで自律的に学習することで、定量的に必要なセンサや記述すべき相関関係を抽出できるようになる。これら身体・道具・環境に関するモデル化困難性を解消することで、これまで扱うことの出来なかった身体や道具、対象物体を制御することが可能になる。

2.4.2 身体-道具-動作環境の時間的変容の分類と攻略

身体-道具-動作環境に関する逐次的モデル変化 (時間的変容) の分類を Fig. 2.11 に示す。例えば筋骨格ヒューマノイドのような柔軟で冗長な身体における身体変化には多くの原因がある。まず、例えば再初期化による身体変化が挙げられる。これは、基本的に関節角度センサを持たない筋骨格ヒューマノイドにおいて最も頻繁に起こりうる身体変化である。筋骨格ヒューマノイドを起動する際、それぞれの筋アクチュエータの原点を初期化する必要がある。このとき、Kengoro [86] のようなロボットで

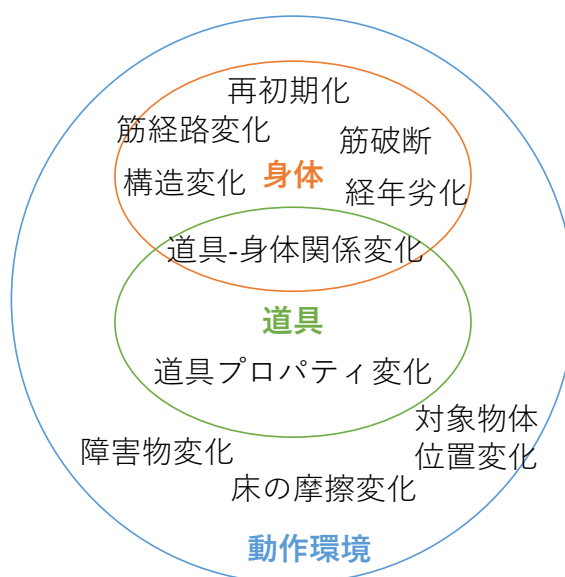


Fig. 2.11: The classification of the temporal change in terms of the body, tool, and surrounding environment.

は人間が関節角度を0になるように押さえながら、筋を巻き、原点を初期化する。Musashi [87] のような関節角度センサを持つロボットの場合も関節角度と筋長は一対一対応しないため、毎回同じ状態で初期化できるとは限らず、必ず初期化ごとに多少状態が異なる。この身体変容により、関節角度・筋張力・筋長の間の関係が変化してしまう。この他にも身体における筋経路変化、筋破断や経年劣化による身体変容は避けられない。また、成長や改善のための能動的な身体構造変化も考えられる。そして、これらは筋骨格ヒューマノイドに限った話ではなく、軸駆動型や台車型のロボットにおいても、バネの経年劣化やモータの損傷等、様々な身体変化が考えられる。次に、道具の変容が挙げられる。道具は様々な種類があり、長さが変わったり、重さが変わったり、形が変わったりと、様々な変容を起こす。そのため、その変容を認識し、適応して利用する必要がある。そして、身体と道具の関係変化も無視できない。使用しているうちに手と道具の間の角度や位置が変化し、これを修正しない限り思った通りに道具を使用し続けられない場合は多い。最後に、動作環境の変容が挙げられる。時間的変容には、周りの障害物変化や床の摩擦変化、対象物体の位置変化等の様々な原因が考えられる。これらも、筋骨格構造だけでなく、様々なロボットにおいて考慮されるべき、時間的変容による問題点である。

また、これらの時間的変容をそのスパンの違いから分類したものを Fig. 2.12 に示す。ここでは、突発的、短時間、中時間、長時間の4つの変化に分けている。突発的変化の例としては、関節の故障、筋破断や筋追加、身体構造変化等が挙げられる。これらは、身体図式のネットワーク構造やその入出力が変化し得るような大きな変化を伴う。本研究では、これをセンサマスクの導入・ネットワークの再学習

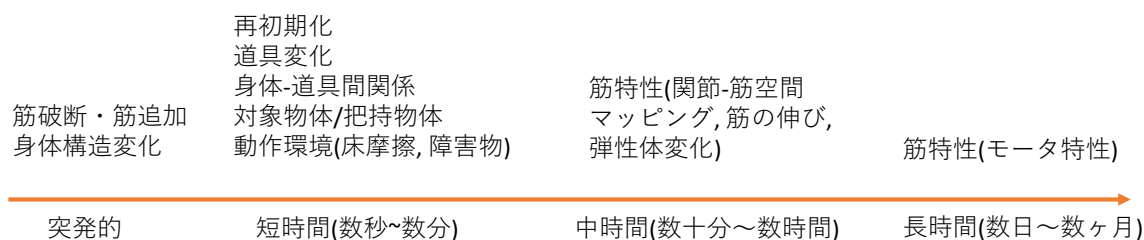


Fig. 2.12: The classification of the temporal change in terms of the span of the change.

により対処する。短時間変化としては、再初期化や道具変化、身体-道具間関係変化や把持物体・対象物体の変化、また、場所移動による床摩擦や障害物の変化が挙げられる。これらは、身体図式の一部のみの特性が変化するような場合が多い。中時間変化としては、徐々に変化するリンクのたわみや、筋ワイヤ・弾性体等の特性変化による関節-筋空間マッピングの変化が挙げられる。長時間変化としては、モータ特性のような短時間では変化しにくい筋や骨格の特性が挙げられる。本研究では、短時間変化を Parametric Bias により、中時間変化を身体図式のオンライン学習により、長時間変化を明示的なパラメータに基づく身体図式のオンライン学習により更新する。

2.5 筋骨格構造・身体図式学習に関する基礎知識

本研究の基礎知識として、一般に普及しているロボットとは異なる筋骨格ヒューマノイドの基本的な構造と特徴、その筋骨格構造の制御と状態推定、身体図式学習に用いるニューラルネットワークの基礎について順に述べる。

2.5.1 筋骨格ヒューマノイドの基本構造と特徴

基本的な筋骨格構造

Fig. 2.13 に、筋骨格ヒューマノイドの基本的な筋骨格構造について示す。筋骨格ヒューマノイドは人体を模倣した関節と筋構造を有する。骨格である骨が関節によって繋がれている。軸駆動型ロボットと違い、関節には一切のアクチュエータは入っておらず、基本的には関節角度も測ることはできない。これは、人間の肩関節や背骨のように、非常に複雑で多自由度な関節、球関節、ばねを複数繋げたような劣駆動関節等、関節角度センサを入れることができない場合が多いためである。一方、特殊な構造により関節角度を測定できるようにした筋骨格ヒューマノイドも存在する。[99] は小型カメラと LED を用いることで球関節の関節角度を測定可能とした。[87] では、球関節の利点は損なうものの、3 軸直交の擬似球関節により関節角度を測定可能としている。

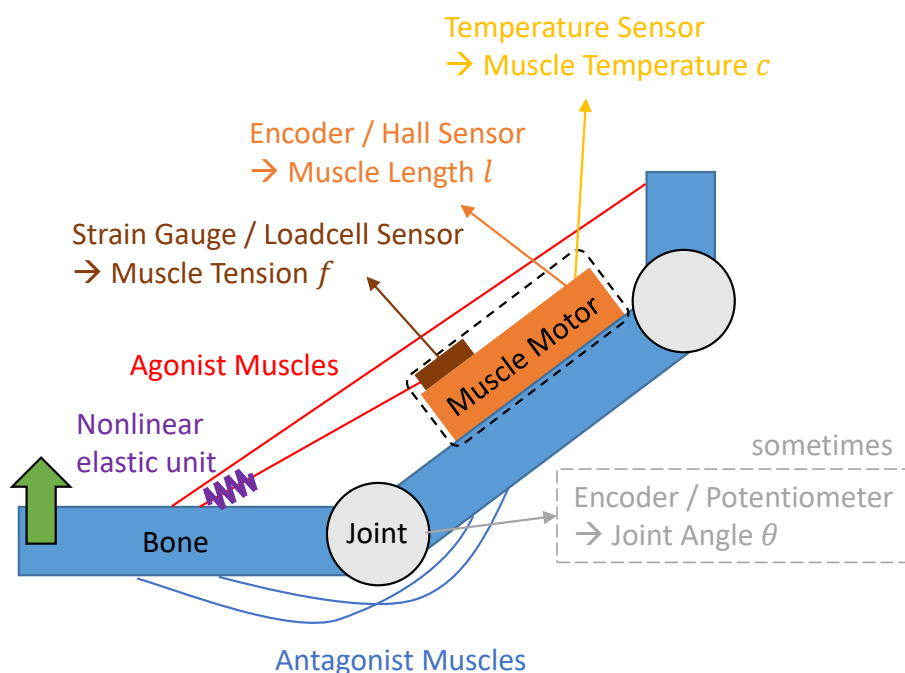


Fig. 2.13: The basic musculoskeletal structure.

関節の周りには冗長に筋が配置されており、動作方向に対して働く筋を主動筋、動作を妨げる方向に働く筋を拮抗筋と呼ぶ。筋モータに接続されたプーリが回転することで筋を巻き取る。関節が半径一定のプーリによって構成され、関節に対する筋のモーメントアームが常に一定の腱駆動方式 [88, 89] とは違い、人間のように、それぞれの筋のモーメントアームが関節角度に応じて複雑に変化する。そのため、モデル化は非常に困難である。筋ワイヤは骨格に沿うため、主に摩擦に強い化学繊維である Dyneema が使われており、このワイヤ自体に多少の弾性がある。筋の終端には基本的に非線形弾性要素が配置されており、これにより可変剛性制御が可能となる。ロボットによっては、筋のモーメントアームを増加させるために筋経由点用のプーリが追加され、それを介して筋が折り返される場合がある。また、筋の周りには環境接触のための柔軟な発泡性カバー等をつける場合もあり、よりモデル化は困難になる。それぞれの筋において、筋モータについてのエンコーダ、またはホールセンサから筋長、筋が出る位置に配置されたひずみゲージ、またはロードセルにより筋張力、筋モータに接着した温度センサにより筋温度を測定することができる。また、関節角度 θ については、前述のように基本的に測定することはできないが、筋長の変化と手先につけたマーカ等によって推定することが可能である [100] (これについては後述する)。

これまでに開発された筋骨格ヒューマノイド

これまでに開発されてきた筋骨格ヒューマノイドを Fig. 2.14 に示す. なお, 参考のため, 空圧拮抗腱駆動型のヒューマノイドについても載せている.

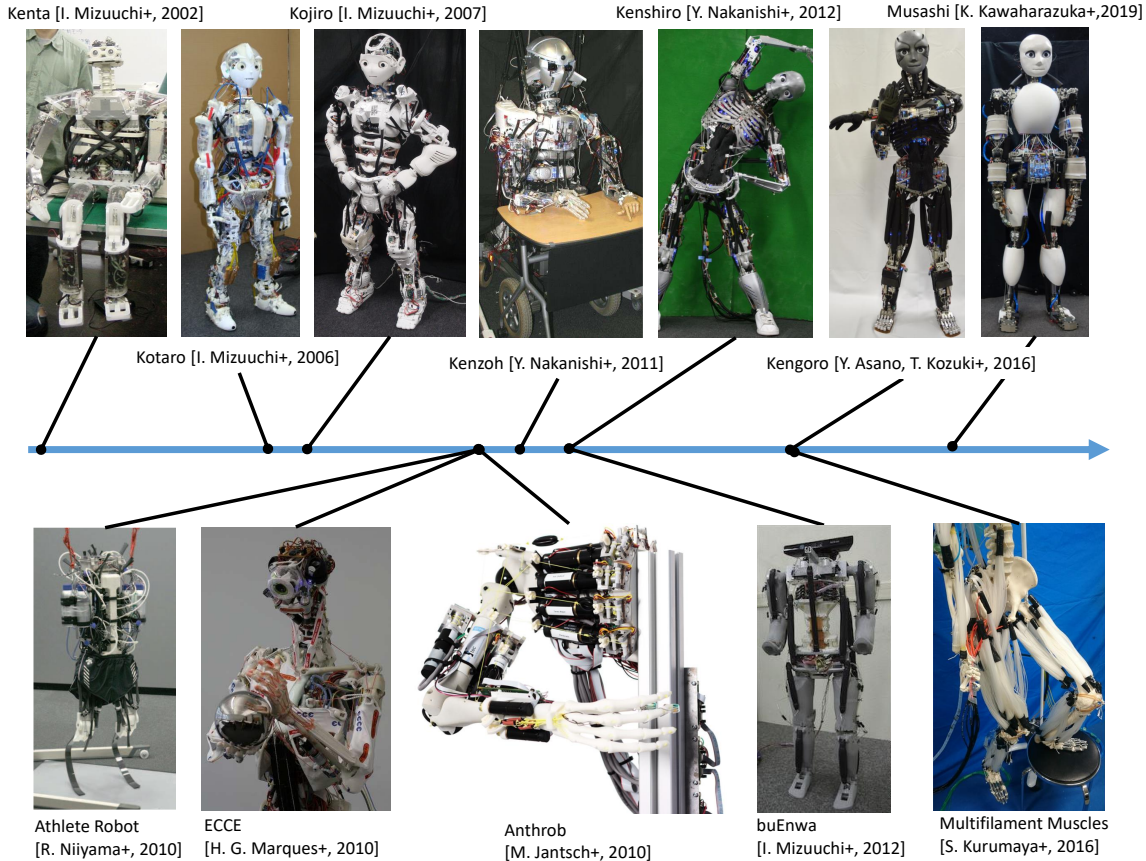


Fig. 2.14: The musculoskeletal humanoids developed so far.

Kenta [101] は 2002 年に東京大学 JSK の水内らによって発表された, 背骨を持つ世界初の等身大筋骨格ヒューマノイドである. 73 関節が 94 本の筋によって駆動され, 高さは 123 cm, 重さは 19 kg であった. 当時はまだ 3D プリンタはなく, 樹脂が切削されていた.

Kotaro [102] と Kojiro [103] は 2006 年, 2007 年に水内らにより発表された. 3D プリンタの登場により, 球状の胴体・鎖骨・肩甲骨を含む肩複合体が開発された [104]. 筋を後から付け足すというシステムも構築された [105]. また, 非線形弾性要素が追加され, 柔らかな動き, そしてその硬さを変えられる身体構造が出来上がった.

Athelete Robot [82] は 2010 年, 新山らによって発表された空圧拮抗腱駆動ロボットである. 人間の

筋活動パターンを元に制御入力を加えることで、3 m/s の走行を 8 ステップまで可能とした [106].

ECCE [85] は H. G. Marques らによって、Anthrob [107] は M. Jantsch らによって 2010 年に発表された筋骨格ヒューマノイドである。ECCE の骨格構造は熱成形プラスチックによって構築され、複雑な骨の形状を再現している。

Kenzoh [108] は 2011 年、中西らによって発表された、全身に非線形弾性要素を持つ筋骨格ヒューマノイドである。全身の非線形弾性ユニットにポテンショメータが導入され、現在の弾性を直接測定することができた。一方、重量の増加により、下肢は断念し、上肢のみが開発された。

Kenshiro [109] は 2012 年、中西らによって発表された、人体をより詳細に模倣した筋骨格ヒューマノイドである。より大きな力に耐えられるよう、これまでの 3D プリント用樹脂から素材を金属へと変更させた。肋骨のロストワックス鋳造、肩甲骨の金属 3D プリント等が大きな特徴である。

buEnwa [84] は 2012 年、水内らによって発表された空圧拮抗腱駆動ロボットである。これまでスタンダードアローンで動作するために問題だったエアータンクとコンプレッサーを、ロボット内部に搭載している。小型のコンプレッサーのみのため、まだ圧力に問題はああるものの、大きな発展と言える。

Kengoro [86] は 2016 年、浅野、上月らによって発表された筋骨格ヒューマノイドである。これまで位置制御だったモータドライバが、小型ながら電流制御に対応し、より環境接触性能に優れた身体が構築された。また、筋が完全にモジュール化され、アクチュエータ・ギア・プーリ・モータドライバ・筋張力測定ユニットが集約されることで、116 本の筋を持ちながらも信頼性の高い身体が完成した。

細径マッキベンアクチュエータ [110] は 2016 年、車谷らによって発表された空気圧アクチュエータであり、これを使ったロボットが多数開発されている。細径マッキベンを束ねることでより人間らしい筋のしなやかさを実現し、軽量で大きな力を出すことができる。

Musashi [87] は 2019 年、河原塚らによって発表された筋骨格ヒューマノイドである。人間らしい身体だけでなく、学習による脳型情報処理を目指し、身体の関節・筋・骨・筋経由点等をモジュール化し、容易な構築・再構築による実験サイクルのスピード向上を目指すと同時に、多数の冗長なセンサによる学習性能向上・学習アルゴリズムの評価機構を充実させている。本研究では、このヒューマノイド Musashi の構成法についても第 4 章で述べる。

筋骨格ヒューマノイドの特徴

筋骨格ヒューマノイドの特徴的な構造について Fig. 2.15 に示す。これまで開発されてきた構造の中でも、より人間に近く利点の多い部位を主に示している。それぞれの部位について、以下にその特徴と利点について述べていく。

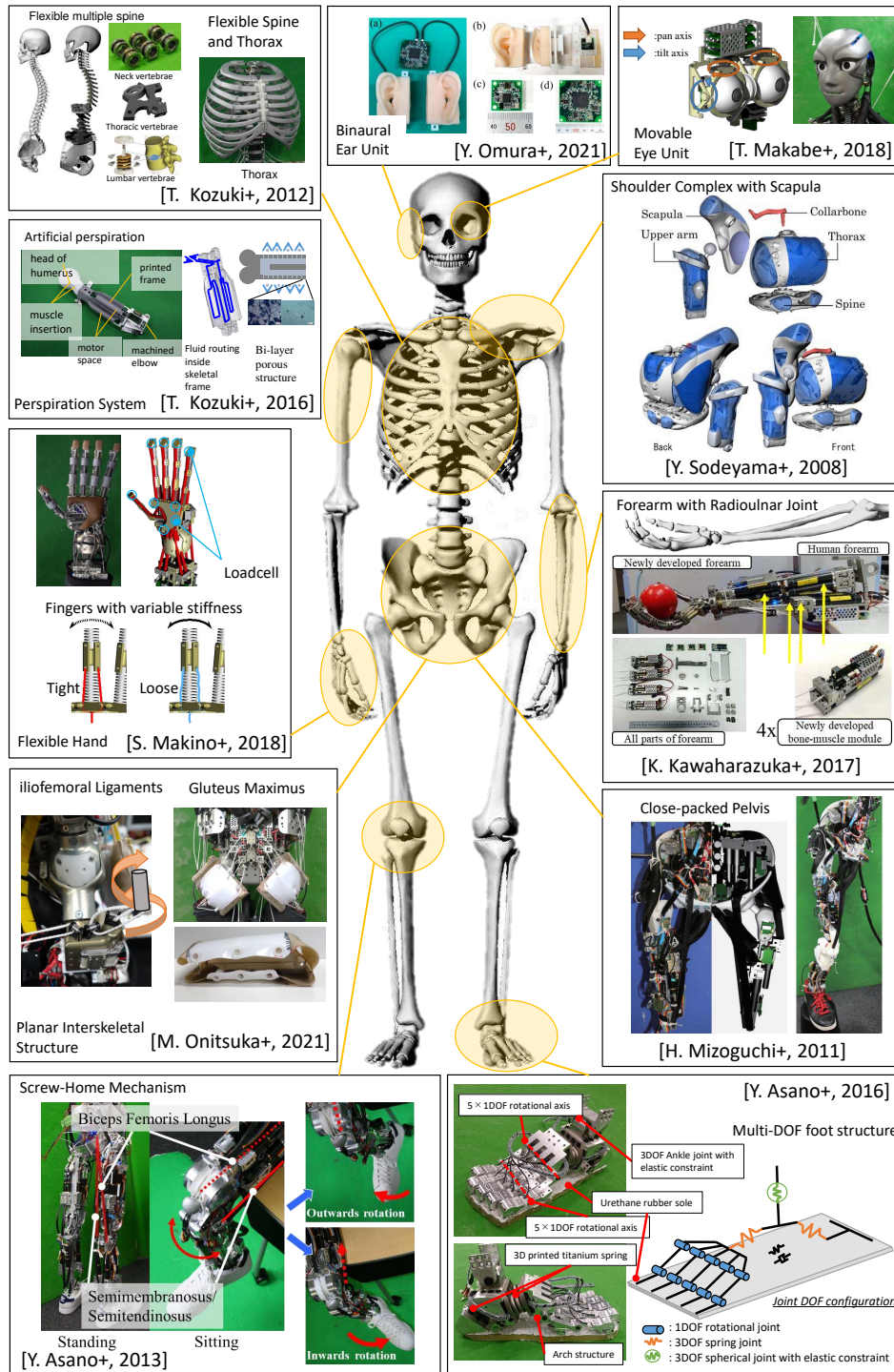


Fig. 2.15: The features of musculoskeletal humanoids.

まず、人間と同様の両耳構造は非常に特徴的である [111]. 人間に備わる外耳機構を取り付けることで、その耳の形により頭部伝達関数に変化し、上下左右方向3次元の音源定位が可能となる。マイクユニットはFPGAで構成し、高速フーリエ変換等の重い処理は耳内部で処理することで、脳への負荷を下げた。

次に、通常のロボットとは違い、人間同様に tilt と pan の関節軸のある可動眼球も特徴的である [112]. 左右の目がそれぞれ pan 軸に回転するため、輻輳による物体位置認識が可能になる。また、広い視野も特徴的である。カメラは4K画像を取得可能であり、可動眼球と合わせることで、小さな物体に焦点を合わせ、認識することが可能となる。

次に、人間同様の胸郭・鎖骨・肩甲骨・上腕を含む肩複合体も特徴的である [104]. 胸鎖関節と肩鎖関節の6自由度が胸郭との接触によって拘束されることで、擬似的な3自由度関節が成り立っている。これにより関節の可動範囲が大きく広がり、広可動域動作が可能となる。また、肩甲骨上腕関節は球関節となっており、特異点がなく、面と面の接触であるため衝撃に強い構造となっている。

次に、人間同様の背骨・肋骨構造は機能的にも見た目にも特徴的な構造である [113]. 人間の首から背骨は頸椎・胸椎・腰椎の24椎によって構成され、これと全く同じ脊椎構造をバネによって構成している。また、肋骨も人間同様の本数の金属をロストワックス鑄造によって造形している。背骨は筋に対して関節の数が多い劣駆動系を成し、環境接触の際にその環境に馴染むように変形することができる。肋骨も同様に、胸椎と合わせて変形する。

次に、人間のように汗をかく構造が一部の部位に導入されている [114]. 骨格となる多孔質金属の中に水を充填し、この骨格に筋モータを取り付ける。モータの熱によって金属内の水が外に蒸発し、その気化熱によって身体を冷やすことができる。

次に、人間同様の橈骨尺骨構造を持った前腕構造も特徴的である [115]. 筋モジュール自体を骨格に利用可能な筋骨一体小型筋モジュールを開発し、これを直列に繋げることで橈骨と尺骨を構成した。この2本の骨がねじれるように回転することで、特徴的な斜めの関節軸を生み出す。これにより、尺骨を接触させた状態での橈骨の広可動域動作やモーメントアームの上昇による高速動作が可能となっている。

次に、人間のような柔軟で多数の感覚を持つ筋骨格ハンドも特徴的である [116]. 指の関節を切削ばねによって構成することで、強い発揮力を可能としつつ柔軟さを失わない構造となっており、ハンマー等で叩いても壊れることはない。指の先や手の平にはロードセルが挿入されており、様々な点で接触力を測定することが可能である。また、指はMP関節に可変剛性構造を持っており、切削ばねの圧縮具合によって剛性を変化させることができる。

次に、骨盤は大量の筋を挿入可能な最密充填構造を持っている [117]. これにより、多数の筋配置が可能となり、最も負荷の大きな股関節や膝関節を冗長に駆動することができる。

次に、人間のように腰関節や臀部等に対して面状骨格間構造が採用されている [118]. 筋ワイヤのように線ではなく面で骨格と接触することで、安定した強度の高い関節角度リミット構造や環境との接触部位を構成することができる。特に臀部については複数の筋を束ねるようにした面状筋が構成され、さらにその中に接触センサを入れることで、お尻と環境の間の接触力を測定することが可能である。モデル化は難しい一方、環境との接触に有利な構造となっている。

次に、人間の膝に備わる終末強制回旋機構が導入されている [119]. 終末強制回旋とは、膝が伸びているときは膝の yaw 軸がロックされ、膝が曲がると膝の yaw 軸が可動可能になるという構造である。この構造により、立位姿勢では身体を安定に保ちつつも、座ったときや腰を落としたときは膝の yaw 軸を動かすことで、ペダル操作や twist squat が可能となる。

最後に、人間のように無数の骨が合体された多自由度な足構造にも特徴がある [120]. 足首・踵・つま先への接続にそれぞれ3自由度のバネ構造が、それぞれの足の指には2自由度のリンク構造が採用されている。それらリンクをゴムによる足底腱膜が接続することで、柔軟な足構造が構成され、環境に馴染む動作が可能となる。

2.5.2 筋骨格構造の基本的な制御と状態推定

筋骨格構造の基礎的な定式化

筋骨格ヒューマノイドの基礎的な定式化について述べる。基本的な変数は Table 2.1 にまとめ、それ以外については文章中で定義している。関節と筋の間には以下の関係が成り立つ。

$$l = h(\theta) \quad (2.1)$$

$$dl = G(\theta)d\theta \quad (2.2)$$

$$\tau = -G(\theta)^T f \quad (2.3)$$

この関節筋空間マッピング h は基本的に、Fig. 2.16 のような幾何モデルによって表現される。これは、筋の経路を、その起始点・中継点・終止点を線形につなぎ合わせることで構成した簡易的なモデルである。よって、骨格に対する筋の巻きつきや柔軟素材の変形等を考慮することはできず、幾何モデルと実機の誤差は大きい。なお、本研究では最終的にこの幾何モデルは身体図式によって置き換えられる。

Table 2.1: Notations in this thesis.

Notation	Definition
N	the number of joints
M	the number of muscles
θ	joint angle ($\in \mathcal{R}^N$) [rad]
τ	joint torque ($\in \mathcal{R}^N$) [Nm]
f	muscle tension ($\in \mathcal{R}^M$) [N]
l	muscle length ($\in \mathcal{R}^M$) [mm]
c	muscle temperature (motor housing) [$^{\circ}\text{C}$]
c_1	muscle temperature (motor core) [$^{\circ}\text{C}$]
c_2	muscle temperature (motor housing) [$^{\circ}\text{C}$]
G	muscle Jacobian ($\in \mathcal{R}^{N \times M}$)
\bullet^{ref}	the target value of \bullet
\bullet^{est}	the estimated value of \bullet
\bullet^{pred}	the predicted value of \bullet
\bullet^{opt}	the optimized value of \bullet
\bullet^{init}	the initial value of \bullet
\bullet_t	\bullet at the time step t
$\bullet_{t_a:t_b}$	\bullet from the time step t_a to t_b
$\bullet[t_a:t_b]$	\bullet from the time step t_a to t_b

筋剛性制御

本研究で扱う筋骨格ヒューマノイドは、以下の筋剛性制御 [121] により動作している。

$$\mathbf{f}^{ref} = \mathbf{f}_{bias} + k_{stiff}(\mathbf{l} - \mathbf{l}^{ref}) \quad (2.4)$$

ここで、 \mathbf{f}_{bias} は筋剛性制御のバイアス項、 k_{stiff} は筋剛性制御の筋剛性係数である。本来であれば、Eq. 2.4 の右辺第二項は $\max(\mathbf{0}, k_{stiff}(\mathbf{l} - \mathbf{l}^{ref}))$ である。しかし、筋を巻き出す方向に力を働かせない場合、動きにおけるヒステリシスが極端に大きくなってしまうため、現状このような形としている。 k_{stiff} が大きいほど \mathbf{l} が \mathbf{l}^{ref} に追従するものの、モデル誤差による無駄な内力等は大きくなってしまう。一方、 k_{stiff} が小さいほど内力は溜まりにくい、 \mathbf{l} が \mathbf{l}^{ref} に追従しにくくなる。

\mathbf{f}^{ref} が求まった場合、それに対応した電流が計算され、実機に送られる。よって通常ロボットを動かす手順は、関節筋空間マッピングによって θ^{ref} を \mathbf{l}^{ref} に変換し、これを Eq. 2.4 をもとに \mathbf{f}^{ref} に変換、最終的に電流によってモータを動かすことになる。

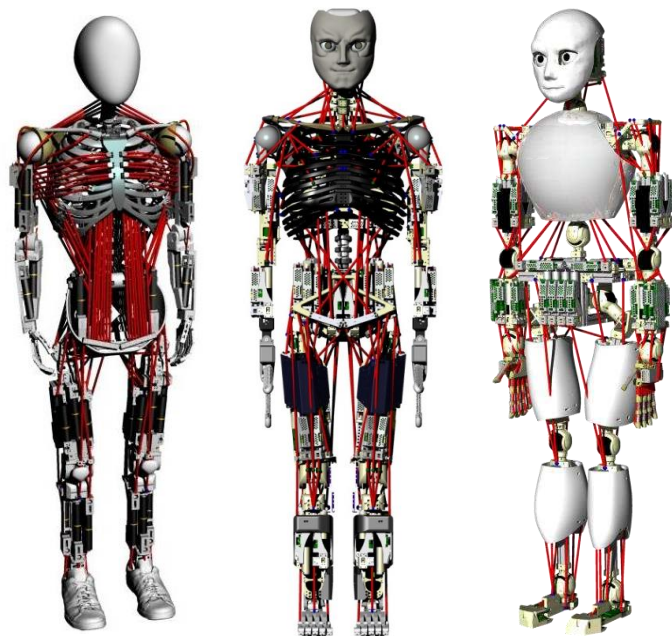


Fig. 2.16: The geometric models of musculoskeletal humanoids.

関節角度推定

2.5.1 節で述べたように、筋骨格ヒューマノイドは基本的に関節角度センサを持たない。そのため、身体状態を推定するためには、その関節角度を推定する必要がある。これを筋長の変化から拡張カルマンフィルタを用いて推定した手法 [122] について述べる。拡張カルマンフィルタの予測式と更新式は

以下のようにになっている.

Predict:

$$\Delta \mathbf{l}_t = \mathbf{l}_t - \mathbf{l}_{t-1} \quad (2.5)$$

$$\boldsymbol{\theta}_{t|t-1}^{est} = \boldsymbol{\theta}_{t-1|t-1}^{est} + G^+(\boldsymbol{\theta}_{t-1|t-1}^{est})\Delta \mathbf{l}_t \quad (2.6)$$

$$P_{t|t-1} = P_{t-1|t-1} + Q \quad (2.7)$$

Update:

$$\mathbf{e}_t = \mathbf{l}_t - \mathbf{h}(\boldsymbol{\theta}_{t|t-1}^{est}) \quad (2.8)$$

$$G_t = \left. \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t|t-1}^{est}} = G(\boldsymbol{\theta}_{t|t-1}^{est}) \quad (2.9)$$

$$S_t = G_t P_{t|t-1} G_t^T + R \quad (2.10)$$

$$K_t = P_{t|t-1} G_t^T S_t^{-1} \quad (2.11)$$

$$\boldsymbol{\theta}_{t|t}^{est} = \boldsymbol{\theta}_{t|t-1}^{est} + K_t \mathbf{e}_t \quad (2.12)$$

$$P_{t|t} = (I - K_t G_t) P_{t|t-1} \quad (2.13)$$

ここで, t がタイムステップ数, G^+ が筋長ヤコビアン of 擬似逆行列, $\Delta \mathbf{l}$ が実機の筋長変化分, P が関節角度推定値の誤差の共分散行列, Q が時間遷移に関する誤差の共分散行列, \mathbf{e} が観測残差, R が観測空間の共分散行列, S が観測残差の共分散行列, K がカルマンゲインである. これらの予測と更新を繰り返すことで, 得られた $\boldsymbol{\theta}_{t|t}^{est}$ が現在の関節角度推定値 $\boldsymbol{\theta}^{est}$ となる.

一方, ここで筋長変化から得られた推定値 $\boldsymbol{\theta}^{est}$ は筋の伸びや摩擦等, モデル化できていない項の影響を大きく受け, 実機関節角度とは誤差がある. そのため, 視覚を使った AR マーカの認識によりこの推定値を補正し, より実機に近い関節角度を推定することができる. この値は例えば, 関節-筋空間マッピングを学習するときの, 関節角度の真値として用いることも可能である. 手順は以下である. 腕の関節角度を測定したい場合, その手の先端に AR マーカ等を取り付け, その位置を \mathbf{p}_{marker} とする. 筋長 \mathbf{l} の時間的变化から, 関節角度推定値 $\boldsymbol{\theta}^{est}$ を計算する [122]. これを初期値 $\boldsymbol{\theta}_{init}$ として, 指令座標 $\mathbf{p}^{ref} = \mathbf{p}_{marker}$ に対して以下のように逆運動学を解く.

$$\boldsymbol{\theta}^{est'} = \text{IK}(\mathbf{p}^{ref} = \mathbf{p}_{marker}, \boldsymbol{\theta}_{init} = \boldsymbol{\theta}^{est}) \quad (2.14)$$

ここで, IK は逆運動学を, $\boldsymbol{\theta}^{est'}$ は視覚により補正された実機関節角度の推定値である. ここで得られた $\boldsymbol{\theta}^{est'}$ を, 実機で測定された関節角度 $\boldsymbol{\theta}$ としても用いることが可能である.

筋張力制御

関節に対する筋のモーメントアームが常に一定でありモデル化が容易な腱駆動ロボットの場合、筋張力制御が一般的である [88]. この筋張力制御を誤差の大きな筋骨格ヒューマノイドにおいても適用した例である [123] について説明する. この制御の考え方は必要な筋張力を見積もるために有用である. 一方, 本研究で扱う筋骨格ヒューマノイドにおいては, 摩擦やヒステリシスの影響から筋張力制御は難しく, 筋長制御を基本とする.

ある指令関節角度 θ^{ref} を実現する際に必要な関節トルク τ^{ref} を以下のように計算する.

$$\tau^{ref} = \text{PID}(\theta^{ref} - \theta) + g(\theta) \quad (2.15)$$

ここで, $\text{PID}(e)$ は誤差 e に対する PID 制御, $g(\theta)$ は θ において必要な重力補償トルクを表す. このとき, τ^{ref} を発揮するために必要な筋張力 f^{ref} を以下のように 2 次計画法によって求める.

$$\begin{aligned} &\text{minimize} && \mathbf{x}^T W_1 \mathbf{x} \\ &\text{subject to} && \tau^{ref} = -G^T(\theta) \mathbf{x} \\ &&& \mathbf{x} \geq \mathbf{f}^{min} \end{aligned} \quad (2.16)$$

$$(2.17)$$

ここで, W_1 は重み行列, \mathbf{x} は最適化する変数 ($\in \mathcal{R}^M$), \mathbf{f}^{min} は筋張力の最小値である. これは, 筋張力の最小値制約, 指令関節トルク合致制約を満たしつつ, 筋張力を最小化するという 2 次計画法である. ここで得られた \mathbf{x} を \mathbf{f}^{ref} とし, この \mathbf{f}^{ref} を実現するようにモータに電流を流す.

しかし, 筋骨格ヒューマノイドの場合モデル化が難しいため, 筋長やコビアン G に大きく誤差がある場合がある. G におけるある関節方向のモーメントアームが極端に小さくなると, 筋張力が求められなくなってしまうことが多い. そのため, 以下のように Eq. 2.16 を変換することで, 誤差を許容できるように変更する.

$$\begin{aligned} &\text{minimize} && \mathbf{x}^T W_1 \mathbf{x} + (G^T(\theta) \mathbf{x} + \tau^{ref})^T W_2 (G^T(\theta) \mathbf{x} + \tau^{ref}) \\ &\text{subject to} && \mathbf{x} \geq \mathbf{f}^{min} \end{aligned} \quad (2.18)$$

Eq. 2.16 における制約式を最小化の項に加えることで, 誤差を許容することができる.

関節-筋空間マッピングの学習と制御

[100] では、 θ と l の関係である h をニューラルネットワークでオンラインに学習している。身体をランダムに動かし、その際の θ と l を取得することで、このデータから h を学習させる。一方、筋張力 f の影響を無視しているため、データによっては内力が高まりやすい h が獲得される場合がある。そのため、視覚をもとに $\theta^{est'}$ と l^{ref} のデータを取得し学習させる Vision Updater と、 θ^{est} と l を使うことで筋張力の高まりを抑えながら学習する Antagonism Updater を用意し、これらにより適切な h を獲得していた。また、[122] では、これをニューラルネットワークではなく多項式近似により獲得している。

また、筋張力の影響を直接 h に反映した研究 [124, 125] にも簡単に触れる。これは、 $l = h(\theta, f)$ を学習により獲得する。これにより、非線形弾性要素やワイヤの伸びの影響が考慮され、より正確な制御が可能となる。なお、[124] では $(\theta^{est'}, f, l^{ref})$ のデータを、[125] では $(\theta^{est'}, f, l)$ のデータを用いる。このネットワークを用いる場合、制御は以下のように2段階で行う。

$$l_{soft}(f) = -(f - f_{bias})/k_{stiff} \quad (2.19)$$

$$l^{ref} = h(\theta^{ref}, f_{const}) + l_{soft}(f_{const}) \quad (2.20)$$

$$l^{ref} = h(\theta^{ref}, f) + l_{soft}(f) \quad (2.21)$$

ここで、 f_{const} はある一定の筋張力、 l_{soft} は Eq. 2.4 の影響による筋の伸びを補償する項である。まず、 f_{const} を用いて Eq. 2.20 を行い、次に現在の筋張力 f を用いて Eq. 2.21 を行う。Eq. 2.20 によりある程度の θ^{ref} を実現し、このときに発揮されている、つまり θ^{ref} の実現に必要な f を用いて Eq. 2.21 を行うことで、より正確な θ^{ref} を実現する。なお、[124] の場合は h から直接 l^{ref} が出力されるため、 l_{soft} の項は必要ない。また、前述の関節角度推定における h にこれらの学習された関節-筋空間マッピングを利用することもできる。

筋骨格構造における身体動作学習

筋骨格ヒューマノイドに限って、これまで行われてきた身体動作学習の先行研究をまとめる。ここでは、複雑な身体の状態推定、動作制御、シミュレーションについて述べる。

状態推定については、主に関節角度推定について述べる。大久保らは、慣性センサ等から得たデータを使って関節と筋の関係を多項式近似により学習し、拡張カルマンフィルタを用いて実機関節角度を推定している [122]。中西らは、関節と筋の関係を表すデータテーブルを用いてマッチングを行うことで、筋長センサから実機の関節角度を推定している [126]。大久保らの方法は、関節と筋の関係

を多項式近似だけでなく、ニューラルネットワークやガウス過程によって計算することも可能である [100, 124, 125]. つまり、 $\theta \rightarrow l$ の関係性を学習することで、カルマンフィルタを使い Δl の情報から θ を計算していく.

動作制御については、まず主に関節角度制御について述べる. 茂木らは、手先位置と筋長を対応付けるテーブルを構築し、正確な手先位置を実現する手法を提案している [127]. 水内らは、モーションキャプチャデータから関節と筋の関係をニューラルネットワークで表現し、身体制御を行っている [128]. 河原塚らは、関節と筋の関係を表すニューラルネットワークをオンラインで学習し身体制御を行う手法 [100], さらにそれを拡張し、筋張力の影響による身体組織の柔軟性を考慮した手法を開発している [124, 125]. つまり、 θ or $(\theta, f) \rightarrow l$ の関係性を学習することで、指令関節角度を指令筋長に変換する. この他にも、手先のリーチング動作やブランコ・ペダル操作等の動作学習が行われている. 手先のリーチング動作は、強化学習によって行う手法 [129], ニューラルネットワークによるフィードフォワード学習と SQP (Sequential Quadratic Programming) によって行う手法 [130], ガウス過程モデルによる予測モデル学習を使った手法 [131] が存在する. GA によるニューラルネットワークの重み調節を用いたブランコ動作学習 [132], 勾配による試行型ペダル操作 [133] も行われてきている.

シミュレーションとしては、いくつかの研究例があるが [134, 135, 136], 摩擦やヒステリシス、複雑な非線形弾性要素・Dyneema ワイヤの伸び等までをシミュレートできた研究はない.

これまで、これら状態推定や動作制御、シミュレーション等は個別に学習されてきており、統一的な手法は開発されていない. また、多項式近似やデータテーブルを用いた手法については、感覚や制御入力の増加によって指数関数的に計算量が増えてしまい、オンライン学習等は難しい. 遺伝的アルゴリズムや勾配法、強化学習等を使った場合は単一の動作目標に従って身体動作を最適化するため、汎用的に身体の動作学習を行う身体図式学習には合わない. これらは、動作目標が変わったり、身体が変化したりしたときへの対応が難点である.

2.5.3 身体図式学習に用いるニューラルネットワークの基礎

本研究で扱うニューラルネットワークの基礎的な技術について述べる. 特に、本研究の身体図式学習では誤差逆伝播を応用的に用いる. なお、本研究では主に深層学習フレームワークである Chainer [137] を使用している.

層構造

一般的な層構造である全結合層 (FC), 畳み込み層 (Conv), 逆畳み込み層 (Deconv), Batch Normalization (BN) について述べる.

まず, 最も基本の構造である全結合層 (FC) について述べる. 層 i への入力を $\mathbf{x}^{(i-1)} (\in \mathcal{R}^{N^{(i-1)}})$, $N^{(i)}$ は $\mathbf{x}^{(i)}$ の次元 (チャンネル方向の次元), 層 i における重み行列を $W^{(i)} (\in \mathcal{R}^{N^{(i)} \times N^{(i-1)}})$, バイアス項を $\mathbf{b}^{(i)} (\in \mathcal{R}^{N^{(i)}})$, 活性化関数を $\phi^{(i)}$ とすると, この FC 層は以下のように表される.

$$\hat{\mathbf{x}}^{(i)} = W^i \mathbf{x}^{(i-1)} + \mathbf{b}^{(i)} \quad (2.22)$$

$$\mathbf{x}^{(i)} = \phi^{(i)}(\hat{\mathbf{x}}^{(i)}) \quad (2.23)$$

N_{layer} 層のニューラルネットワークは, 入力 x_0 から始まり, $N_{layer} - 1$ 層の FC 層を通して, 最終的に $x_{N_{layer}-1}$ を出力する. ここでは W と \mathbf{b} の項が学習される. それぞれの層におけるチャンネル数が主なパラメータである.

次に, 畳み込み層 (Conv) について説明する. これは主に画像の特徴量を抽出するために利用され, 本研究では 2 次元配列を畳み込む場合を考える. 層 i における入力を $\mathbf{x}^{(i-1)} (\in \mathcal{R}^{N_{ch}^{(i-1)} \times N_x^{(i-1)} \times N_y^{(i-1)}})$, $N_{ch}^{(i)}$ はチャンネル方向の次元, $N_x^{(i)}$ は x 方向の次元, $N_y^{(i)}$ は y 方向の次元を表す, フィルタの重みを $W^{(i)} (\in \mathcal{R}^{N_{ch}^{(i-1)} \times N_{ch}^{(i)} \times N_{ksize}^{(i)} \times N_{ksize}^{(i)}})$, $N_{ksize}^{(i)}$ はそのフィルタのカーネルサイズを表す, バイアスを $\mathbf{b}^{(i)} (\in \mathcal{R}^{N_{ch}^{(i)} \times N_{ksize}^{(i)} \times N_{ksize}^{(i)}})$ と表すと, この Conv 層は以下のように表される.

$$\hat{\mathbf{x}}^{(i)} = \sum_j \text{Conv}(W_j^{(i)}, \mathbf{x}_j^{(i-1)}) + \mathbf{b}_j^{(i)} \quad (2.24)$$

$$\mathbf{x}^{(i)} = \phi^{(i)}(\hat{\mathbf{x}}^{(i)}) \quad (2.25)$$

ここで, $\text{Conv}(W_j^{(i)}, \mathbf{x}_j^{(i-1)})$ は特殊な構造をしている. $\mathbf{x}_j^{(i-1)}$ に対して, $N_{ksize}^{(i)} \times N_{ksize}^{(i)}$ のフィルタを, ストライド数 $N_{stride}^{(i)}$ 間隔で適用していく. また, $\mathbf{x}_j^{(i-1)}$ に対してその周囲をパディング数 $N_{padding}^{(i)}$ 分だけ 0 埋め (ゼロパディング) する. ここでは W と \mathbf{b} の項が学習される. それぞれの層におけるチャンネル数, カーネルサイズ, ストライド数, パディング数が主なパラメータである.

次に, 逆畳み込み層 (Deconv) について説明する. これは主に抽出された特徴量から画像を復元するために利用され, 本研究では 2 次元配列を畳み込む場合を考える. 層 i における入力を $\mathbf{x}^{(i-1)} (\in \mathcal{R}^{N_{ch}^{(i-1)} \times N_x^{(i)} \times N_y^{(i)}})$, フィルタの重みを $W^{(i)} (\in \mathcal{R}^{N_{ch}^{(i-1)} \times N_{ch}^{(i)} \times N_{ksize}^{(i)} \times N_{ksize}^{(i)}})$, バイアスを $\mathbf{b}^{(i)} (\in \mathcal{R}^{N_{ch}^{(i)} \times N_{ksize}^{(i)} \times N_{ksize}^{(i)}})$

と表すと、この Deconv 層は以下のように表される。

$$\hat{\mathbf{x}}^{(i)} = \sum_j \text{Deconv}(W_j^{(i)}, \mathbf{x}_j^{(i-1)}) + \mathbf{b}_j^{(i)} \quad (2.26)$$

$$\mathbf{x}^{(i)} = \phi^{(i)}(\hat{\mathbf{x}}^{(i)}) \quad (2.27)$$

ここで、 $\text{Deconv}(W_j^{(i)}, \mathbf{x}_j^{(i-1)})$ は特殊な構造をしている。まず、 $\mathbf{x}_j^{(i-1)}$ を $N_{stride}^{(i)}$ 間隔で配置し、その間をゼロパディングする。その際、 $\mathbf{x}_j^{(i-1)}$ に対しての周囲を $N_{ksize}^{(i)} - N_{padding}^{(i)} - 1$ だけゼロパディングする。その後、 $\mathbf{x}_j^{(i-1)}$ に対して、 $N_{ksize}^{(i)} \times N_{ksize}^{(i)}$ のフィルタを $N_{stride}^{(i)}$ 間隔で適用していく。ここでは W と \mathbf{b} の項が学習される。それぞれの層におけるチャンネル数、カーネルサイズ、ストライド数、パディング数が主なパラメータである。

最後に、Batch Normalization (BN) について説明する。これは学習の勾配消失・勾配爆発を防ぎ、学習速度を高めるものである。あるバッチに含まれるデータ \mathbf{x}_b ($1 \leq b \leq N_{batch}$) について、以下のような計算を行う。

$$\mu_B = \frac{1}{N_{batch}} \sum_{b=1}^{N_{batch}} \mathbf{x}_b \quad (2.28)$$

$$\sigma_B^2 = 1/N_{batch} \sum_{b=1}^{N_{batch}} (\mathbf{x}_b - \mu_B)^2 \quad (2.29)$$

$$\hat{\mathbf{x}}_b = \frac{\mathbf{x}_b - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2.30)$$

$$\mathbf{y}_b = \gamma \hat{\mathbf{x}}_b + \beta \equiv \text{BN}(\mathbf{x}_b) \quad (2.31)$$

ここで、 γ と β は学習される項である。推論時には、 μ_B と σ_B は学習時に計算された統計量を用いる。

活性化関数

一般的な活性化関数である ReLU, Tanh, Sigmoid について述べる。

ReLU は以下の式で表される [138]。

$$\text{ReLU}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x}) \quad (2.32)$$

Tanh は以下の式で表される。

$$\text{Tanh}(\mathbf{x}) = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}} \quad (2.33)$$

Sigmoid は以下の式で表される。

$$\text{Sigmoid}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} \quad (2.34)$$

更新則

一般的な更新則である確率的勾配降下法 (SGD), MomentumSGD, Adam について述べる. ここでは, 損失 L に対する重み W の勾配 $\partial L / \partial W$ を \mathbf{g} と表記する.

SGD の更新則は最も単純であり, 以下の式で表される.

$$\Delta W_t = -\eta \mathbf{g}_t \quad (2.35)$$

$$W_{t+1} \leftarrow W_t + \Delta W_t \quad (2.36)$$

ここで, η は学習率を表す.

MomentumSGD は, 前回更新時の勾配情報を用いることでより効率的な更新が可能であり, 以下の式で表される [139].

$$\Delta W_t = \mu \Delta W_{t-1} - (1 - \mu) \eta \mathbf{g}_t \quad (2.37)$$

$$W_{t+1} \leftarrow W_t + \Delta W_t \quad (2.38)$$

ここで, μ と η は定数であり, Chainer のデフォルトは $\mu = 0.9$, $\eta = 0.01$ である.

Adam は次元ごとに学習率を調整する RMSprop の改良版であり, 以下の式で表される [46].

$$\mathbf{m}_t = \rho_1 \mathbf{m}_{t-1} + (1 - \rho_1) \mathbf{g}_t \quad (2.39)$$

$$\mathbf{v}_t = \rho_2 \mathbf{v}_{t-1} + (1 - \rho_2) \mathbf{g}_t^2 \quad (2.40)$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \rho_1^t} \quad (2.41)$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \rho_2^t} \quad (2.42)$$

$$\Delta W_t = -\frac{\eta}{\sqrt{\hat{\mathbf{v}}_t + \epsilon}} \hat{\mathbf{m}}_t \quad (2.43)$$

$$W_{t+1} \leftarrow W_t + \Delta W_t \quad (2.44)$$

ここで, $\eta, \rho_1, \rho_2, \epsilon$ は定数であり, Chainer のデフォルトは $\eta = 0.001$, $\rho_1 = 0.9$, $\rho_2 = 0.999$, $\epsilon = 10^{-8}$ である.

誤差逆伝播

本節では, 誤差逆伝播 [140], 通時的誤差逆伝播 [141] について述べる.

まず、通常の誤差逆伝播について考える。3層のニューラルネットワークを例に挙げて説明する。順伝播計算は以下のように計算可能である。

$$\hat{\mathbf{x}}^{(1)} = W^{(1)}\mathbf{x}^{(0)} + b^{(1)} \quad (2.45)$$

$$\mathbf{x}^{(1)} = \phi^{(1)}(\hat{\mathbf{x}}^{(1)}) \quad (2.46)$$

$$\hat{\mathbf{x}}^{(2)} = W^{(2)}\mathbf{x}^{(1)} + b^{(2)} \quad (2.47)$$

$$\mathbf{x}^{(2)} = \phi^{(2)}(\hat{\mathbf{x}}^{(2)}) \quad (2.48)$$

ここで、 $\mathbf{x}^{(i-1)}$ は i 層における入力、 $W^{(i)}$ は重み、 $b^{(i)}$ はバイアス項、 $\phi^{(i)}$ は活性化関数を表す。損失関数 h_{loss} を用いて損失 $L = h_{loss}(\mathbf{x}^{(2)})$ を計算する。ここで W の勾配法による更新に必要なのは、 $\partial L / \partial W$ である。連鎖率を用いることで $\partial L / \partial W^{(2)}$ は以下のように計算することができる。

$$\frac{\partial L}{\partial W^{(2)}} = \left(\frac{\partial L}{\partial \mathbf{x}^{(2)}} \otimes \frac{\partial \mathbf{x}^{(2)}}{\partial \hat{\mathbf{x}}^{(2)}} \right) \cdot \frac{\partial \hat{\mathbf{x}}^{(2)}}{\partial W^{(2)}} \quad (2.49)$$

$$\frac{\partial L}{\partial W^{(2)}} = \left(\frac{\partial L}{\partial \mathbf{x}^{(2)}} \otimes \phi'^{(2)}(\hat{\mathbf{x}}^{(2)}) \right) \cdot \mathbf{x}^{(1),T} \quad (2.50)$$

$$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} \cdot \mathbf{x}^{(1),T} \quad (2.51)$$

$$\delta^{(2)} = \frac{\partial L}{\partial \mathbf{x}^{(2)}} \otimes \phi'^{(2)}(\hat{\mathbf{x}}^{(2)}) \quad (2.52)$$

ここで、 \otimes はアダマール積を表す。また、 $\partial L / \partial W^{(1)}$ は以下のように計算することができる。

$$\frac{\partial L}{\partial W^{(1)}} = \left(\left(\frac{\partial L}{\partial \hat{\mathbf{x}}^{(2)}} \cdot \frac{\partial \hat{\mathbf{x}}^{(2)}}{\partial \mathbf{x}^{(1)}} \right) \otimes \frac{\partial \mathbf{x}^{(1)}}{\partial \hat{\mathbf{x}}^{(1)}} \right) \cdot \frac{\partial \hat{\mathbf{x}}^{(1)}}{\partial W^{(1)}} \quad (2.53)$$

$$\frac{\partial L}{\partial W^{(1)}} = \left(W^{(2),T} \delta^{(2)} \otimes \phi'^{(1)}(\hat{\mathbf{x}}^{(1)}) \right) \cdot \mathbf{x}^{(0),T} \quad (2.54)$$

$$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} \cdot \mathbf{x}^{(0),T} \quad (2.55)$$

$$\delta^{(1)} = W^{(2),T} \delta^{(2)} \otimes \phi'^{(1)}(\hat{\mathbf{x}}^{(1)}) \quad (2.56)$$

つまり、層 i について以下が成り立つ。

$$\frac{\partial L}{\partial W^{(i)}} = \delta^{(i)} \cdot \mathbf{x}^{(i-1),T} \quad (2.57)$$

$$\delta^{(i)} = W^{(i+1),T} \delta^{(i+1)} \otimes \phi'^{(i)}(\hat{\mathbf{x}}^{(i)}) \quad (2.58)$$

よって全ての層の重みについて勾配が計算でき、前述した SGD 等により W を更新していくことが可能である (b についても同様である)。

ここで重要であることは、連鎖率を用いることで、 W や b だけでなく、どの項に対しても勾配は計算できるということである。これまでの式から分かるように、 $\partial L / \partial \mathbf{x}^{(i)}$ を求めることは非常に簡単であ

る。ゆえに、任意の損失関数を定義し、 W や b を固定した状態で $x^{(0)}$ を更新する、 b のみを更新するといったことを自由自在に行うことができる。この入力 x への誤差逆伝播を利用した研究は多くはないが、主な研究に Adversarial Examples [142] がある。これは、画像の分類タスクを騙すために如何に画像にノイズを加えるべきかを示唆している。つまり、間違っただけに分類するような損失関数 h_{loss} を定義し、ネットワークの入力である画像 x に対する勾配を計算していることに相当する。この他に、EMD Net [25] がこの性質を利用している。これは布を畳むタスクに対して、現在の布の状態と制御入力を出力して布の次状態を出力するネットワークを構築し、指令の布状態を実現するように制御入力に対して誤差逆伝播を行うという手法である。

次に、通時的誤差逆伝播について述べる。3層のリカレントニューラルネットワーク (RNN) を例に挙げて説明する。順伝播計算は以下ようになる。

$$\hat{z}_t = Ux_t + Wz_{t-1} + b \quad (2.59)$$

$$z_t = \phi(\hat{z}_t) \quad (2.60)$$

$$y_t = \sigma(Vz_t + c) \quad (2.61)$$

ここで、重みを $\{U, V, W\}$ 、バイアスを $\{b, c\}$ 、活性化関数を $\{\phi, \sigma\}$ とする。また、ステップ t における入力を x_t 、潜在変数を z_t 、出力を y_t とする。損失関数 h_{loss} を用いて損失 $L = h_{loss}(y_{t-N_{step}+1:t})$ を計算する ($y_{t-N_{step}+1:t}$ は $[t - N_{step} + 1, t]$ における y 、 N_{step} は RNN の時系列展開数を表す)。通常の誤差逆伝播と同様の議論から、 $\partial L / \partial U$ 、 $\partial L / \partial V$ 、 $\partial L / \partial W$ は以下のように計算することができる。

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial \hat{z}_t} \frac{\partial \hat{z}_t}{\partial U} = \delta_{z,t} x_t^T \quad (2.62)$$

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial V} = \delta_{y,t} z_t^T \quad (2.63)$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{z}_t} \frac{\partial \hat{z}_t}{\partial W} = \delta_{z,t} z_{t-1}^T \quad (2.64)$$

$$\delta_{y,t} = \frac{\partial L}{\partial \hat{y}_t} = \frac{\partial L}{\partial y_t} \otimes \sigma'(\hat{y}_t) \quad (2.65)$$

$$\delta_{z,t} = \frac{\partial L}{\partial \hat{z}_t} = V^T \delta_{y,t} \otimes \phi'(\hat{x}_t) \quad (2.66)$$

ここで、 z_t は z_{t-1} に依存しているため、 U と W に関しては過去まで遡って誤差を伝播する必要がある。 $\delta_{z,t}$ と $\delta_{z,t-1}$ の関係を捉えることができれば、それぞれのステップにおける $\frac{\partial L}{\partial U}$ と $\frac{\partial L}{\partial W}$ を計算し、これを合計することで過去を含めた勾配を計算することができる。よって、SGD の場合以下のように

計算を行う.

$$\delta_{z,t-1} = \frac{\partial L}{\partial \hat{z}_t} \cdot \frac{\partial \hat{z}_t}{\partial \hat{z}_{t-1}} = W^T \delta_{z,t} \otimes \phi'(\hat{z}_{t-1}) \quad (2.67)$$

$$U \leftarrow U - \eta \sum_{\tau=0}^{N_{step}-1} \delta_{z,t-\tau} \mathbf{x}_{t-\tau}^T \quad (2.68)$$

$$V \leftarrow V - \eta \delta_{y,t} \mathbf{x}_t^T \quad (2.69)$$

$$W \leftarrow W - \eta \sum_{\tau=0}^{N_{step}-1} \delta_{z,t-\tau} \mathbf{z}_{t-\tau-1}^T \quad (2.70)$$

2.6 本章のまとめ

本研究の全体像を Fig. 2.17 にまとめる. 本研究は知能ロボットにおけるその身体-道具-動作環境の相関複雑性と時間的変容を攻略する環境適応能力の向上に向けた身体図式の逐次学習機能を開発する. これまで多くのロボットが開発されてきたが, それらは全て時間的・空間的柔軟性, 制御的・感覚的冗長性を持つ. この柔軟性と冗長性の利点を活かすハードウェアと, 空間的柔軟性と制御的冗長性の欠点を補う反射制御を開発する. そして, 多感覚性・汎用性・自律獲得性・変化適応性を持った身体図式を逐次的に学習することで, 軸駆動型・車輪型・柔軟軸駆動型・筋骨格型等のロボットの環境適応能力を格段に向上させることが, 本研究の目的である.

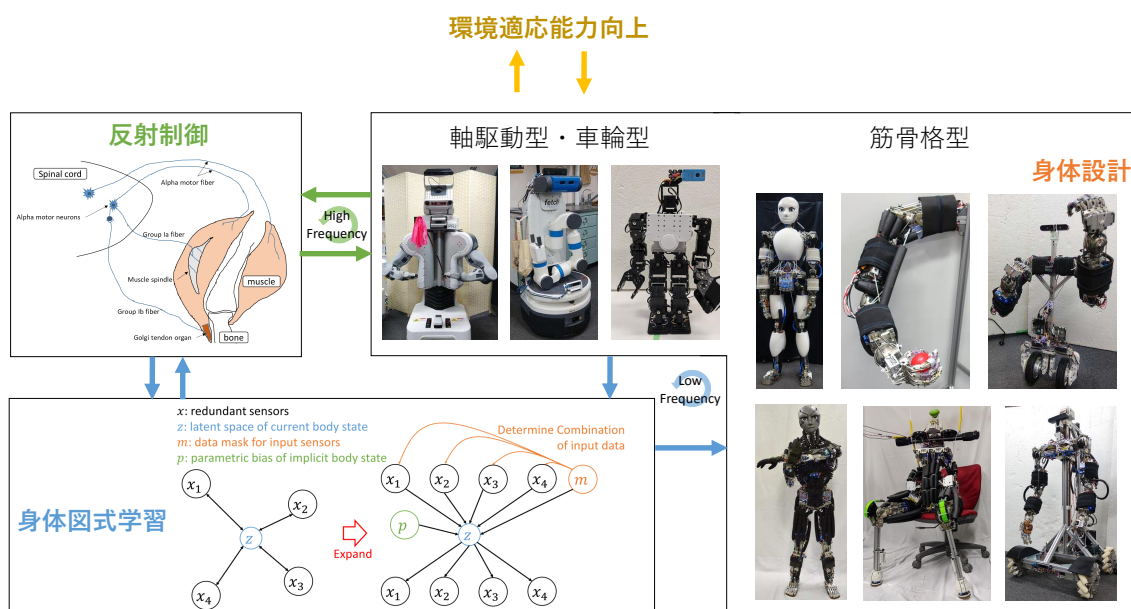


Fig. 2.17: Development of hardware, reflex, and body schema learning for various robots with the flexibility and redundancy.

第3章

身体図式の逐次学習システム

本章の概要を Fig. 3.1 に示す。本章ではまず、3.1.1 節において、センサ・アクチュエータの相互関係を記述する身体図式として、相関関係を表現するマスク変数 m と暗黙的状态表現を可能にする Parametric Bias p を導入した一般化多感覚相関モデルについて述べる。これは静的身体図式と動的な身体図式に分けられ、また、これらから構築し得る基礎的な4種類のネットワーク構成が考えられる。次に、3.1.2 節では、ネットワーク入出力に対する順伝播と誤差逆伝播を応用した6つの基礎的な操作について述べ、3.1.6 節ではこれを利用した最適化計算、3.1.7 節-3.1.10 節ではこれを応用した状態推定・制御・シミュレーション・異常検知について述べる。3.1.3 節では身体図式学習に向けたデータ収集、3.1.4 節では、身体図式ネットワークの訓練、3.1.5 節ではネットワーク重み W または Parametric Bias p の更新について述べる。最後に、3.2 節において、この身体図式ネットワークの入出力を自動決定し、ネットワーク構造の構築から学習、制御や状態推定までを自律的行う仕組みを提案する。

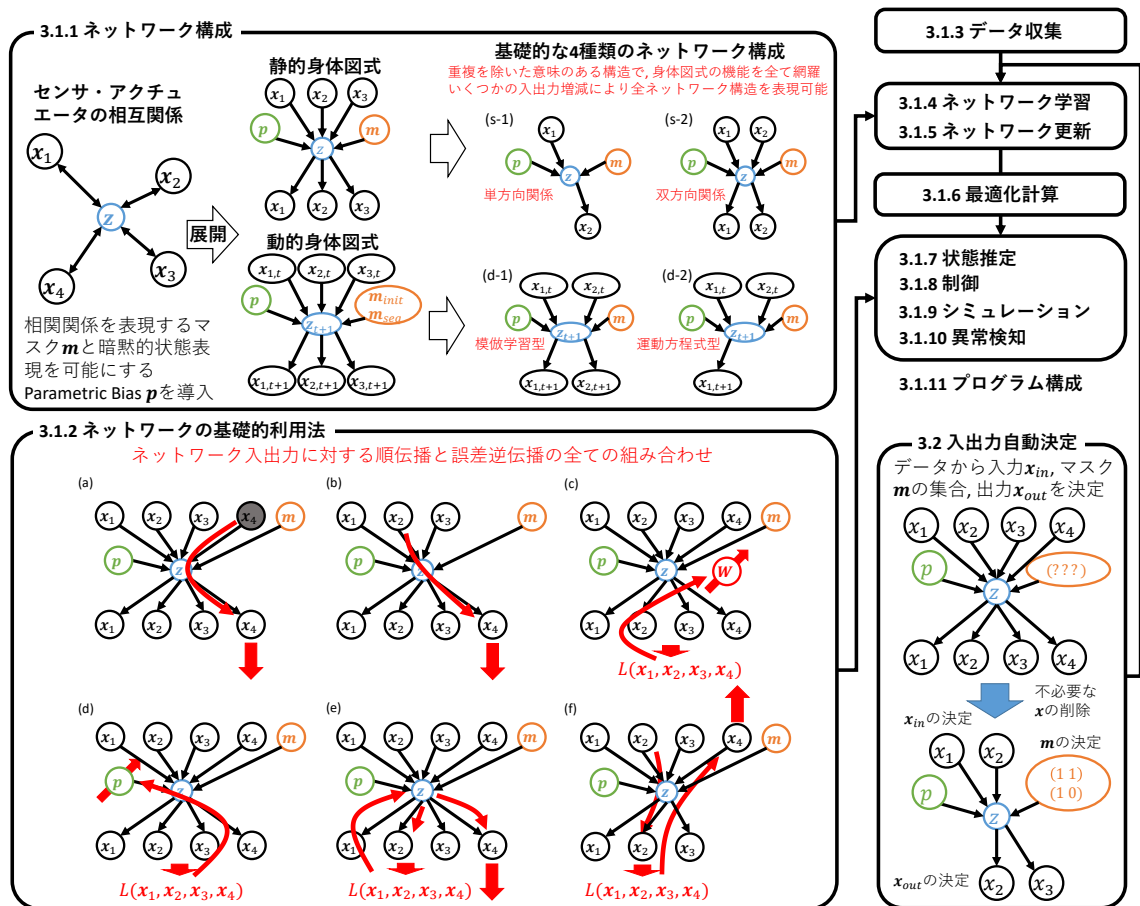


Fig. 3.1: The overview of this chapter.

3.1 身体図式の逐次学習システム - 一般化多感覚相関モデルの学習と利用

本研究における身体図式学習の概要を Fig. 3.2 に示す. 本研究で構築した身体図式を一般化多感覚相関モデル (Generalized Multisensory Correlational model, GelMiCo) と呼び, このモデルを中心としたシステムを構築する. まず, ロボット実機から多様な感覚と制御入力データを収集し, これをもとに GelMiCo を学習させる. また, 実際の運用時についても, 常に感覚と制御入力データを取得し, これをもとに GelMiCo の一部または全部を更新し続けることを行う. ある指令状態が与えられた場合, これをもとに GelMiCo から制御入力を計算し, 実機に指令する. 常に実機の感覚と制御入力を監視し, GelMiCo により互いの値を推論しあうことで異常検知を行い, これを緊急停止や制御・状態推定の変更に用いる. 一部の感覚と制御入力から, 現在のロボットの潜在状態を計算し, 直接得られない感覚を状態推定する. GelMiCo を使い制御入力による実機動作のシミュレーションを行い, これを学習制御等に利用する.

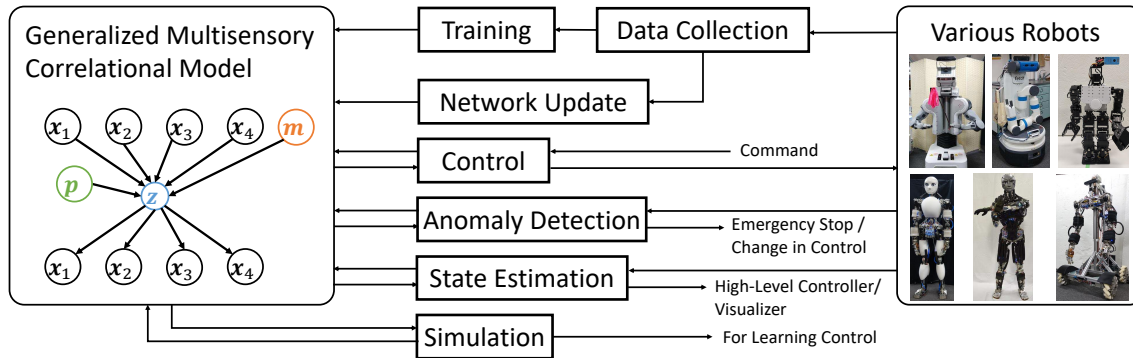


Fig. 3.2: The overview of body schema learning system.

3.1.1 ネットワーク構成

本研究における身体図式である GelMiCo のネットワーク構成を Fig. 3.3 に示す. まず, ロボットには現在の感覚や制御入力を表現可能な潜在状態 z が存在すると考える. つまり, 存在する4種類の感覚や制御入力を $\mathbf{x}_{\{1,2,3,4\}}$ と表現したとき, この z によって $\mathbf{x}_{\{1,2,3,4\}}$ の全ての値が推論できるということを意味する. また, この z は $\mathbf{x}_{\{1,2,3,4\}}$ のうちの一部, または全部を使って推論することができる. これは, z を介してそれぞれの感覚や制御入力が互いに相関を持っていることを意味する. 一方, 実際にこのモデルを作成することを考えた時, このままではデータ構造としての構築が難しい. そのため, この

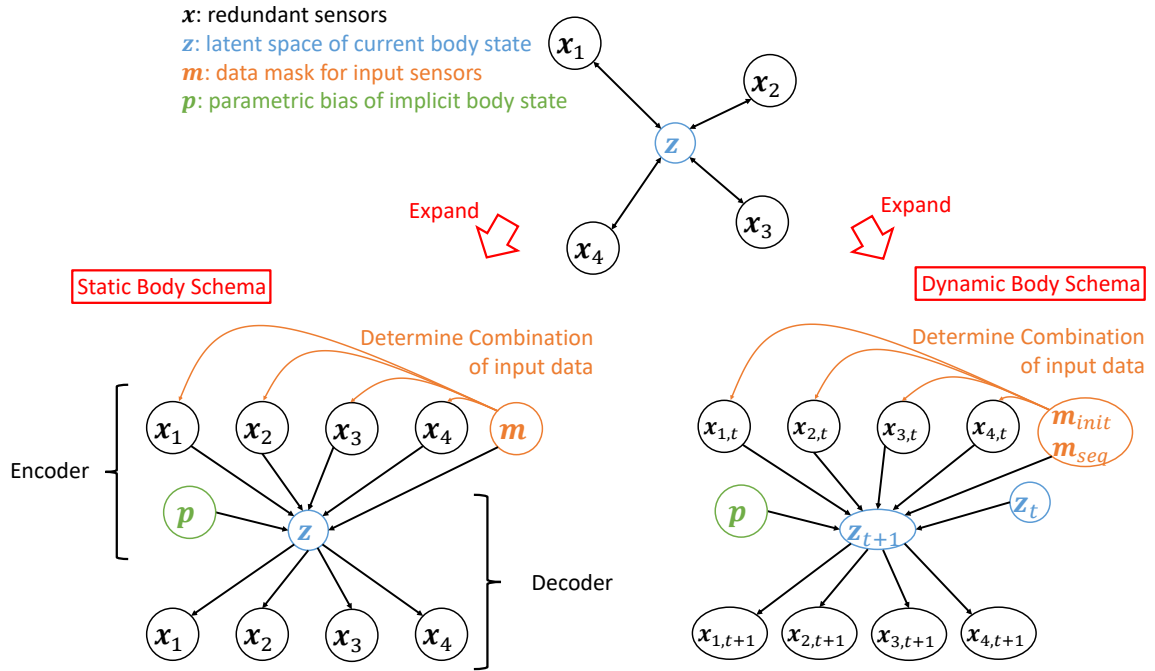


Fig. 3.3: The network structure of the generalized multisensory correlational model.

モデルを展開し、入力が $x_{\{1,2,3,4\}}$ とマスク変数 m 、中間層が潜在表現 z 、出力が $x_{\{1,2,3,4\}}$ であるモデルを考える。ここで、入力から中間層を求めるまでのネットワークを Encoder、中間層から出力を求めるまでのネットワークを Decoder を呼ぶこととする。全体のモデルの関数を h 、Encoder の関数を h_{enc} 、Decoder の関数を h_{dec} とする。 m は入力である $x_{\{1,2,3,4\}}$ をマスクする役目を持ち、 m は x が N_{sensor} 種類ある場合に、 $m \in \{0, 1\}^{N_{sensor}}$ となる変数である。 m の i 行目が 0 である場合、 $x_i = 0$ とし、完全にマスクする。一方、1 である場合は現在の値 x_i をそのまま入力として用いる。つまり、 m によって入力をマスクし、限られた入力から z を計算することになる。これにより、限られた感覚や制御入力から z を計算して、マスクされた値を推論し、状態推定や異常検知に利用することができるようになる。もちろん、どのような m でも良いわけではなく、可能な m の集合 M を保持しておく必要がある。また、潜在空間 z に情報が集約されることで、Encoder 側により z を計算できるだけでなく、Decoder 側のみを用いて z を更新することも可能である。つまり、現在の z から x を推論し、これを現在の x に合うように損失関数を設定、誤差逆伝播と勾配法により z を更新していくということが可能である。

また、特徴的な構造として、 p の Parametric Bias (PB) [72, 143] が入力に与えられている。これは、主に模倣学習において利用されていた構造であり、得られたデータから複数のアトラクターダイナミクスを抽出する目的で、認知ロボティクス研究に使われてきた。マルチモーダルな感覚からの物体ダイ

ナミクスの抽出 [144], 物体操作ダイナミクスの違いの抽出 [145, 146], 道具把持による手先ダイナミクスの変化の抽出 [147] 等が行われている. 一方, 本研究では模倣学習の文脈で直接用いることはない. Parametric Bias に身体や道具, 環境の変化の情報を埋め込み, これを現在の状態に応じて更新することで環境に適応していく.

この GelMiCo であるが, 入力と出力における \mathbf{x} が同じ場合は, 静的な身体図式しか扱うことができない. 動的な要素を含む身体図式, つまり時間発展する身体図式を扱う場合は, 入力を \mathbf{x}_t , 出力を \mathbf{x}_{t+1} とするネットワークとなる (t は現在のタイムステップを表す). つまり, 同様に存在する 4 種類の感覚や制御入力を $\mathbf{x}_{\{1,2,3,4\}}$ と表現したとき, GelMiCo の入力は $\mathbf{x}_{\{1,2,3,4\},t}$, \mathbf{m} , \mathbf{z}_t , 中間層は \mathbf{z}_{t+1} , 出力は $\mathbf{x}_{\{1,2,3,4\},t+1}$ となる. 入力に時刻 t における潜在表現 \mathbf{z}_t を加えることで, 過去の身体状態に関する情報が埋め込まれ, 動的な身体図式を扱うことが可能になる. なお, 後に詳しく述べるが, このときマスクは \mathbf{m}_{init} と \mathbf{m}_{seq} の 2 つを考える必要がある.

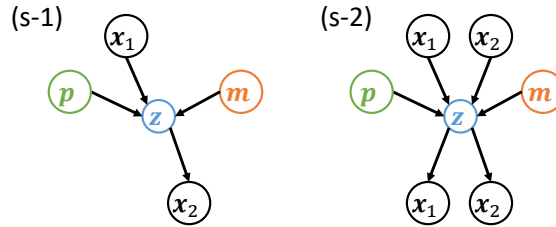
入出力の \mathbf{x} であるが, 必ず入力と出力が全く同じデータであるとは限らず, 様々な構成が考えられる. その中でも, Fig. 3.4 における 4 つのネットワーク構造は最も基礎的な構造で, これらにより身体図式の基本的な機能は表現可能であり, ここからいくつかの入出力増減により全てのネットワークが構成可能である. 静的身体図式において $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2\}$ である場合, \mathbf{x}_1 と \mathbf{x}_2 を反転させたような重複を除くと, その構造には $\{\mathbf{x}_1 \rightarrow \mathbf{x}_2, \mathbf{x}_1 \rightarrow (\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_1, \mathbf{x}_2) \rightarrow \mathbf{x}_2, (\mathbf{x}_1, \mathbf{x}_2) \rightarrow (\mathbf{x}_1, \mathbf{x}_2)\}$ の 4 つが考えられる. このうち, $\mathbf{x}_1 \rightarrow (\mathbf{x}_1, \mathbf{x}_2)$ と $(\mathbf{x}_1, \mathbf{x}_2) \rightarrow \mathbf{x}_2$ は, $\mathbf{x}_1 \rightarrow \mathbf{x}_2$ に付け加えて新しい機能が得られるわけではない. また, 学習効率を考えるとネットワークはあり得る最もシンプルな形が良い. ゆえに, $\mathbf{x}_1 \rightarrow \mathbf{x}_2$ の (s-1) と $(\mathbf{x}_1, \mathbf{x}_2) \rightarrow (\mathbf{x}_1, \mathbf{x}_2)$ の (s-2) の 2 つが静的身体図式における基礎的な 2 つのネットワーク構造となる. これは, (s-1) 感覚と運動の単方向の関係, (s-2) は双方向の関係を表現していると捉えることもできる. 例として, \mathbf{x}_1 から \mathbf{x}_2 は計算可能であるが, \mathbf{x}_2 から \mathbf{x}_1 は計算可能ではない場合は (s-1) が適切である (\mathbf{x}_1 を関節角度, \mathbf{x}_2 を作業空間における位置座標とすると分かりやすい). 一方で, \mathbf{x}_1 から \mathbf{x}_2 が, \mathbf{x}_2 から \mathbf{x}_1 が計算可能であるような場合は, (s-2) のネットワークが適切であると考えられる.

また, 動的な身体図式について, \mathbf{x}_1 を状態変数 \mathbf{s} , \mathbf{x}_2 を制御入力 \mathbf{u} とした場合に, その構造には $\{\mathbf{u} \rightarrow \mathbf{s}', \mathbf{u} \rightarrow (\mathbf{s}', \mathbf{u}'), \mathbf{s} \rightarrow \mathbf{u}', \mathbf{s} \rightarrow (\mathbf{s}', \mathbf{u}'), (\mathbf{s}, \mathbf{u}) \rightarrow \mathbf{s}', (\mathbf{s}, \mathbf{u}) \rightarrow \mathbf{u}', (\mathbf{s}, \mathbf{u}) \rightarrow (\mathbf{s}', \mathbf{u}')\}$ の 7 つが考えられる ($\{\mathbf{s}, \mathbf{u}\}'$ は次時刻の $\{\mathbf{s}, \mathbf{u}\}$) を表す. このうち, $\{\mathbf{u} \rightarrow (\mathbf{s}', \mathbf{u}'), \mathbf{s} \rightarrow \mathbf{u}', \mathbf{s} \rightarrow (\mathbf{s}', \mathbf{u}'), (\mathbf{s}, \mathbf{u}) \rightarrow \mathbf{u}', (\mathbf{s}, \mathbf{u}) \rightarrow (\mathbf{s}', \mathbf{u}')\}$ の 5 つは出力に \mathbf{u} を含むため, 現在状態から次時刻の制御入力を出力するような, 外部から影響を受けることなく時間発展する模倣学習型のネットワーク構造となる. また, $\mathbf{u} \rightarrow \mathbf{s}'$ と $(\mathbf{s}, \mathbf{u}) \rightarrow \mathbf{s}'$ は出力に \mathbf{s} のみを含むため, 状態を予測する運動方程式型のネットワーク構造と言える. 模倣学習型のネットワークについては, $(\mathbf{s}, \mathbf{u}) \rightarrow (\mathbf{s}', \mathbf{u}')$ は他 4 つのネットワーク構造の情報を全て含

み, 運動方程式型のネットワークについては, $(s, u) \rightarrow s'$ は他 1 つのネットワーク構造の情報を含む. ゆえに, $(x_{1,t}, x_{2,t}) \rightarrow (x_{1,t+1}, x_{2,t+1})$ の (d-1) と $(x_{1,t}, x_{2,t}) \rightarrow x_{1,t+1}$ の (d-2) の 2 つが動的な身体図式における基礎的な 2 つのネットワーク構造となる. 例として, x_1 を台車型ロボットの位置姿勢, x_2 を車輪速度入力とすると, $(x_{1,t}, x_{2,t}) \rightarrow x_{1,t+1}$ のような (d-2) のネットワークが適切である. 車輪速度入力は現在状態から推論できないため, 模倣学習型ではなくこの形になる.

このように, ネットワークには様々な形が考えられる. このネットワーク構造は人間が直接決定することができる一方で, 3.2 節に述べる方法により自動的に決定することもできる.

Static Body Schema



Dynamic Body Schema

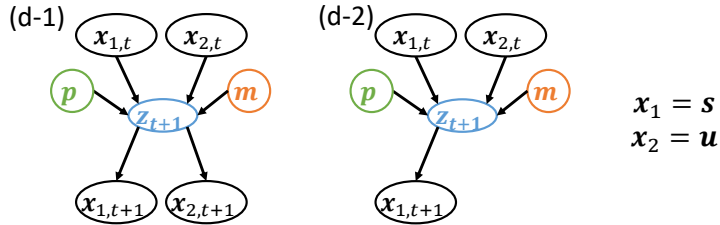


Fig. 3.4: Examples of the network structure of the generalized multisensory correlational model.

また, それぞれのネットワークの要素は必須のものではない. x の入出力が異なるように, m や p も無くすることができる. z を求める入力の組み合わせが一通りの場合, m は必要ない. また, 身体・道具・環境変化等を組み込まない場合は p は必要ない.

なお, 本研究では, ネットワークの入力に使われる値を x_{in} , 出力に使われる値を x_{out} として表現する. また, ネットワークの入出力に用いる全ての値を並べたものを x と表現する. つまり, 4 種類のモダリティ $x_{\{1,2,3,4\}}$ があるとき, それらを合わせたものを $x^T = \begin{pmatrix} x_1^T & x_2^T & x_3^T & x_4^T \end{pmatrix}$ によって表現する.

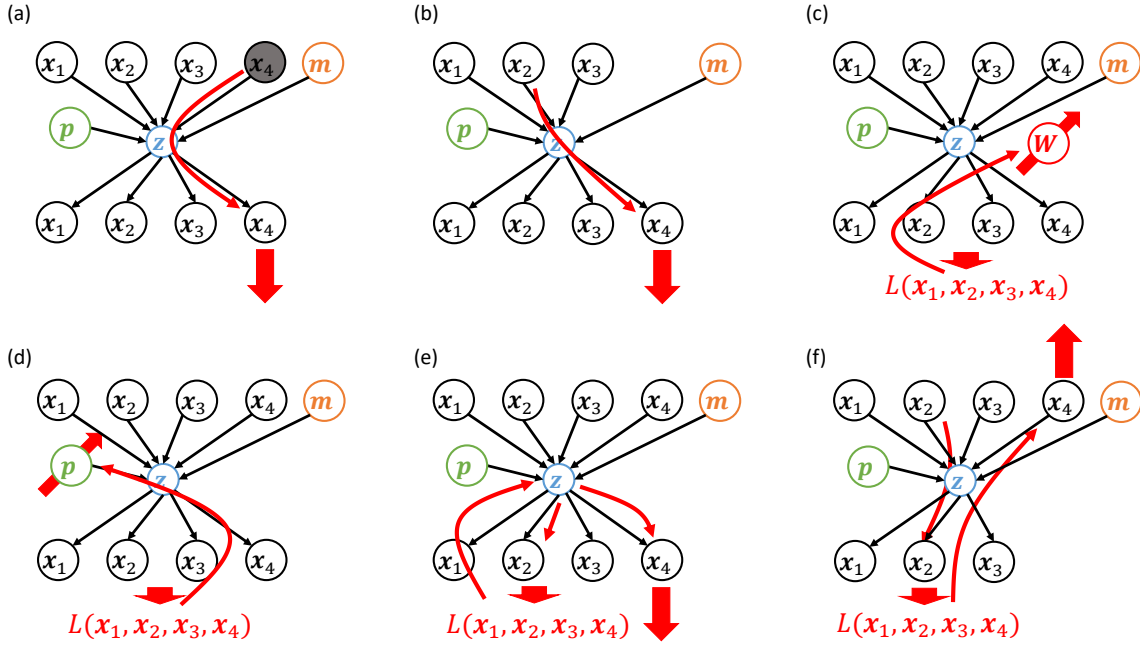


Fig. 3.5: The basic use of the generalized multisensory correlational model.

3.1.2 ネットワークの基礎的利用法

このネットワークの基礎的な6つの利用方法(a)–(f)をFig. 3.5に示す。これはネットワーク入出力に対する順伝播と誤差逆伝播の全ての組み合わせを表現している。この(a)–(f)は互いに組み合わせることができる。

(a)は、現在得ることができないデータを推定する方法である。例えば x_4 が得られなかった場合、 m を $\begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}^T$ に設定することで、 $x_{\{1,2,3\}}$ から x_4 を推論する。つまり、得ることができないデータをマスクして残りのデータからそのデータを推論する方法である。

(b)は(a)とほとんど同じであるが、出力のデータの方が入力に比べて多い場合に利用される。例えば x_4 を得たい場合、 m を $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T$ に設定することで、 $x_{\{1,2,3\}}$ から x_4 を推論する。つまり、入力側 x_{in} にはないデータを推論する、通常のニューラルネットワークと同じ形と言える。

(c)はネットワークの重み W の調節に相当する。出力に関して損失関数 L を定義し、 $\partial L / \partial W$ から一般的な学習と同様に重み W を更新する。

(d)はParametric Bias p の調節に相当する。出力に関して損失関数 L を定義し、 $\partial L / \partial p$ から p を更新する。重み W の更新はネットワーク全体の構造を変化させてしまうのに対して、Parametric Bias p の更新はネットワークの全体的な構造は保ったまま、一部の関係性・ダイナミクスを変更する。

(e) は順伝播と誤差逆伝播の繰り返しにより、ある損失関数を最適化する \mathbf{x}_{out} を計算することに相当する。まずは (a) や (b) 等により \mathbf{z} を計算する。ここで、例えばある条件を満たす \mathbf{x}_4 を計算したい場合、 \mathbf{z} から $\mathbf{x}_{\{1,2,3,4\}}$ を推論し、これらに対して損失関数を定義、 $\partial L / \partial \mathbf{z}$ から \mathbf{z} を更新する。この推論と更新を繰り返すことで損失関数を最小化するような \mathbf{x}_4 を計算することができる。つまり、Decoder 側における推論と更新の繰り返しによる損失関数最小化を意味する。

(f) は (e) と同様の繰り返し計算であるが、求めたい値が出力側でない場合に相当する。例えば求めるべき \mathbf{x}_4 が出力にない場合、 $\mathbf{x}_{\{1,2,3,4\}}$ から出力を推論し、これらに対して損失関数を定義、 $\partial L / \partial \mathbf{x}_4$ から \mathbf{x}_4 を更新する。この推論と更新を繰り返すことで損失関数を最小化するような \mathbf{x}_4 を計算することができる。つまり、Encoder と Decoder を含むネットワーク全体における推論と更新の繰り返しによる損失関数最小化を意味する。

3.1.3 データ収集

ネットワークの学習のためには、それに必要な \mathbf{x} のデータを収集する必要がある。この方法には主に、ランダム動作、決定的動作、教示の3つが存在する。まず、ランダム動作は制御入力をランダムに与え、このときの \mathbf{x} を取得する。なお、ランダムと言っても、制御入力の最大値や最小値等の制約を加えたうえで動作を行う。また、あるランダムな数値から制御入力への写像を考え、これを使って制約を加えながら動作させることも可能である。次に、決定的動作は、人間が予め定義した動作を行い、このときの \mathbf{x} を取得する。ある一定の動きを先に決めておき、その動きのパラメータを直接指定しながら動作を行っても良い。最後に、教示は VR デバイスやセンサグローブ、GUI アプリケーション等を使って、人間が直接動作指令を決めることを言う。ランダムでは難しいタスクに対しては、より効率的にデータを収集することができる。

収集する \mathbf{x} は、常に全てのデータが集まる必要はない。たまに視界が遮られたり、一部のデータの得られる間隔が長かったりしても良い。また、実際にロボット運用する際には得られないセンサ入力を、学習用のデータ収集時にのみ取得するのも良い。

集める \mathbf{x} はロボットのセンサから直接得られる情報だけとは限らない。例えば画像情報から得られた物体認識結果や特定の周波数に関する音情報など、存在するセンサの値を処理したうえで入力する場合も考えられる。

3.1.4 ネットワーク学習

まず、静的身体図式の学習について述べる。 \mathbf{x} のデータ D が得られたとき、通常であれば \mathbf{x}_{in} を入力として、出力を推論し、平均二乗誤差を損失関数としてこれが \mathbf{x}_{out} に近づくように学習を行う。一方、GelMiCo にはマスク変数 \mathbf{m} が伴うため、これを学習に含める必要がある。まず、実行可能な \mathbf{m} の組み合わせ \mathcal{M} を保持する。そして、学習時には \mathbf{x}_{in} に対して \mathcal{M} に含まれる全ての \mathbf{m} 、またはランダムな \mathbf{m} により、対応する \mathbf{x}_{in} の一部をマスクし、 \mathbf{x}_{in}^{masked} を作成する。この \mathbf{x}_{in}^{masked} と対応する \mathbf{m} を入力し、ネットワークの重み W の学習を行う。また、 \mathbf{x} は常に全てのモダルのデータが得られているとは限らない。例えば、 $\mathbf{x}_{\{1,2,4\}}$ は得られるものの、場合によって \mathbf{x}_3 が得られない状態等が考えられる。このとき、この \mathbf{x} に対するマスクについては、得られなかったデータをマスクするような \mathbf{m} を選び用いる。そのような \mathbf{m} が \mathcal{M} に含まれなかった場合にはそのデータを使って学習は行わない。また損失関数についても、得られたデータのみに対して平均二乗誤差を計算し、重み W を更新する。

Parametric Bias \mathbf{p} を用いる場合は、さらに一段学習方法を変化させる必要がある。この場合、身体や道具・環境の状態を変化させながらデータを取る必要がある。ある状態 k について得られたデータを $D_k = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T_k}\}$ ($1 \leq k \leq K$) とする (K を全体の状態数、 T_k を状態 k におけるデータ数とする)。よって、学習に用いるデータは $D = \{(D_1, \mathbf{p}_1), (D_2, \mathbf{p}_2), \dots, (D_K, \mathbf{p}_K)\}$ となる。ここで、 \mathbf{p}_k はその試行 k に関する Parametric Bias であり、そのデータ D_k については共通の値で、異なるデータについて別の値となる変数である。このデータ D を使って、ネットワークの重み W と Parametric Bias \mathbf{p}_k を同時に更新していく。

動的身体図式の学習は静的身体図式の学習を時系列に拡張した形で行われる。 \mathbf{x} のデータ D は $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t, \dots\}$ のように時系列で得られる (\mathbf{x}_t はタイムステップ t における \mathbf{x} を表す)。時系列ネットワークの展開数 N_{step} を決め、得られたデータを N_{step} ごとに分割し、これを用いて GelMiCo を学習させる。つまり、例えば $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_{step}}\}$ というデータに対して、 $\mathbf{x}_1, \mathbf{x}_2$ と順にネットワークへと入力し、出力された $\mathbf{x}_2, \mathbf{x}_3$ に対して順に損失を計算して、ネットワークの重み W を更新する。ここで、マスク \mathbf{m} はネットワークへの初期入力に対するマスク \mathbf{m}_{init} と、その後の入力に対するマスク \mathbf{m}_{seq} の2種類を考える必要がある。つまり、潜在変数 \mathbf{z}_t を推論できるものと、一度計算された \mathbf{z}_t を \mathbf{z}_{t+1} へ更新できるモダルの組み合わせは異なるということを意味する。これは、例えば \mathbf{x}_1 を台車型ロボットの位置姿勢、 \mathbf{x}_2 を車輪速度入力とするとわかりやすい。現在の状態には \mathbf{x}_1 が必ず必要であるが、その後の状態は \mathbf{x}_2 のみから更新することができることを意味する。よって、 \mathbf{m}_{init} と \mathbf{m}_{seq} に関してそれぞれ可能なマスクの組み合わせ $\mathcal{M}_{\{init, seq\}}$ を決め、一度目の推論時は \mathcal{M}_{init} の中からランダム、または全てのマスクを、それ以降の推論時には \mathcal{M}_{seq} の中からランダム、または全てのマスクを使

い GelMiCo の計算を行う。

動的な身体図式において Parametric Bias \mathbf{p} を用いる場合は、 D_k が時系列データになるのみで、その他の操作は静的な身体図式と同じである。

3.1.5 ネットワーク更新

ネットワークの更新には W の更新、 \mathbf{p} の更新、 W と \mathbf{p} の同時更新の3種類が考えられる。これは、Fig. 3.5 における (c) と (d)、またはその組み合わせに相当する。データ D が得られたとき、3.1.4 節と同様に損失関数を計算し、この誤差を使って勾配法により W のみ、 \mathbf{p} のみ、または W と \mathbf{p} を同時に更新する。オフライン更新の際には、ある一定のデータが溜まった後に一度にこの更新を行う。一方、オンライン更新の際には、ある閾値よりもデータが溜まってから逐次的にネットワークを更新していく。データの最大数を決め、それよりも多くデータが集まった場合は古いものから順にデータを捨てていく。学習の際、実際に得られた D に加えて、原点や幾何モデル、最小値や最大値等何らかの制約がある場合には、これを表現するデータも D に加えて更新を行う。 \mathbf{p} のみの更新では、ネットワーク全体の構造は変化せず一部のダイナミクスのみが変化するため、現在のデータに対する過学習が起きにくい。一方、 W の更新や W と \mathbf{p} の同時更新ではネットワーク全体の構造が変化するため、過学習が起きやすい。

3.1.6 順伝播と誤差逆伝播の繰り返しによる最適化計算

後の状態推定・制御・シミュレーション等で多用する最適化計算について述べる。これは Fig. 3.5 の (e), (f) に相当する操作であり、ある損失関数をもとに \mathbf{x} や \mathbf{z} を繰り返し最適化する手順を以下に示す。ここでは例として、 \mathbf{z} を \mathbf{x}_{out} に関する損失関数をもとに最適化することとし、 $\mathbf{x}_{out} = \mathbf{h}_{dec}(\mathbf{z})$ の関係があるとする。

- (1) 最適化される変数 \mathbf{z}^{opt} に初期値 \mathbf{z}^{init} を代入する。
- (2) \mathbf{x}_{out} の予測値 $\mathbf{x}_{out}^{pred} = \mathbf{h}_{dec}(\mathbf{z}^{opt})$ を推論する。
- (3) 損失関数 \mathbf{h}_{loss} を使い、損失 L を計算する。
- (4) 誤差逆伝播により、 $\partial L / \partial \mathbf{z}^{opt}$ を計算する。
- (5) 勾配法により、 \mathbf{z}^{opt} を更新する。

(6) (2)–(5) の工程を繰り返し, \mathbf{z}^{opt} を最適化する.

ここで, 5 の工程について詳しく説明する. 5 の工程は基本的に以下のような操作を行う.

$$\mathbf{z}^{opt} \leftarrow \mathbf{z}^{opt} - \gamma \frac{\partial L}{\partial \mathbf{z}^{opt}} \quad (3.1)$$

ここで, γ は学習率である. この γ は定数でも良いが, 変数として様々な γ を試し, より速い収束を目指すこともできる. 例えば, γ の最大値 γ_{max} を決め, $[0, \gamma_{max}]$ を N_{batch} 等分し (N_{batch} は学習のバッチサイズとなる定数), それぞれの γ で \mathbf{z}^{opt} を更新する. その後, (2) と (3) の工程で最も L の小さかった \mathbf{z}^{opt} を選び, これに対して (4) と (5) の工程を行うことを繰り返す.

3.1.7 状態推定

状態推定は, 現在得られないセンサ値をネットワークから推定することを意味する. これは Fig. 3.5 における (a), (b), (e), (f) が用いられる. \mathbf{x}_{out} に推定したい値が含まれる場合, (a) または (b) の実行を考え, これらが実行出来ない場合に (e) の実行を考える. \mathbf{x}_{out} に推定したい値が含まれない場合, (f) の実行を考える. まず静的身体図式の場合を述べ, 次に動的身体図式の場合を述べる.

(a) において, 現在得られていないデータについて 0, 得られているデータについて 1 としたマスク \mathbf{m} を考える. もしこのマスク \mathbf{m} が実行可能なマスクの集合 \mathcal{M} に含まれていれば, 得られていないデータを $\mathbf{0}$ とした \mathbf{x}_{in}^{masked} とこの \mathbf{m} をネットワークに入力することで, 得られていないデータを推定することが可能である.

(b) においても同様で, ネットワークに必要な入力揃っている場合には, 直接残りのデータを推定することができる. 入力揃っていない場合でも, 実行可能な \mathbf{m} が存在すれば, (a) と同様の形で得られていないデータを推定することができる.

(a) と (b) の形で実行可能な \mathbf{m} が推定できない場合は, (e) の形で状態推定を行う. これは, 3.1.6 節において, 損失関数を以下のように設定した場合に相当する.

$$h_{loss}(\mathbf{x}_{out}^{pred}, \mathbf{x}_{out}^{data}) = \|\mathbf{m}_{x_{out}} \otimes (\mathbf{x}_{out}^{pred} - \mathbf{x}_{out}^{data})\|_2 \quad (3.2)$$

ここで, \mathbf{x}_{out}^{data} は現在得られた \mathbf{x}_{out} のデータであり, 得られなかった一部のデータが欠損している. また, $\mathbf{m}_{x_{out}}$ は欠損しているデータについて 0, していないデータについて 1 を取るマスクである. \otimes はアダマール積を表している. つまり, 得られたデータについてのみ予測が合致するように \mathbf{z}^{opt} を更新していくことに相当する. 最終的に, 得られた \mathbf{z}^{opt} から \mathbf{x}_{out}^{opt} を推定する.

\mathbf{x}_{out} に推定したい値が含まれない場合は (f) の形で状態推定を行う。これは、3.1.6 節において、最適化される変数 \mathbf{z}^{opt} とその初期値 \mathbf{z}^{init} が、 \mathbf{x}_{in}^{opt} と \mathbf{x}_{in}^{init} に変更された場合に相当する。損失関数は前述の (e) の場合と同様である。つまり、潜在表現 \mathbf{z} ではなく、直接ネットワーク入力 \mathbf{x}_{in} に誤差を伝播し、得られた \mathbf{x}_{in}^{opt} を推定値として用いる。

動的身体図式の場合も本質は変わらない。(a) や (b) において、 $\mathbf{x}_{out,t}$ を得るためには、 $\mathbf{x}_{in,t-1}$ を同様の形で入力すれば良い。なお、基本的にマスクは \mathbf{m}_{seq} を用いるが、毎回 $\mathbf{z} = \mathbf{0}$ として \mathbf{m}_{init} を用いても良い。(e) と (f) の場合は、損失関数は以下ようになる。

$$h_{loss}(\mathbf{x}_{t,out}^{pred}, \mathbf{x}_{t,out}^{data}) = \|\mathbf{m}_{x_{out}} \otimes (\mathbf{x}_{t,out}^{pred} - \mathbf{x}_{t,out}^{data})\|_2 \quad (3.3)$$

3.1.8 制御

制御はネットワークの構造次第で、3.1.7 節の状態推定と同様に、Fig. 3.5 における (a), (b), (e), (f) のいずれかの計算が行われる。制御入力が \mathbf{x}_{in} または \mathbf{x}_{out} のどちらに含まれるか、指令状態を直接入力できる、または指令状態を損失関数の形で表現する必要があるかによってどの計算を用いるかが変わる。まず静的身体図式の場合を述べ、次に動的身体図式の場合を述べる。

制御入力が \mathbf{x}_{out} に含まれ、指令状態が全て \mathbf{x}_{in} から直接入力できる場合は、(a) または (b) が実行される。つまり、 $\mathbf{x}_{out} = \mathbf{h}(\mathbf{x}_{in}, \mathbf{m})$ となる。

次に、制御入力が \mathbf{x}_{out} に含まれ、指令状態を損失関数の形で表現しなければならない場合、(e) が実行される。これは、3.1.6 節において、損失関数を $h_{loss}(\mathbf{x}_{out}^{pred}, \mathbf{x}_{out}^{ref})$ の形で実行した場合に相当する。 \mathbf{x}_{out}^{ref} は \mathbf{x}_{out} の指令状態であり、 h_{loss} は例として、 $\|A\mathbf{x}_1^{pred} - \mathbf{x}_1^{ref}\|_2$ や $\|\mathbf{x}_1^{pred} - \mathbf{x}_1^{ref}\|_2 + \|\mathbf{x}_2^{pred}\|_2$ 等、様々な形が考えられる (A は何らかの変換行列を表す)。これをもとに \mathbf{z}^{opt} を計算し、最終的に \mathbf{x}_{out}^{opt} から制御入力を計算、実機に指令する。

最後に、制御入力が \mathbf{x}_{out} に含まれない場合には (f) が実行される。これは、3.1.6 節において、最適化される変数 \mathbf{z}^{opt} とその初期値 \mathbf{z}^{init} が、 \mathbf{x}_{in}^{opt} と \mathbf{x}_{in}^{init} に変更された場合に相当する。損失関数は前述の (e) の場合と同様である。つまり、潜在表現 \mathbf{z} ではなく、直接ネットワーク入力 \mathbf{x}_{in} に誤差を伝播する。得られた \mathbf{x}_{in}^{opt} から制御入力を計算、実機に指令する。

動的身体図式の場合も、(a), (b) は基本的には変わらない。しかし、時間発展するモデルの場合、指令状態はモデルの出力に対してしか考えることができないため、現在状態から次の行動を決定するような、模倣学習や強化学習と同じ構造になる。なお、時間発展する系において直接 \mathbf{z} を更新することはできないため、(e) は使うことができない。一方、(f) においては Eq. 3.1 の更新則が以下のように変化する

る。これは、動的な身体図式において入力 $\mathbf{x}_{in,t}$ 、出力は $\mathbf{x}_{out,t+1}$ であり、 t における制御入力以外のセンサ情報は実機から得られたものであり、変更できないためである。

$$\mathbf{x}_{cmd,t}^{opt} \leftarrow \mathbf{x}_{cmd,t}^{opt} - \gamma \frac{\partial L}{\partial \mathbf{x}_{cmd,t}^{opt}} \quad (3.4)$$

ここで、 $\mathbf{x}_{cmd,t}^{opt}$ は $\mathbf{x}_{in,t}^{opt}$ に含まれる制御入力部である。また、この時系列ニューラルネットワークを展開し、以下のようにより長い区間において制御入力を最適化することができる。

$$\mathbf{x}_{out,t+1:t+N_{step}}^{pred} = \mathbf{h}^{expand}(\mathbf{x}_{in,t:t+N_{step}-1}^{opt}) \quad (3.5)$$

$$L = \mathbf{h}_{loss}(\mathbf{x}_{out,t+1:t+N_{step}}^{pred}) \quad (3.6)$$

$$\mathbf{x}_{cmd,t:t+N_{step}-1}^{opt} \leftarrow \mathbf{x}_{cmd,t:t+N_{step}-1}^{opt} - \gamma \frac{\partial L}{\partial \mathbf{x}_{cmd,t:t+N_{step}-1}^{opt}} \quad (3.7)$$

ここで、 \mathbf{h}^{expand} は動的な身体図式における時系列ニューラルネットワークを N_{step} 回展開した関数を表す。(a), (b), (f) については、指令状態に対して損失関数を定義したいかどうかによって使い分ける必要がある。

3.1.9 シミュレーション

シミュレーションとは、制御入力といくつかの拘束条件から現在のロボット状態を推定することを意味する。

動的な身体図式の場合はシミュレーションは非常に簡単である。入力が $\mathbf{x}_{in,t}$ 、出力が $\mathbf{x}_{out,t+1}$ のように時間発展するため、 $\mathbf{x}_{out,t+1}$ で得られた値と $t+1$ における制御入力を合わせて $\mathbf{x}_{in,t+1}$ を作成し、これをネットワークに入力することを繰り返していくことで \mathbf{x} の時間発展をシミュレートすることができる。

一方、静的な身体図式の場合は 3.1.7 節における状態推定とほぼ同じ形で、(a), (b), (e), (f) のいずれかの形でシミュレーションを行うことができる。このとき異なるのは (e), (f) における損失関数のみである。損失関数は状態推定のように現在のデータ \mathbf{x}_{out}^{data} が存在しないため、 $\mathbf{h}_{loss}(\mathbf{x}_{out}^{pred})$ の形で記述される。この損失関数はロボットの動作に関する関節トルクや筋張力、動作速度等に関する様々な拘束を記述することになる。制御入力と損失関数の形で与えられた拘束をもとに、現在状態を遷移させていく。

3.1.10 異常検知

異常検知は \mathbf{x}_{out} に関する現在値 \mathbf{x}_{out} と推定値 \mathbf{x}_{out}^{est} の誤差の大きさに対して行う。一つの最も単純な異常検知方法は、 $\|\mathbf{x}_{out} - \mathbf{x}_{out}^{est}\|_2$ に対して閾値を設定し、これよりも誤差が大きくなった場合に異

常と見なす方法である。一方、その誤差の平均と分散を使うことで、より正確に異常検知を行うことができる。まず、異常のない、正常な状態において 3.1.7 節の状態推定のデータ \mathbf{x}_{out}^{est} と現在状態データ \mathbf{x}_{out}^{data} を収集する。なお、これは現在状態が全て得られたときについてのみ収集されるべきである。このデータに対して、誤差 $\mathbf{x}_{out}^{data} - \mathbf{x}_{out}^{est}$ の平均 μ と分散 Σ を計算しておく。実際に異常検知をする際には、 \mathbf{x}_{out} に関する現在値 \mathbf{x}_{out} と推定値 \mathbf{x}_{out}^{est} を常に取得し、これらに対して以下のマハラノビス距離 d を計算する。

$$d = \sqrt{(\mathbf{x}_{out} - \mu)^T \Sigma (\mathbf{x}_{out}^{est} - \mu)} \quad (3.8)$$

この d が設定した閾値を超えた時、異常が検知されたこととする。この閾値であるが、 μ と Σ を計算する際に得られたデータに対して d の分散を計算しておき、その 3σ 区間等を用いることが可能である。

3.1.11 一般化多感覚相関モデルの特徴

以下に 2.3.4 節で述べた一般化多感覚相関モデルの特徴について再掲する。

- ニューラルネットワークによりモデル化困難な身体図式を表現
- マスク表現を使い様々な感覚・制御入力の間関係性を表現
- 自己教師あり学習による自律的な実機学習
- オンライン学習・Parametric Bias 等を使った逐次的な身体図式変化への適応
- 順伝播と逆伝播の繰り返し計算による多様な指令状態実現・身体状態推定・異常検知
- ネットワーク入出力とマスク変数の自動決定による様々な身体・道具・環境・タスクへの適用

本モデルは、ニューラルネットワークとマスク表現により、様々な感覚・制御入力の間関係を記述することに成功している。実機から得られたデータのみを使い、アノテーション等なしの自己教師あり学習を行うことができる。ネットワークの重み W を直接学習させるオンライン学習、または Parametric Bias p のみを学習させることによる逐次的な環境適応が可能である。制御・状態推定・シミュレーション・異常検知の機能がネットワークの順伝播と誤差逆伝播の繰り返し計算により実現される。この制御目標値や、シミュレーションの拘束条件などは損失関数を変更することで任意に調整することが可能である。そして、後述する統一的なネットワーク表現、ネットワークの入出力構造に基づいた制御や状態推定方法の変化が可能であり、様々な身体図式に対応可能である。

3.1.12 プログラム構成

最後に、本身体図式学習のプログラム構成について述べる (Fig. 3.6). 本研究では深層学習フレームワークとして Chainer [137] を使用しており、身体図式のモデルファイルは npz ファイルとして保存される。学習や制御、状態推定など、全てのファイルは Python によって記述されている。基本的なモデルの情報を管理する設定ファイルは YAML として管理されている。この設定には、用いる \mathbf{x} のセンサ値や制御入力値をやり取りするための、ROS (Robot Operating System) のトピック名が記述されている。また、Parametric Bias (PB) や身体図式モデルの隠れ層の大きさ、モデルファイル名、その他データ取得の周期等が記述されている。モデルファイルには、この設定ファイルに書かれたトピックのうち、入力、出力として何を用いるかが記述されている。また、実行可能なマスク集合 \mathcal{M} 、ネットワークの重み W 、学習時に取得した Parametric Bias \mathbf{p}_k が格納される。

設定ファイルをもとに、Data Collector がデータ (CSV ファイル) を収集する。得られたデータをもとに入出力自動決定プログラムがモデルファイルの入出力トピックと実行可能なマスク集合を決定し保存する。この際、静的または動的な身体図式のどちらを使うか、また、それぞれの損失に関する閾値を決定する必要がある。次に、学習器がこの入出力トピック、実行可能なマスク集合をもとに、収集されたデータを用いて W と \mathbf{p}_k を学習しモデルファイルに保存する。この際、学習のバッチサイズやエポックサイズ等を指定する必要がある。運用時については、オンライン学習、制御、状態推定、異常検知、シミュレーションに関するプログラムが設定ファイルとモデルファイルをもとに動作する。この際、制御については指令値と損失関数の形を、異常検知についてはその閾値を与える必要がある。

人間が設定するパラメータをまとめると、データ収集時は{扱う ROS のトピック名、データ取得の周期}、ネットワーク構造の自動決定時・学習時は{静的か動的か、PB のユニット数、学習率、バッチ数、エポック数、層数、各層のユニット数、入出力決定に関する閾値}、運用時は{損失関数の形、学習率、バッチ数、エポック数、指令値 (制御の場合)、閾値 (異常検知の場合)}である。これ以外のパラメータについては、自動で決定、または学習が行われる。特にネットワーク構造の自動決定と学習時のパラメータはデフォルトの値が設定されており、そこから大きく変更する必要はない。一方で、どのデータを扱うか、得られた身体図式の運用時における損失関数の設定等は、大きくユーザに委ねられている。

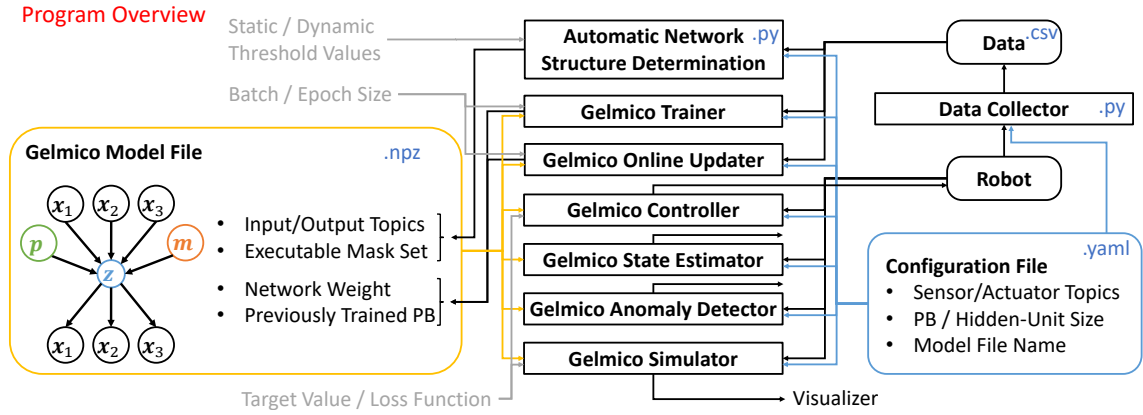


Fig. 3.6: The program overview of body schema learning system.

3.2 身体図式ネットワークの入出力自動決定

3.2.1 概要

本節では身体図式ネットワーク構造を決定する方法について述べる. 具体的には, x_{in} , x_{out} , そして実行可能なマスク m の集合 M である. このネットワーク構造を, 与えられたデータから自動で決定することができれば, 人間の作業を削減することができるだけでなく, ロボットの自律性を飛躍的に向上させることができる. つまり, 得られたデータから自律的にネットワーク構造を決定・学習し, これをもとに状態推定器や制御器等を自動構築することが可能になる. 本操作は主に, 不必要な入出力値の削減, 潜在空間から推論可能なネットワーク出力の決定, 潜在空間を推論可能なネットワーク入力とマスクの組み合わせの決定からなる (Fig. 3.7). なお, ネットワークの層数やユニット数は人間が外から与えられる仕組みとなっており, これらについては自動決定しない.

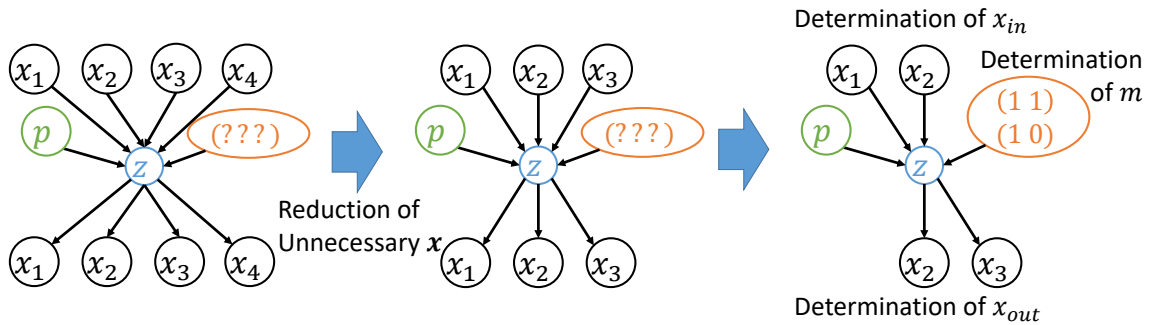


Fig. 3.7: The concept of the automatic determination of network input and output for body schema learning.

3.2.2 身体図式ネットワークの入出力自動決定

本節では順に, 相互情報量によるセンサ・アクチュエータ値 \mathbf{x} の関係性抽出による不必要な \mathbf{x} の削減, マスク \mathbf{m} のランダム化による身体図式ネットワークの学習, 学習されたネットワークにおける \mathbf{z} から \mathbf{x} の推論誤差に基づく \mathbf{x}_{out} の決定, \mathbf{x} から \mathbf{z} の推論誤差に基づく \mathbf{x}_{in} の決定について述べる. なお, 静的身体図式を使うか動的身体図式を使うかは本研究では自動決定せず, 予め決めておく. これを自動で決定したい場合は, データの取得方法を揃える必要がある. また, 静的身体図式と動的身体図式における損失を計算し, これを比較して設定した閾値から決定することとなる.

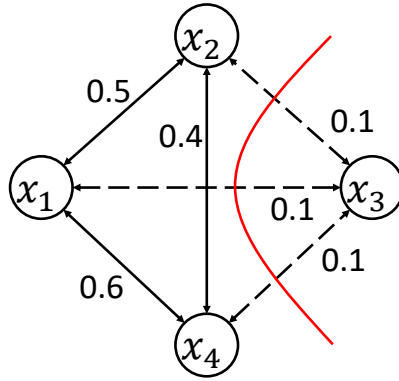


Fig. 3.8: The reduction of unnecessary sensors and actuators based on mutual information.

不要なセンサ・アクチュエータ情報の削除

まず, 与えられたセンサ・アクチュエータ \mathbf{x} の中から不必要な情報を削除する. ここでは相互情報量を用いる. 相互情報量には様々な計算手法があるが, 本研究では k -nearest neighbor を用いた相互情報量 [148] を用いる.

Fig. 3.8 に概要を示す. まず, それぞれのセンサ・アクチュエータの値 \mathbf{x}_i ($1 \leq i \leq N_{sensor}$) に対して, 相互情報量 $MI(\mathbf{x}_i, \mathbf{x}_j)$ ($1 \leq j \leq N_{sensor}$) を計算する. なお, $MI(\mathbf{x}_i, \mathbf{x}_j) = MI(\mathbf{x}_j, \mathbf{x}_i)$ であり, MI の最大値 MI_{max} は $MI(\mathbf{x}_i, \mathbf{x}_i)$ である. このとき, \mathbf{x}_i に対して $MI(\mathbf{x}_i, \mathbf{x}_j)/MI_{max}$ ($i \neq j, 1 \leq j \leq N_{sensor}$) が全て C_{thre}^{reduce} よりも小さかった場合, この \mathbf{x}_i はどの他のセンサとも関係が浅いため, 身体図式の獲得に必要なないと判断し, この \mathbf{x}_i を除いた \mathbf{x} を改めて \mathbf{x} として定義する. 動的身体図式学習の場合は, 時間的な遅延などにより $\mathbf{x}_{i,t}$ と $\mathbf{x}_{j,t}$ が相互に関係していても, 相互情報量が大きくなるとは限らない. そこで, $\mathbf{x}_{i,[t,t+T]}$ と $\mathbf{x}_{j,[t,t+T]}$ について相互情報量 MI を計算する (T は考慮するタイムステップ数を表す). これにより, 多少の時間遅れに対してもロバストに相互情報量を計算することが可能である. なお, 本

研究では直接扱わないが, 不要なセンサ間に関係性があり, それらセンサを削除できない場合も存在する. そのようなケースに対しては, 最小化カットや最小全域木等のアルゴリズムを使ってグループ分けしてから削除することが望ましい.

本節では相互情報量の計算については $k = 5$ とした.

ネットワーク学習

\mathbf{x} から \mathbf{x}_{in} , \mathbf{x}_{out} , 実行可能な \mathbf{m} の集合 \mathcal{M} を決定するにあたり, 得られたデータ D をもとに一度一般化多感覚相関モデル \mathbf{h} を学習させる. この学習された \mathbf{h} を使った際の推論誤差を元に入出力が決定される. ここで, それぞれのマスク \mathbf{m} に対して推論誤差を計算するため, 学習の際には実行できる可能性のある全てのマスクの組み合わせ \mathcal{M}_{all} (全て 0 であるマスクを除いた $2^{N_{sensor}} - 1$ の全組み合わせ) から \mathbf{m} をランダムに取り出して用いる. 動的な身体図式の場合も同様であり, \mathbf{m}_{init} と \mathbf{m}_{seq} は \mathcal{M}_{all} からランダムに選択して学習させる.

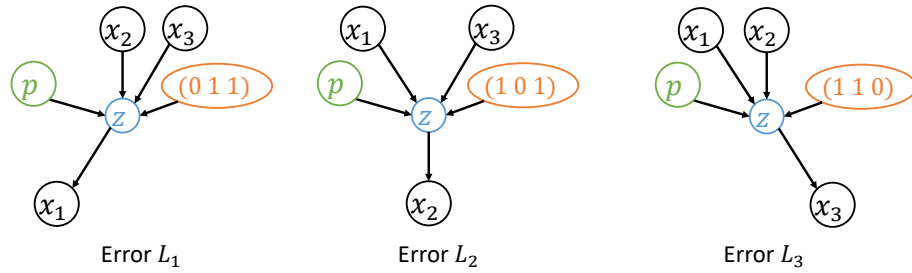


Fig. 3.9: The loss definition to determine the output of the network \mathbf{x}_{out} for static body schema learning.

ネットワーク出力決定

一般化多感覚相関モデルのネットワーク出力 \mathbf{x}_{out} を決定する. これは, ある値 \mathbf{x}_i ($1 \leq i \leq N_{sensor}$) が, その他の値 \mathbf{x}_j ($i \neq j, 1 \leq j \leq N_{sensor}$) から推論可能であるかどうかから判断することができる. もし推論可能であればこの \mathbf{x}_i は他のセンサ・アクチュエータと関係しあっており, ネットワークの出力としても推論すべき対象である. 一方, 推論可能でなければネットワークの学習に悪い影響を及ぼすため推論すべきではない.

静的な身体図式学習の場合は, Fig. 3.9 に示すようにある値 \mathbf{x}_i をそれ以外の値 \mathbf{x}_j から推論し, その誤差を L_i とする. $L_i < C_{thre}^{out}$ である \mathbf{x}_i のみ集め, \mathbf{x}_{out} を構築する. 動的な身体図式学習の場合は, Fig. 3.10 に示すように, まず, 全てが 1 のマスク $\mathbf{m}^T = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$ により \mathbf{z}_t を計算する. 次に, ある値 \mathbf{x}_i

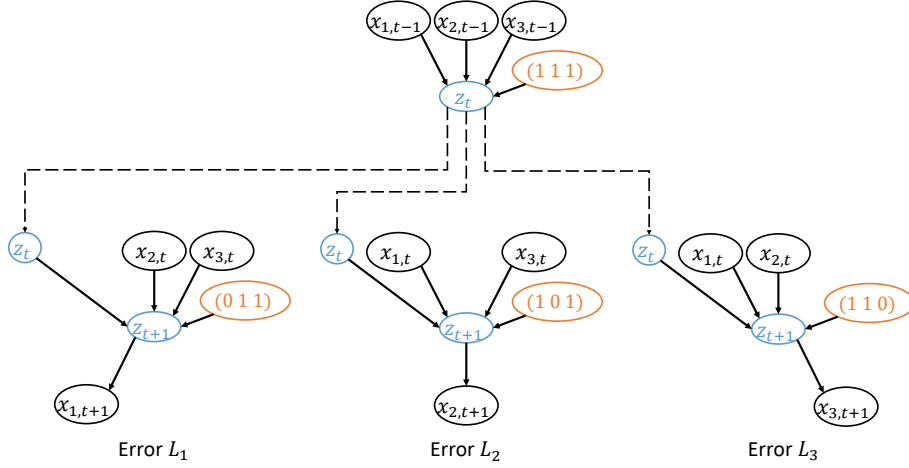


Fig. 3.10: The loss definition to determine the output of the network \mathbf{x}_{out} for dynamic body schema learning.

をそれ以外の値 \mathbf{x}_j と計算された \mathbf{z}_t から推論し、その誤差を L_i とする。同様に、 $L_i < C_{thre}^{out}$ である \mathbf{x}_i のみを集め、 \mathbf{x}_{out} を構築する。

ネットワーク入力決定

一般化多感覚相関モデルのネットワーク入力 \mathbf{x}_{in} を決定する。また、これと同時にネットワーク入力に対するマスク \mathbf{m} も決定する。これは、前節で決定した \mathbf{x}_{out} の値を、それぞれのマスクがどの程度推論可能かどうかによって判断することができる。推論可能であったマスク、つまり推論を可能にする \mathbf{x}_i の組み合わせを抜き出し、それらの集合が \mathbf{x}_{in} となる。もしある \mathbf{x}_i を含む全てのマスク \mathbf{m} を使った際の推論誤差が大きい場合は、その \mathbf{x}_i は \mathbf{x}_{in} から削除すべきである。

静的身体図式の場合は Fig. 3.11 に示すように、実行可能な全てのマスク \mathbf{m} に関して \mathbf{x}_{out} の推論誤差 L_m を計算する。ここで、 \mathbf{x}_{out} に含まれるセンサの集合を X_{out} とすると、 $X_{out} \subseteq X$ となるセンサの集合 X と対応するマスク \mathbf{m} によって \mathbf{x}_{out} が推論できるのは自明である。ゆえに、 $X_{out} \subsetneq X$ となるようなセンサの集合 X についてのみ、 L_m は計算する。Fig. 3.11 の例では、 $X_{out} = \{1, 2, 3\}$ であるため、 $X = \{1, 2, 3\}$ は除外されている。 $L_m < C_{thre}^{in}$ である \mathbf{m} とそれに対応する \mathbf{x}_i を集め、これらの和集合を \mathcal{M} と \mathbf{x}_{in} とする。

動的身体図式の場合はマスクが \mathbf{m}_{init} と \mathbf{m}_{seq} の2種類あるため、計算過程が多少異なる。まず、 \mathbf{x}_{in} と \mathbf{m}_{init} の集合を求める。Fig. 3.12 に示すように、 $\mathbf{m}_{init} = \mathbf{m}_{seq}$ として、実行可能な全てのマスク \mathbf{m}_{init} に関して \mathbf{x}_{out} の推論誤差 L_m^{init} を計算する。 \mathbf{m}_{init} によって \mathbf{z}_t を計算し、その後 $\mathbf{m}_{seq} = \mathbf{m}_{init}$ によ

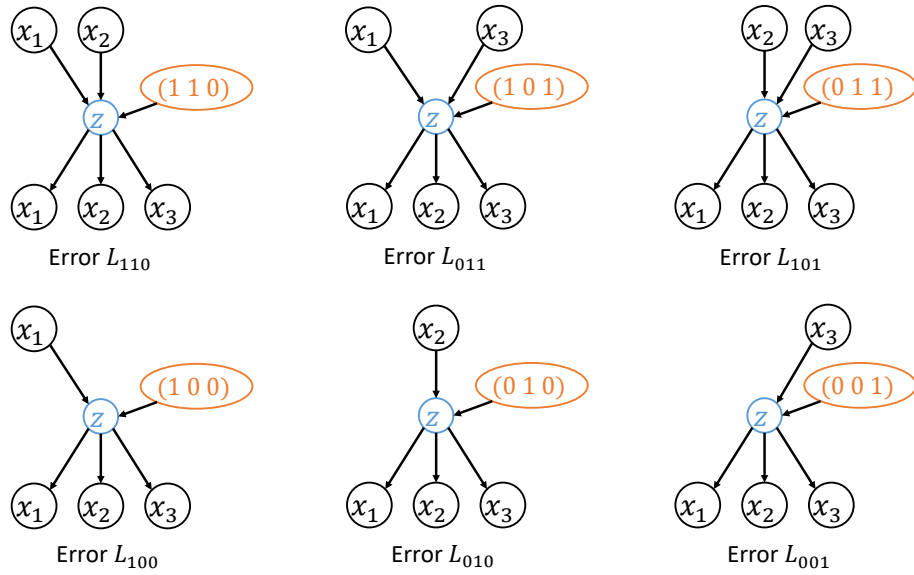


Fig. 3.11: The loss definition to determine the input of the network \mathbf{x}_{in} and possible masks \mathbf{m} for static body schema learning.

て $\mathbf{x}_{out,t+1}$ を計算する. ここでは, 静的身体図式の場合のように実行する \mathbf{m}_{init} を制限しない. これは, 静的身体図式と違い, 動的身体図式の場合は $\mathbf{x}_{in,t}$ から次時刻における $\mathbf{x}_{out,t+1}$ を計算するため, 推論する時刻が異なり, $\mathcal{X}_{out} \subseteq \mathcal{X}$ となるセンサ集合 \mathcal{X} を使ったとしても $\mathbf{x}_{out,t+1}$ が推論可能かどうかはわからないからである. $L_m^{init} < C_{thre}^{init}$ である \mathbf{m}_{init} とそれに対応する \mathbf{x}_i を集め, これらの和集合を \mathcal{M}_{init} と \mathbf{x}_{in} とする. 次に, \mathbf{m}_{seq} の集合を求める. ここでは, 先の \mathbf{m}_{init} の集合の計算過程と異なる点は一点だけである. Fig. 3.13 に示すように, $\mathbf{m}_{init}^T = (1 \ 1 \ \dots \ 1)$ として \mathbf{z}_t を計算し, 実行可能な全ての \mathbf{m}_{seq} を使って $\mathbf{x}_{out,t+1}$ を計算, 推論誤差 L_m^{seq} を計算する点である. $L_m^{seq} < C_{thre}^{seq}$ である \mathbf{m}_{seq} を集め, これらの集合を \mathcal{M}_{seq} とする.

これらネットワークの自動入出力決定は, C_{thre}^{reduce} , C_{thre}^{out} , C_{thre}^{in} , C_{thre}^{init} , C_{thre}^{seq} のいくつかの閾値によって変化する. これらの選び方次第でネットワーク構造は変化し, それに応じて制御や状態推定等において, 可能な操作も変化する. これらの閾値を低く設定すると推論の精度は上がる一方, 入出力のセンサは減り, 可能な操作は減少するというトレードオフがある. 複数のセンサ・アクチュエータと収集されたデータに対して, 様々なネットワーク構造をとり得ることが, その変化から考察できる.

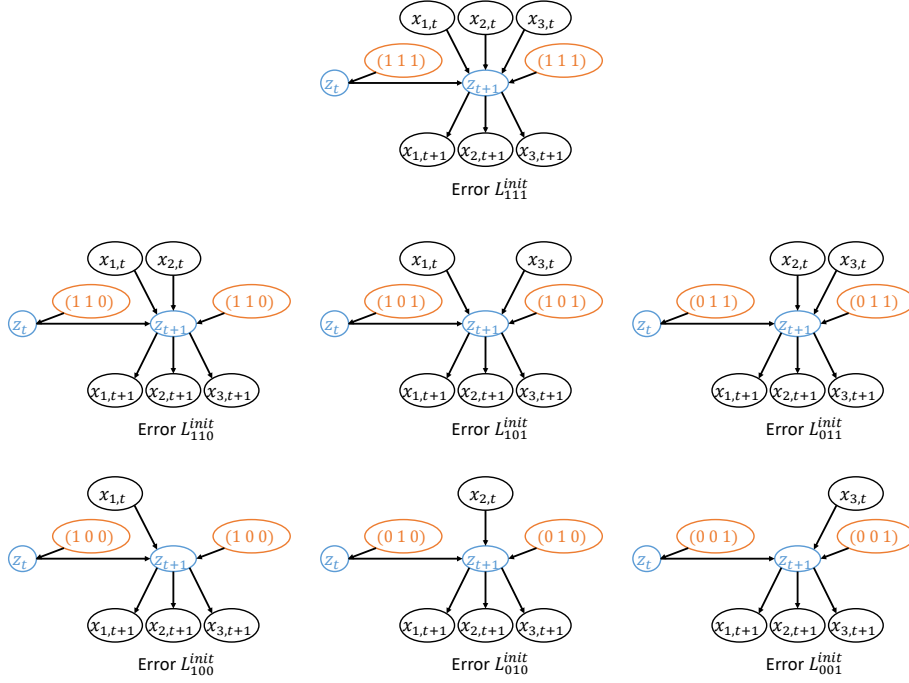
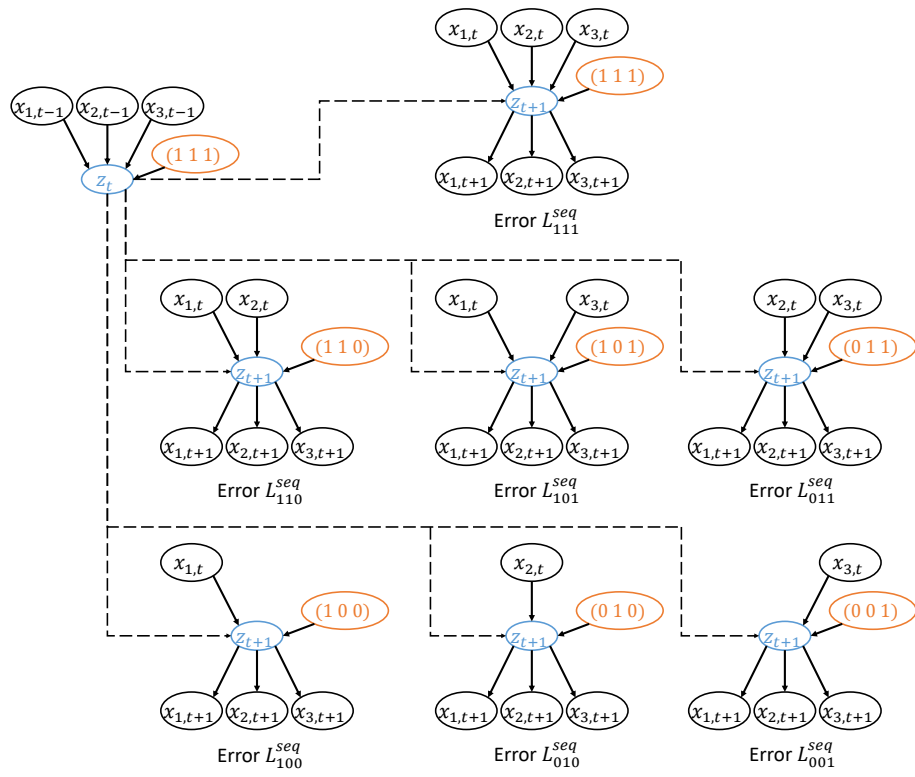


Fig. 3.12: The loss definition to determine the input of the network x_{in} and possible masks m_{init} for dynamic body schema learning.

3.3 本章のまとめ

本章では身体図式学習のモデルとして一般化多感覚相関モデルを提案した。多様なモーダル間の関係性を潜在変数を通して学習させることで、それらの関係と順伝播・逆伝播の繰り返し計算により、多様なネットワーク構造において状態推定・制御・シミュレーション・異常検知等が可能になる。損失関数の定義次第で入出力モーダルに様々な拘束をかけることができ、汎用性に優れている。Parametric Bias の導入により、小さな次元に身体図式の変容に関する情報を埋め込むことができ、逐次的なモデル変化適応が可能となっている。また、得られたデータから一般化多感覚相関モデルのネットワーク入出力と実行可能なマスク変数の集合を自動で決定することが可能である。

Fig. 3.13: The loss definition to determine possible masks m_{seq} for dynamic body schema learning.

第4章

身体図式学習に向けたハードウェア構成

本章の概要を Fig. 4.1 に示す。本章ではまず, 4.1 節-4.3 節において, 軸駆動型ロボット PR2, 台車型ロボット Fetch, 低剛性樹脂製ロボット KXR の, 本研究で用いる既存のハードウェアの構成について, その柔軟性と冗長性に焦点を当てながら述べる。次に, 本研究で新規開発した筋骨格ヒューマノイドの設計詳細について述べる。筋骨格ヒューマノイドは未だその構成法が確立されておらず, 筆者が中心となってその柔軟性と冗長性の利点最大化の観点から全身を開発した。4.4 節では, 柔軟性と冗長性の利点を活かすことが可能な筋骨格ヒューマノイドの設計要件として, モジュール化, 非線形弾性, 冗長筋配置, 冗長センサ

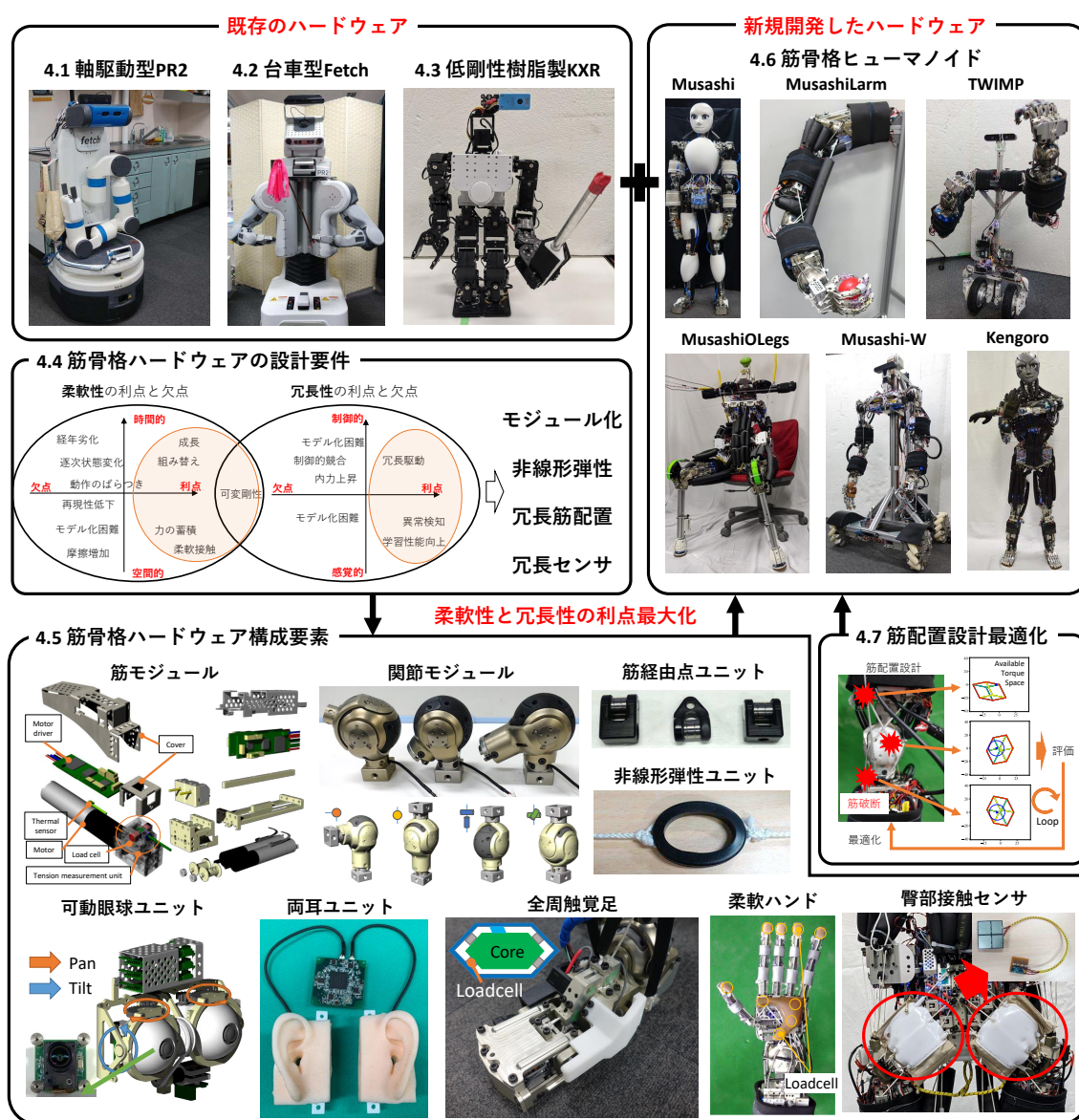


Fig. 4.1: The overview of this chapter.

性, 冗長筋配置, 冗長センサの4つの特徴に焦点を当てる. 4.5節ではこれらの要件に基づき, ハードウェアの構成要素として, 筋モジュール, 関節モジュール, 筋経由点ユニット, 非線形弾性ユニット, 可動眼球ユニット, 人体模倣型両耳ユニット, 全周触覚足, 柔軟ハンド, 臀部接触センサの開発について述べる. 4.6節では, これらの開発したモジュールにより構築された筋骨格ヒューマノイド群として MusashiLarm, Musashi, MusashiOLegs, TWIMP, Musashi-W, Kengoro の詳細を述べる. 最後に, 4.7節では, 筋骨格ヒューマノイドにおける筋破断を補償する冗長性最大化に基づく筋配置設計最適化について述べる.

4.1 軸駆動型ロボット PR2 のハードウェア構成

PR2 [149] は Willow Garage 社によって開発された台車型双腕ロボットである (Fig. 4.2). その最大の特徴は腕におけるカウンターバランス機構であり, 重力を打ち消すことで外力から容易に腕を動かすことが可能である. そのため Kinesthetic Teaching 等による教示が可能であり, 模倣学習等の実験にも有用である.

ハードウェア構成について簡単に述べる. PR2 の重量は約 220 kg であり, 高さは 1330 mm である. 腰部の直動機構によって最大 1645mm まで高くなる. 両腕にはグリップを含めた 8 自由度ずつの自由度があり, 1.8kg までの物体を運ぶことが可能である. なお, グリップは平行リンクによる 1 自由度開閉機構である. 頭部は 2 自由度を有している. 台車はステア用モータと車輪がそれぞれ四隅に配置された 4 輪ステア構造を持つ.

PR2 に備わる冗長なセンサ群について述べる. まず, レーザ測距センサ (LRF) が台車部と頭部に一つずつ配置されており, 2 次元上の平面について障害物までの距離を測定することができる. 台車側の LRF は環境からロボットの位置を推定するために用いられ, 頭部の LRF はさらにチルト自由度を用いることで 3 次元空間の点群を得ることが可能である. 次に, カメラは頭部上方に 1 台, 頭部正面に 5 台, 両腕にそれぞれ 1 台ずつ配置されている. 特に頭部正面のカメラは 2 台が広角ステレオカメラ, 2 台が望遠ステレオカメラ, 1 台が高解像度カメラである. 前腕部には手元を写すことが可能なカラーカメラが配置されている. 次に, 3 次元点群センサとして, 頭部上方に Microsoft Kinect (Microsoft Corp.) が配置されている. 赤外パターン光を照射しその歪みによって障害物までの距離を計測する. カラーカメラの画素とのマッチングによって色つきの 3 次元点群情報を生成することが可能である. 最後に, 頭部全方位マイクとして, Respeaker Mic Array v2.0 (Seeed Technology Co., Ltd.) が配置されている. 円状に並べた 4 チャンネルのマイクロフォンアレイから音声方向検知が可能である.

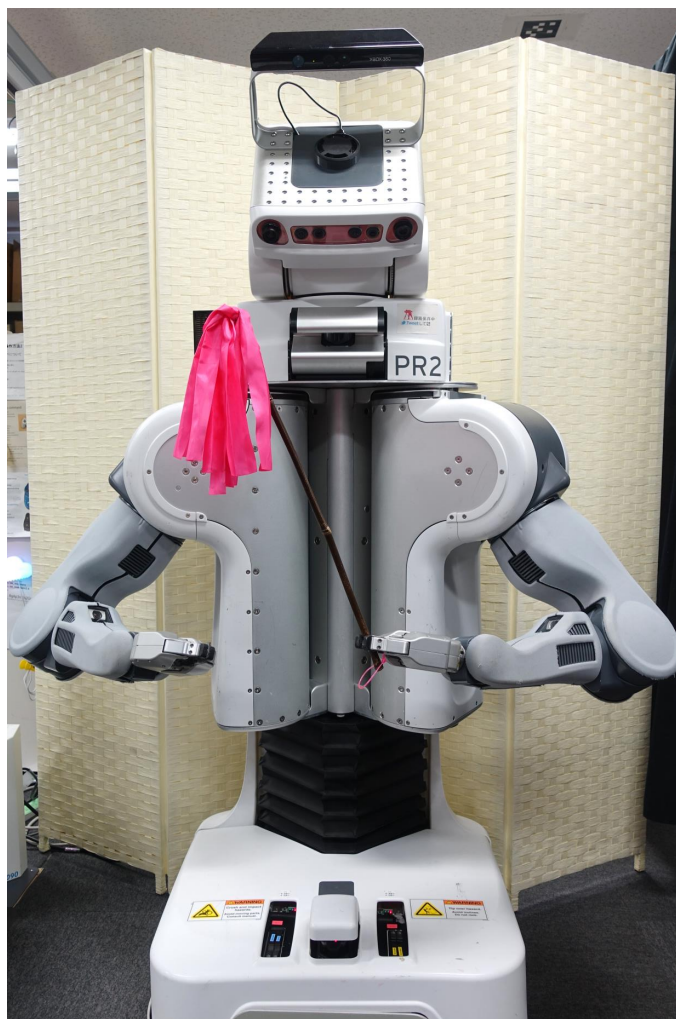


Fig. 4.2: PR2: The mobile robot with axis-driven dual arms.

4.2 台車型ロボット Fetch のハードウェア構成

Fetch [150, 151] は Fetch Robotics 社によって開発された台車型単腕ロボットである (Fig. 4.3).

ハードウェア構成について簡単に述べる. Fetch の重量は約 113 kg であり, 高さは 1096 mm である. 腰部の直動機構によって最大 1491 mm まで高くなる. 腕にはグリッパを含めた 8 自由度があり, 手を伸ばした状態で最大 6 kg のペイロードがある. なお, グリッパは 1 自由度の直動機構による平行グリッパである. 頭部は 2 自由度を有している. 台車は 2 つの車輪が平行に並び, 前後にキャスターがついた二輪構造を持つ.

Fetch に備わる冗長なセンサ群について述べる. まず, レーザ測距センサ (LRF) が台車部に一つ配置



Fig. 4.3: Fetch: The mobile robot with an axis-driven single arm.

されており、自己位置推定用に用いられている。次に、カメラ兼3次元点群センサが頭部に1台配置されており、3次元点群とカラー画像を15Hzで取得することができる。改造として、Intel RealSense T265 (Intel Corp.) や Intel RealSense L515 (Intel Corp.) が頭部や台車に取り付けられている場合もある。最後に、PR2と同様に Respeaker Mic Array v2.0 (Seeed Technology, Co., Ltd.) が頭部に配置されている。

4.3 低剛性樹脂製ロボット KXR のハードウェア構成

KXR [97] は近藤科学が販売する組み立てキットであり、その軸数やセンサを追加して用いているヒューマノイドである。

ハードウェア構成について簡単に述べる。KXR は様々な構成が可能であるが、主に本研究で直接用いるヒューマノイド型の KXR の構成に関して述べる。KXR の重量は約 1.3 kg であり、立ったときの高さは約 370 mm である。両腕にはそれぞれグリップを含めた 5 自由度があり、グリップは 1 自由度

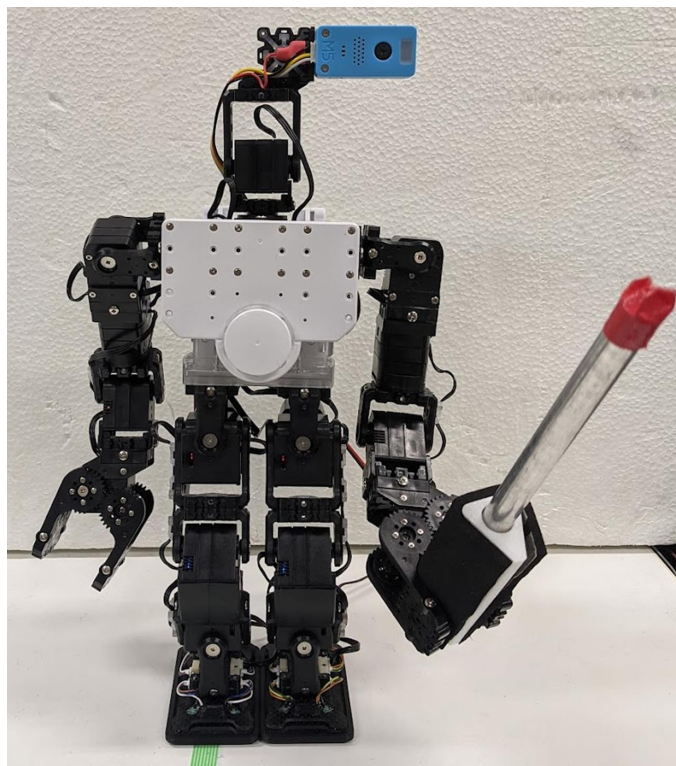


Fig. 4.4: KXR: The low-rigidity plastic-made axis-driven humanoid.

の回転による開閉機構を持つ。両脚にはそれぞれ5自由度、首には2自由度がある。

KXRに備わる冗長なセンサ群について述べる。まず、カメラとして頭部にはM5StickV (M5Stack Technology Co., Ltd.) が配置されており、カラー画像を取得することができる。次に、腰部にはIMUが配置されている。最後に、足平には圧力センサが四隅に配置されており、ZMPを算出することが可能である。

4.4 柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成要件

第2章で述べた柔軟性と冗長性の利点を増強するハードウェア開発に必要な点を再掲し、Fig. 4.5にまとめる。

- 時間的柔軟性

開発するハードウェアは、成長・組み替えが容易である必要がある。特に筋骨格ヒューマノイドは筋モジュールや筋経由点、骨格や関節までもモジュール化し、容易に構造を替え、様々な身体

構造でロボットを動作させることが出来るような環境を整える。その利点を活かし、手足のある全身筋骨格ヒューマノイドだけではなく、足に車輪や倒立振子がついたような特殊構造も開発し、その利点を示していく。

- 空間的柔軟性

筋骨格構造において、力の蓄積や柔軟接触ができるような非線形弾性要素を開発する必要がある。これまでの非線形弾性要素は金属とバネで構成されるため硬く、環境接触には向かなかったが、ワイヤとゴムのみで構成可能な非線形弾性要素を構築することで、柔軟接触行動を促進する。

- 制御的冗長性

冗長駆動が可能となるよう、多数の筋が配置可能なモジュール化を考えなければならない。そのため、筋の小型化や骨格と筋の一体化を目指す。また、骨格や筋同士の接続を単一のアタッチメントによって可能にし、筋を増やしていける構造を開発する。

- 感覚的冗長性

視覚や聴覚、接触覚等の多数の感覚器を開発し、これを搭載する。より多くの感覚器を搭載することで、後の身体図式学習に利用する。

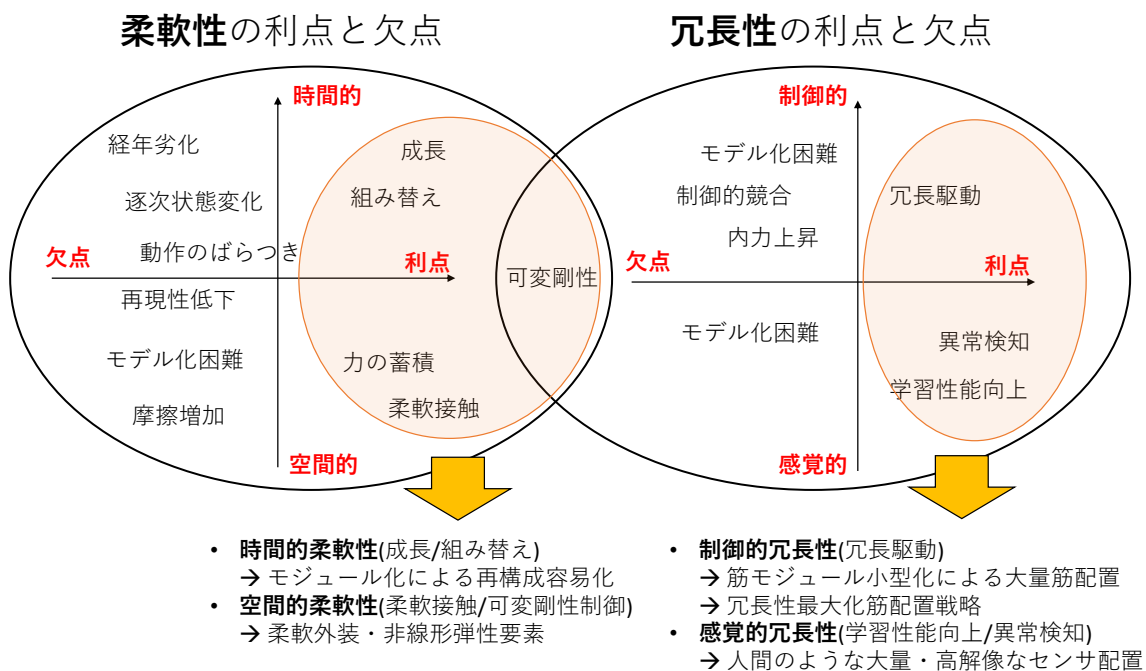


Fig. 4.5: The hardware requirements of the musculoskeletal humanoid with the redundancy and flexibility.

4.4.1 学習制御プラットフォームの構成要件

学習制御プラットフォームの構成要件は以下だと考える。

- 学習制御に用いることのできる大量・高解像度のセンサが搭載されている。
- 長期的な学習制御を実行できるだけの信頼性のあるハードウェア。
- 学習結果の評価のための冗長センサ配置。
- 様々な身体構成と再構成・筋配置変化について実験を試行できるプラットフォーム化のためのモジュール設計。

これらをまとめると、学習制御とその評価のための冗長センサ配置 (感覚的冗長性) と、構成・再構成可能かつ信頼性のあるモジュール化されたハードウェア設計 (時間的柔軟性) となる。これは、前述の柔軟性と冗長性を備えた筋骨格ヒューマノイドの構成要件と合致している。

4.4.2 開発する構成要素

これらの構成要件をもとに、本研究で開発した以下の構成要素について述べる。

- **筋モジュール**
筋配置を自由に変更可能かつ、信頼性のあるモジュール化された筋アクチュエータを開発する。また、筋と構造を一体化し、大量筋配置を可能とする筋アクチュエータの小型化も行う。
- **関節モジュール**
学習制御の評価に利用可能な関節角度センサを内包し、擬似的な球関節モジュールを開発する。5つのパーツを組み替えるだけで容易に身体構成を変化可能にする。
- **筋経由点ユニット**
筋配置を自由に変更可能とする標準化した筋経由点ユニットを開発する。
- **非線形弾性ユニット**
身体の柔軟性を確保する非線形弾性ユニットを開発する。ゴムとワイヤを用いて、それ自体が柔軟で環境接触に適した構造を開発する。
- **冗長性を備えた柔軟ハンド**
指先や手の平の力が計測でき、指が柔軟な切削バネにより構成されたハンドを開発する。またバネの圧縮を使った剛性可変機構も搭載する。
- **可動眼球ユニット**
高解像度カメラを搭載した可動する視覚を搭載する。

- 全周触覚を備えた足

足平にかかった力だけでなく、足の表面全周の感覚を持つ足構造を開発する。

- 臀部接触センサ

柔軟な臀部と環境の間の接触力を計測可能な臀部接触センサを開発する。

また、これらの構成要素により構築された、本研究で使用するヒューマノイドである、MusashiLarm, Musashi, MusashiOLegs, TWIMP, Musashi-W, Kengoro について詳細に述べる。最後に、筋配置の設計最適化手法についても述べる。

4.5 柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成要素

4.5.1 筋モジュール

センサドライバ統合型筋モジュール [152] と骨構造一体小型筋モジュール [115] について述べる。前者を先行研究として、後者を開発した。

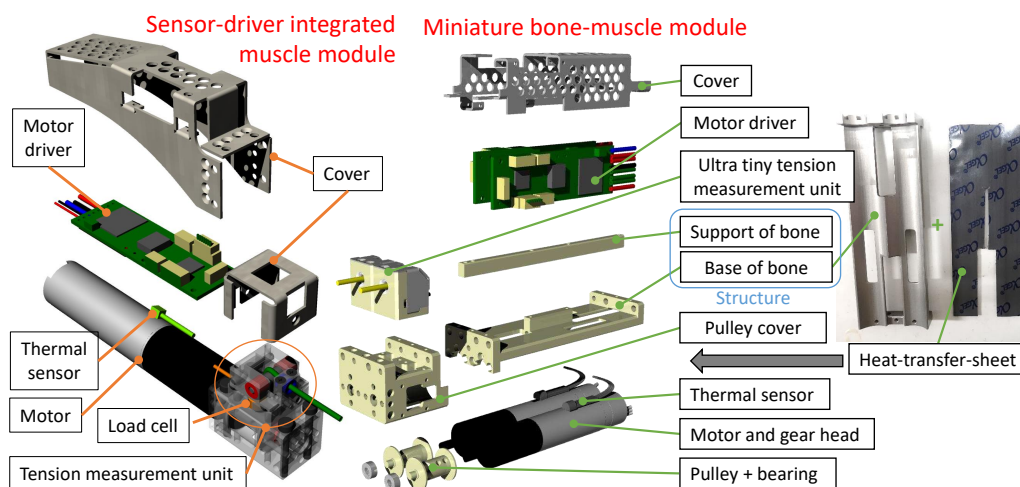


Fig. 4.6: Detailed design of the sensor-driver integrated muscle module [152] and miniature bone-muscle module [115].

まず、センサドライバ統合型筋モジュールの詳細を Fig. 4.6 の左図に示す。これは、アクチュエータとなる $\phi 22$ の BLDC モータ・筋を巻き取るプーリ・モータドライバ・温度センサ・筋張力測定ユニット・回路カバー等を一つのパッケージの中に収めることで、モジュール性・信頼性を格段に上げた筋モジュールである。アクチュエータはギア比を自由に変更することができ、本研究では基本的に 29:1

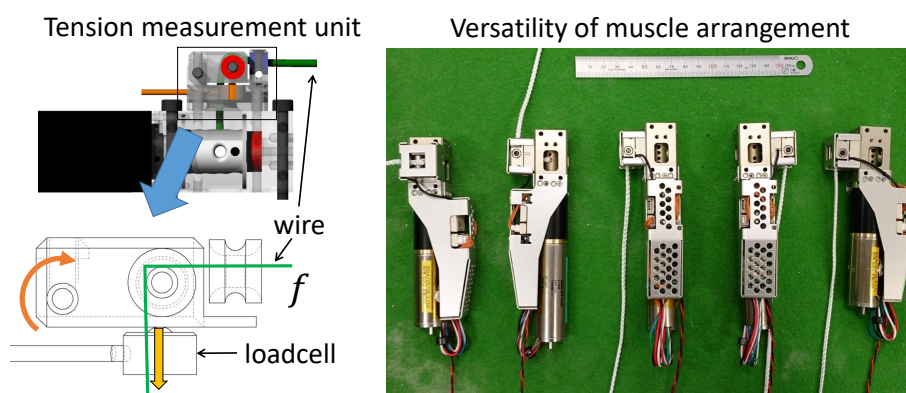


Fig. 4.7: The structure of the tension measurement unit, and the versatility of its arrangements to the muscle actuator [152].

か 53:1 を用いる。モータドライバは電流制御が可能であり、筋張力をフィードバックループに組み込んだ筋張力制御を行うことができる。筋には摩擦に強い化学繊維である Dyneema を使用している。筋張力測定ユニットの構造は Fig. 4.7 の左図のようになっており、軸中心の回転モーメントを使い、筋の張力をロードセルの圧力に変換することで 500 N までの筋張力を測定できる。また Fig. 4.7 の右図のように、筋アクチュエータに対して様々な方向から筋張力測定ユニットを取り付けることができ、様々な筋経路を実現できるという汎用性を持つ。

次に、骨構造一体小型筋モジュールの詳細を Fig. 4.6 の右図に示す。この筋モジュールが先の筋モジュールと最も異なる点は、この筋モジュール自体を骨格として用いることができる点である。アクチュエータはセンサドライバ統合型筋モジュールのアクチュエータより一回り小さな $\phi 16$ の BLDC モータ 2 本を使用している。複数のアクチュエータを一つのモジュール内に収納することで、効率的な空間配置を取ることができ、その 2 つのアクチュエータの間を金属で満たしている。“Base of bone” と “Support of bone” が筋モジュールの骨格となり、この筋モジュールを縦や横に接続することで、筋モジュールのみでコンパクトに骨格を構成できる点において優れている。また、小さなアクチュエータを使うことによる筋張力のビハインドを、“Heat-transfer-sheet” を介してアクチュエータの熱を “Base of bone” に逃がすことで補っている。

これら筋モジュールは、モジュール化という観点で優れており、信頼性・汎用性が高いことから、筋骨格プラットフォームのアクチュエータ基盤となる。

4.5.2 関節モジュール

開発した擬似球関節モジュール [87] について述べる.

関節モジュールの構成要件と先行研究

筋骨格ヒューマノイドの関節として適用でき、冗長センサとして関節角度を測定することのできる関節モジュールの要件を本研究では以下のように設定した.

- (1) 人間のように筋肉が関節を覆うことができるよう、関節は球形である.
- (2) 各関節角度を直接測定することができる.
- (3) コンパクトに回路等を含んでパッケージ化されており、単体として簡易に動作する.
- (4) 汎用性があり、筋骨格ヒューマノイドの全関節をこの関節モジュールで担うことができる.

(2) に関してはいくつかの方法が考えられる. 通常のヒューマノイドであればそれぞれの関節は独立しており、それぞれエンコーダやポテンショメータがついた構成が一般的である [153]. また、IMU を用いて加速度と地磁気から関節角度を測る方法も存在するが、yaw 軸回転のドリフトを抑えるため地磁気を用いており、冗長な多数の筋のモータが体中に存在する筋骨格ヒューマノイドに適用するのは難しい. また、Urata らは球関節の角度を小さなカメラを用いて測定する優れたシステムを開発しているが [99], システムの複雑性等から本研究では採用しない. 本研究では、球関節という形を取りながらも、その中に収納できるよう小さなポテンショメータを用いて通常の軸駆動型ヒューマノイドと同様の構造で関節角度を直接測定可能な擬似球関節モジュールの設計を行う. しかし、通常の軸駆動ヒューマノイドの関節と同様、特異点が生まれてしまうという問題のみ、本研究では妥協する. (3) については、ポテンショメータを読む回路を擬似球関節モジュールの中に内包しパッケージ化することで、単体として USB を繋ぐだけで動作するコンパクトなモジュール化を行う. (4) について、人間の関節は、複雑な肩甲骨関節等を除けば4種類の関節のみによって、シンプルに Fig. 4.8 の左図のように表せる. これらの主要な関節群を少数の部品の組み換えによって全て表現可能な関節モジュールを作ること考える. そこで、Fig. 4.8 の右図のように、これら4種類の関節モジュールを、関節モジュールの核となる2種類の中心パーツと、3種類の回転パーツを複数組み合わせることで構築することができる考えた. また、この5種類のパーツを組み替えることで、ある自由度を消したり、増やしたり等、この4種類以外にも様々な構成の関節モジュールを構成することが可能である.

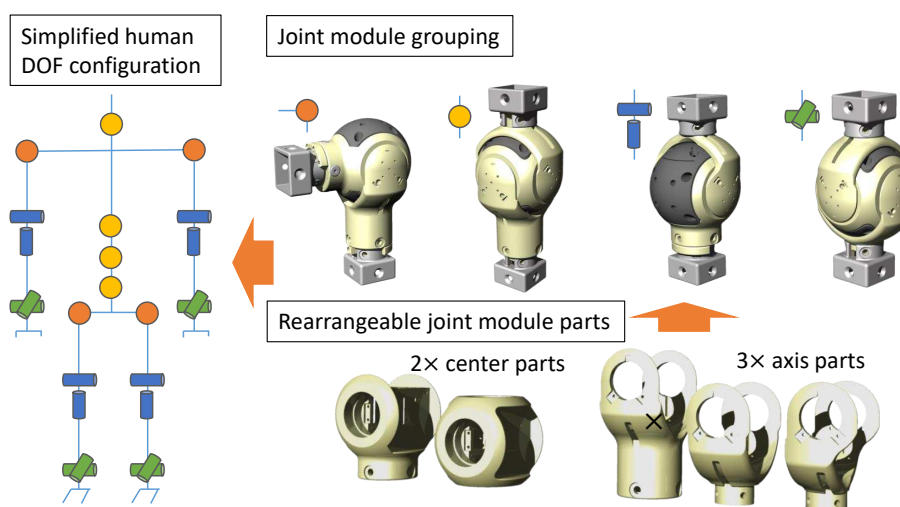


Fig. 4.8: Grouping of simplified human joints, and application of joint modules [87].

関節モジュールの設計

実際に開発された関節モジュールを Fig. 4.9 に示す. Fig. 4.9 にあるのは, 左から, 手首, 肘, 肩に用いる関節モジュールである.

関節モジュールの構造の詳細は Fig. 4.10 となっている. これは肩の関節モジュールの構造であるが, 基本的には4つのリンクに分かれており, その間に3つの回転軸が存在する. 構造材は A7075 を使用して切削されており, 他の素材の部品としてはベアリング・スペーサ・3D プリンタで作成された回路保護カバーが存在する. ポテンショメータを読む Potentiometer Board はそれぞれの関節軸ごとに配置されており, それらを統合する Potentiometer Control Board が1つの関節モジュールに必ず1つ存在し, モジュール内部に収納されている. また, ケーブルの配線は Fig. 4.10 の右図にあるような配線となっている. Potentiometer Control Board と Potentiometer Board を繋ぐ配線は全て関節モジュールの中を通過しており, 外部と USB 通信を行うためのケーブルが1本だけ関節モジュールから外に出ている. Potentiometer Control Board には IMU が搭載されており, 並進加速度・回転角速度・地磁気の9軸の値を得ることができる.

4.5.3 筋経由点ユニット

筋経由点ユニットは, ある一つのベースとなる構造に対して, 周りに3つの連続したベアリングを備える一本の軸を配置した構造として開発している [87]. ベアリングをベースの構造に対して組み込まず, 直接筋を折り返す場所に使うことで, コンパクトな筋経由点ユニットを実現している. 筋経由点

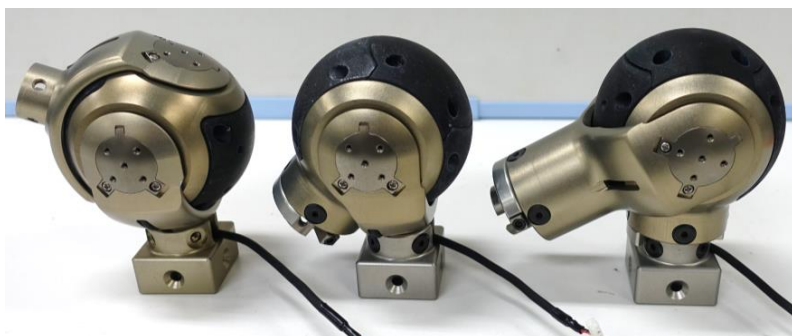


Fig. 4.9: Overview of the newly developed joint modules [87].

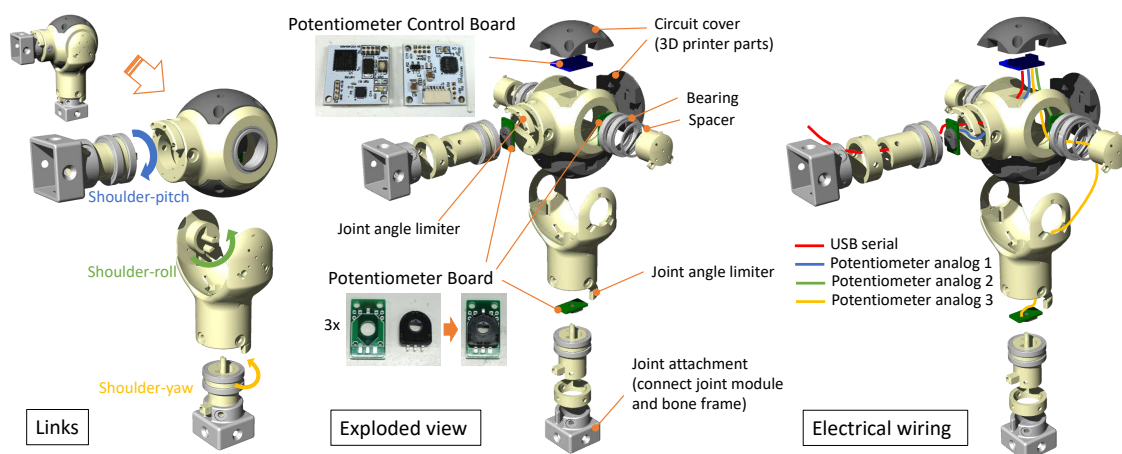


Fig. 4.10: Detailed design of the joint module, for the shoulder joint [87].

ユニットの構造は、それを取り付けるネジの方向と、筋の折り返しの方向の違いによって、Fig. 4.11 の上図のような3種類のみ作成すれば事足りる。Fig. 4.11 の下図に開発した筋経由点ユニットを示すが、これらは一本のネジの方向と、連続したベアリング付きの軸の方向が定まった際に、その中で最もコンパクトな形状を考えた結果である。この3種類の筋経由点ユニットを、骨格のネジ穴の方向と、折り返したい筋の方向から選定し、それぞれ適用することで、筋折り返し構造を実現可能である。

4.5.4 非線形弾性ユニット

開発した非線形弾性ユニット [87] について述べる。

これまで開発されてきた非線形弾性ユニットを Fig. 4.12 に示す。Fig. 4.12 の左2つは [108] に用いられた Nonlinear Spring Tensioner (NST) であり、一番右は長田らによって開発された、腱に対して後

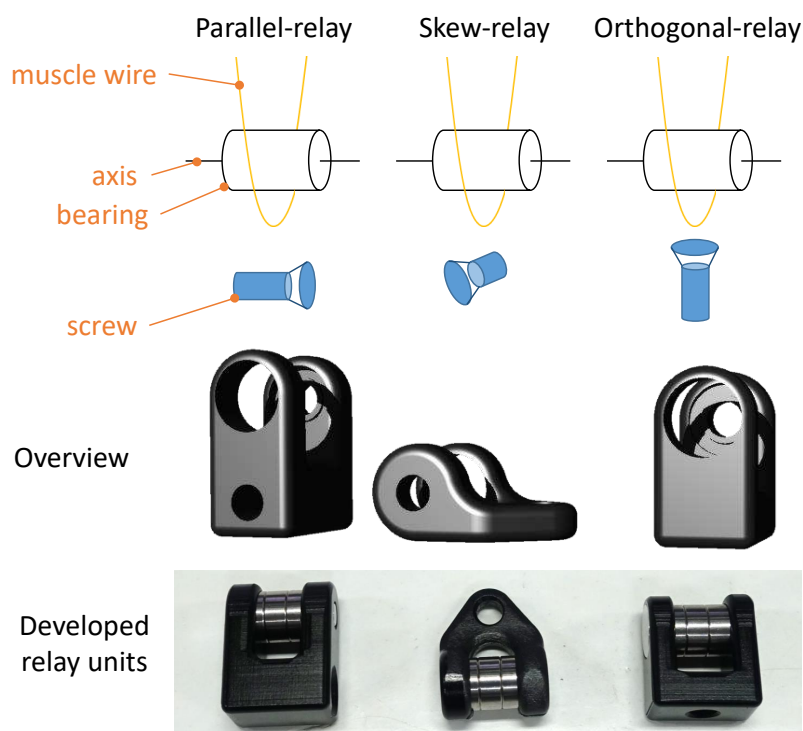


Fig. 4.11: Standardized 3 muscle relay units: Parallel-relay, Skew-relay, and Orthogonal-relay [87].

から装着することができるアドオンユニットである [154]. これらは幾何的に非線形弾性要素を実現できる一方, 基本的に金属とばねで構成されており, 大きく, 硬い構造をしているため, 人体模倣型筋骨格ヒューマノイドの身体への適用は難しい.

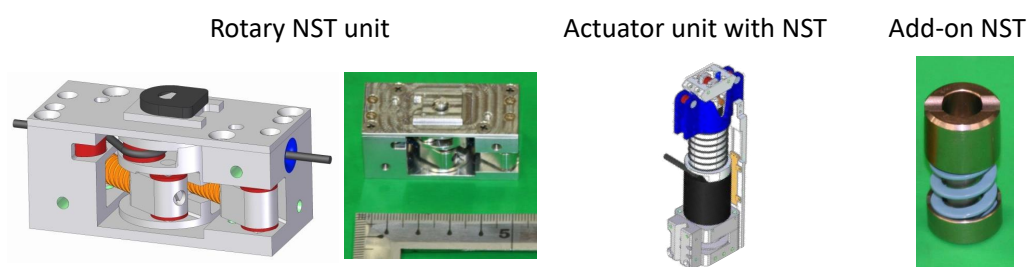


Fig. 4.12: Previous nonlinear spring tensioner units [108, 154].

非線形弾性ユニットの設計

本研究では, コンパクトで柔軟な非線形弾性要素を実現するための, 非線形弾性ユニットの構造について考える. 最もそれに近い構造は [107] でも用いられている, O-ring を使った方法であろう. しか

し, [107] のユニットでは非線形弾性を表現しきれていない。

Fig. 4.13 に本研究で開発した非線形弾性ユニットの詳細を示す。一つ目は Oring-NEU であり, これは基本的に [107] で用いられていたものと同じであるが, いくつかの素材を試し, ニトリルゴムのみが正しく非線形弾性を表現することができた。しかし, ニトリルゴムは大気中に含まれるオゾンや湿気, 及び温度によって劣化を引き起こし, クラックが入ることでゴム自体が切れやすくなってしまうという問題があった。また, この構造はゴムを引き伸ばすことによって非線形弾性を得るため, ゴムの強度によって筋張力に限界が生じてしまう。

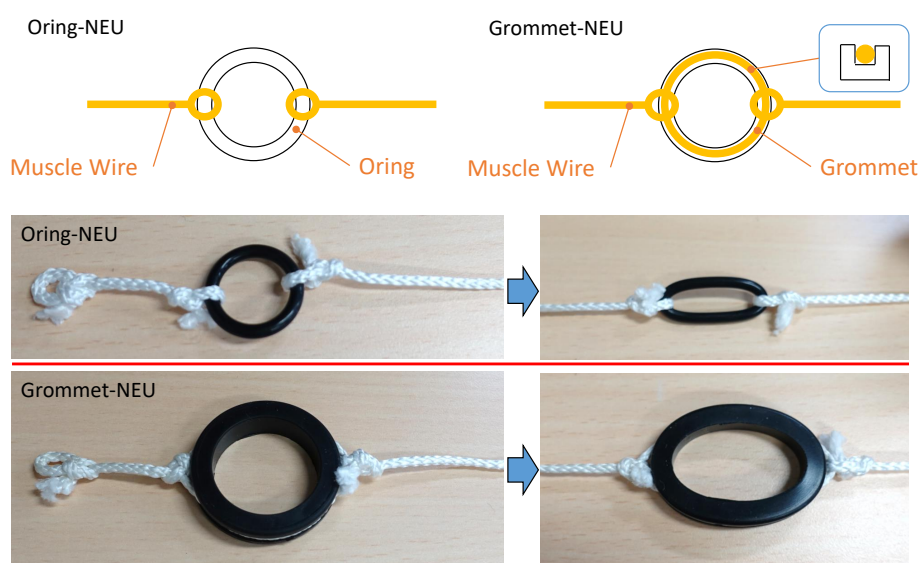


Fig. 4.13: Developed nonlinear elastic units: Oring-NEU and Grommet-NEU [87].

そこで, 2つ目の Grommet-NEU を開発した。これは, 筋のワイヤによってグロメット構造を圧縮することで非線形弾性を表現する。この構造においては非線形弾性を幾何的に実現できるため, 素材を自由に選ぶことができる。また, Oring-NEU と違い, ゴムの圧縮のみを用いるため強度が格段に向上している。本研究では耐候性や耐水性のあるエチレンプロピレンゴムを用い, 非線形弾性要素を実現している。

非線形弾性ユニットの特性

最後に, これらの非線形弾性要素を用いた筋構成の非線形弾性パラメータの同定を行った結果を Fig. 4.14 に示す。Fig. 4.14 の上図のように, 絶対筋長を固定し, 筋を巻き取り, 筋が巻きとられた量・フォースゲージの力の値を調べることで, パラメータを同定する。筋の絶対長さに対して筋長の相対

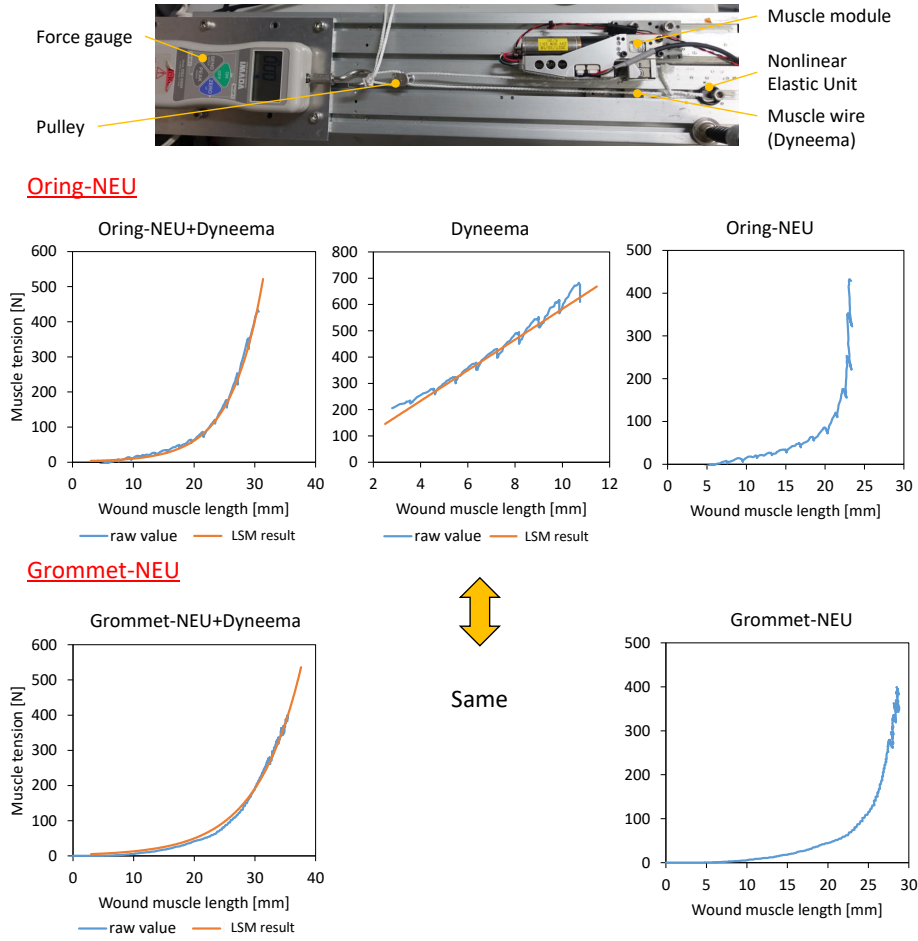


Fig. 4.14: Identification of nonlinear elastic parameters [87].

的な変化は十分小さいものと仮定する。Fig. 4.14 の左図にはこの筋構成全体の筋長変化・筋張力変化のグラフ, 中図には非線形弾性要素を外した際の結果を示している (なお, 絶対筋長は 480 mm に固定しており, 筋張力 0 のときに筋の伸びは 0 という仮定から原点を揃えている)。このとき, Dyneema の伸びは線形, Dyneema のばね定数は絶対筋長に反比例するという仮定を用いて最小二乗法を行うと, Dyneema の特性は以下のように推定できる。

$$f = a_d \Delta l_d / l_{abs} \quad (4.1)$$

ここで, f [N] はフォースゲージから得た筋張力, a_d は単位長さ辺りの Dyneema のばね定数, Δl_d [mm] は筋の伸び, l_{abs} [mm] は筋全体の絶対長さであり, 本推定では $a_d = 2.8E + 4$ と推定された。また, こ

の筋構成全体の特性は指数関数による最小二乗法によって以下のように求まる。

$$f = a_m \exp(b_m \Delta l_m) \quad (4.2)$$

ここで, a_m, b_m は非線形特性を表す定数であり, 本推定では Oring-NEU において $a_m = 1.34, b_m = 0.19$, Grommet-NEU において $a_m = 3.32, b_m = 0.14$ と推定された. Eq. 4.1 と Eq. 4.2 を用いることで, 非線形弾性要素の特性は以下ようになる。

$$\Delta l = \Delta l_m - \Delta l_d = \frac{1}{b_m} \log\left(\frac{f}{a_m} + 1\right) - \frac{f l_{base}}{a_d} \quad (4.3)$$

ここで, l_{base} は今回の推定で固定した筋全体の絶対筋長の 480 mm である. よって, Eq. 4.3 と Eq. 4.1 の効果を足し合わせることで, 本研究で用いる筋構成の特性がわかる. 本推定で用いている f はフォースゲージで測れる張力であり, プーリを介して折り返しているため, 実際には張力センサユニットからはこの f の半分の力が計測される. また, 実際には, 発泡性カバーや摩擦等の影響により, 実機とは異なることがあり, 実機による非線形パラメータの学習をする必要がある [124].

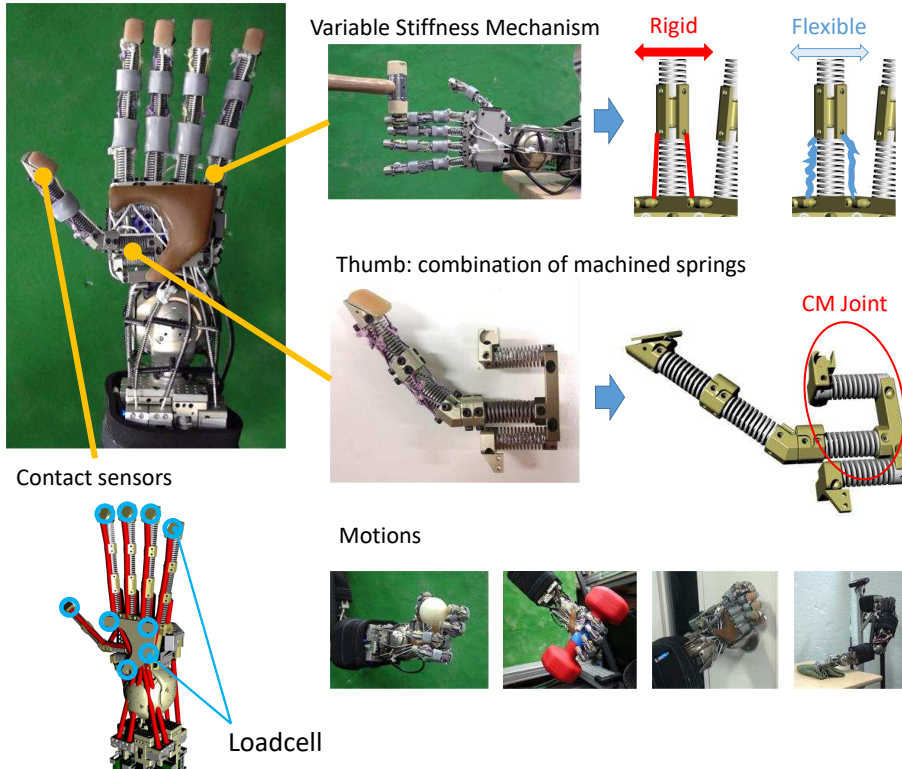


Fig. 4.15: The flexible hand structure including the thumb with combination of machined springs and fingers with variable stiffness mechanism [116].

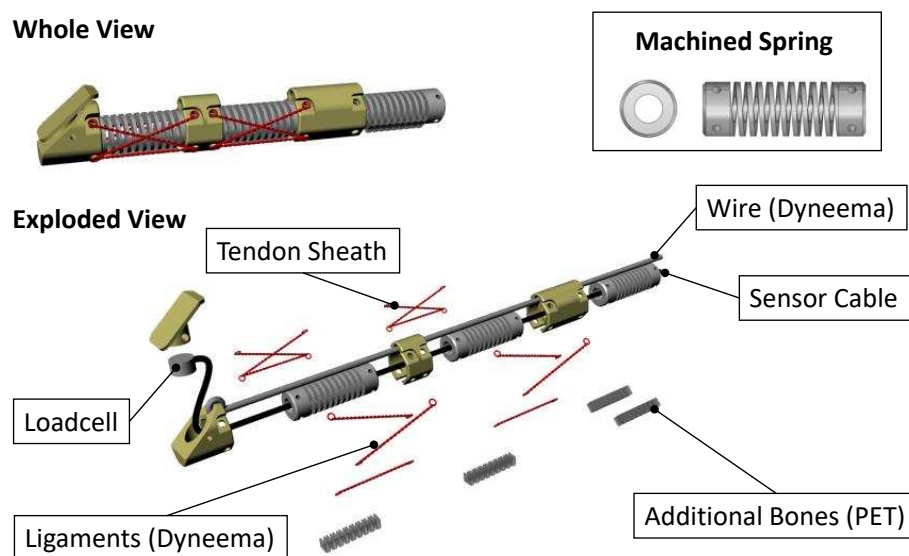


Fig. 4.16: The flexible finger structure with machined springs [116].

4.5.5 冗長センサを備えた柔軟ハンド

開発した冗長センサを備えた柔軟ハンド [155, 116, 156] を Fig. 4.15 に示す。このハンドは4つの特徴を備える。

まず、全ての指の関節が切削ばねにより構成されている。詳細な指の構造については Fig. 4.16 に示す。切削ばね同士が金属パーツによって繋がり、筋ワイヤである Dyneema が沿っている。指の先端にはロードセルがあり、切削ばねの中にはそのケーブルが通っている。切削ばねは様々な方向に対して曲がってしまうため、異方性を出すために糸をねじった腱鞘が配置されている。また、筋張力によりバネが縮むのを防ぐために、バネの一方向に PET の拘束部品を詰めている。この構造により、ハンマーで叩いても壊れない、柔軟かつ丈夫な指が構成される。

また、親指の構造は特徴的である。親指はそれ以外の指に比べて大きく動くが、これを単一の切削バネによって表現することは難しい。そこで、この切削バネを2つ組み合わせて配置することで、親指の広い可動域を実現している。

次に、指の可変剛性制御が可能である。それぞれの指の根本の関節には腱鞘がついておらず、通常指は柔らかく動く。一方、この根本の関節の両端に筋が配置されており、これを引っ張ることで切削バネが縮み、関節が硬くなる。この指の可変剛性制御を活用することで、外力や環境に適応的な動作が可能となる。

最後に、この柔軟ハンドには、指先と手のひらに合わせて9個の接触センサが配置されている。同様

に、筋の筋張力・筋長・筋温度を測ることができ、これらの情報を使った学習制御に利用する。

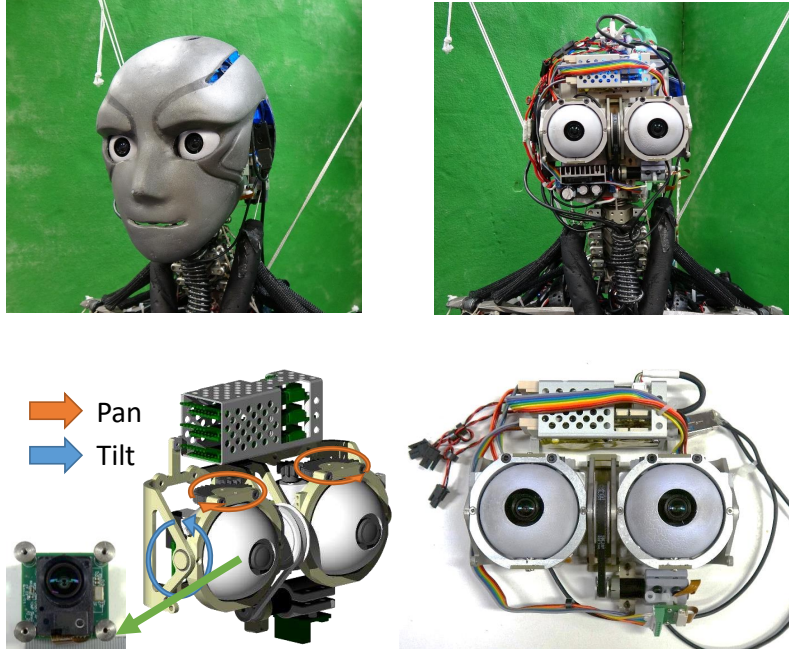


Fig. 4.17: The structure of movable eye unit with high-resolution cameras [112].

4.5.6 可動眼球ユニット

開発した可動眼球ユニット [112] の構成を Fig. 4.17 に示す。それぞれのロボットの仮面の下に収まるこの可動眼球ユニットは、それぞれの目の pan joint と一つの tilt joint の3つの自由度を有している。カメラは高性能な DFK-AFUJ003 (ImagingSource, Inc.) を使用しており、解像度・焦点・露光の変更等がソフトウェア側から可能となっている。目の角度を変更することができるため、広範囲の物体を認識することができる。高解像度画像の一部を切り出し、サイドミラーに映る人のような、細かいものを認識することも可能である。また、可動眼球を使うことで、輻輳を使った物体との距離認識も可能である。

4.5.7 全周触覚を備えた足

開発した全周触覚を備えた足構造 [157] を Fig. 4.18 に示す。この足はつま先と踵それぞれにコア・シェル構造をもった6軸センサユニットを持つ。このユニットはコアとシェルに分割され、このコアとシェルの間に1軸力センサであるロードセルが12個挿入されている。この12個のロードセルから

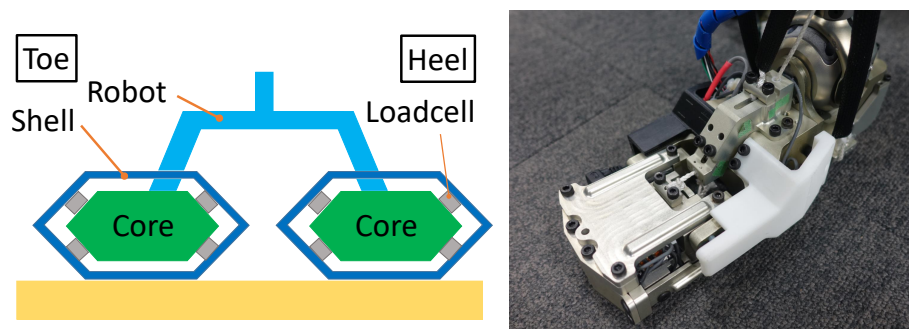


Fig. 4.18: The core-shell foot structure with all-around tactile sensor using multiple loadcells [157].

6 軸の力を計算する。また、通常の足構造とは違い、足の表面全体の力を計測することが可能であり、つま先に乗った物体等を検知することができる。

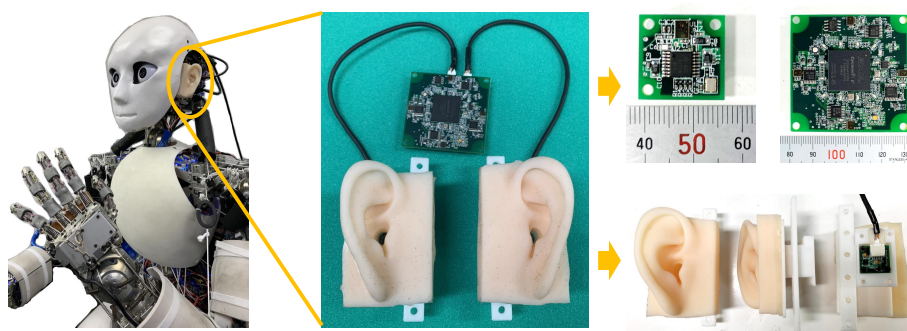


Fig. 4.19: The overview of the binaural ear unit [158].

4.5.8 人体模倣型両耳ユニット

開発した人体模倣型両耳ユニットの構造 [158] を Fig. 4.19 に示す。このユニットは外耳構造、MEMS マイク基板、FPGA 処理基板により構成されている。外耳構造はシリコンゴムの耳介部分と MEMS マイク基板を格納する 3D プリント部品から構成される。人間の耳介と同じ形の複雑な凹凸を持った形状となっており、音源方向に依存するフィルタとしての機能を再現している。FPGA 処理基板はマイクから計測された音を高速に処理・通信することが可能である。三半規管としての IMU も搭載されている。

開発された両耳ユニットによるニューラルネットワークを使った音源方向推定について説明する。このニューラルネットワークを SSDENet (Sound Source Direction Estimation Network) と呼ぶ。左右のマイクから得られた音信号を高速フーリエ変換によってスペクトルに変換する。それぞれをスペクト

ル周波数ごとに正規化し、両耳間レベル差 (Interaural Level Difference, ILD) と両耳間位相差 (Interaural Phase Difference, IPD) を算出する。ある音が鳴った時に、その音の周波数を計算し、学習した SSDENet にその周波数におけるこの両耳の ILD と IPD を入力することで、全方向に対する音源存在確率を出力することが可能である。

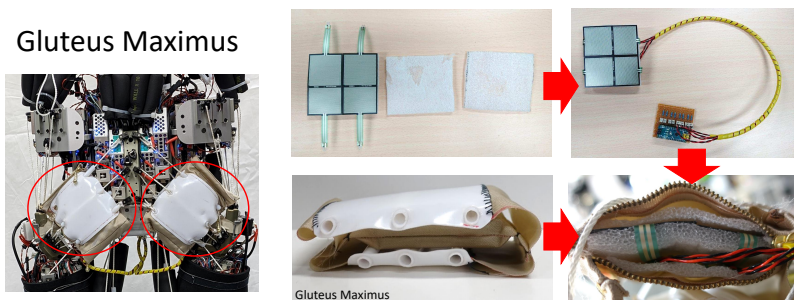


Fig. 4.20: The structure of the buttock sensor using FSR.

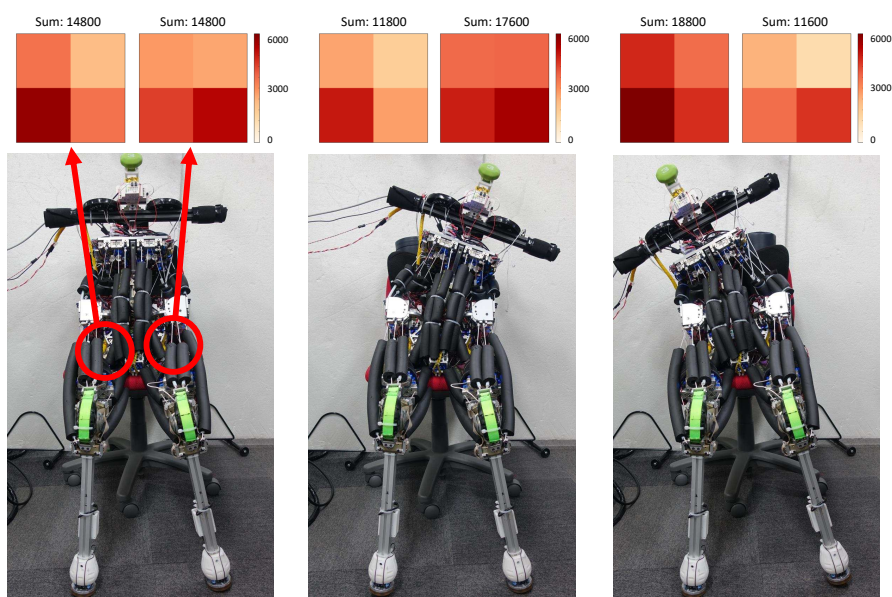


Fig. 4.21: The value of the buttock sensor when sitting at various poses.

4.5.9 臀部接触センサ

開発した臀部触覚センサを Fig. 4.20 に示す。薄い発泡性素材を3枚用意し、その一つに4つの大きな圧力センサ FSR[®]406 を装着する。残りの2枚でそれを挟み、大殿筋の面状骨格間構造内に挿入す

る。ケーブルは恥骨結合部を通り、仙骨付近の Arduino によってアナログ値を計測する。FSR の感度は非線形であり、力が大きいほど電位差は小さくなってしまうため、本研究では $\exp(A/100)$ のような形で補正した値を用いる。なお、 A は 10 bit のアナログ値であり、圧力センサは広い接触面を持つため、正確な力の大きさはわからない。

Fig. 4.21 に、様々な椅子への座り方における臀部接触センサの値、また、左右それぞれにおける 4 つの FSR 出力の合計を示す。それぞれの姿勢によって臀部接触センサの値は異なり、環境と臀部の接触力を測定することができている。この計測値は冗長感覚として学習に用いることができる。

4.6 柔軟性と冗長性を備えた筋骨格ヒューマノイドのハードウェア構成

4.6.1 MusashiLarm

設計

開発した単腕筋骨格ロボット MusashiLarm の全体像 [87] を Fig. 4.22 に示す. 柔軟な身体によりハンマーのような撃力を伴う道具や動作が扱えると同時に, その剛性を高めることで重いダンベルを持ち上げることも可能である. この MusashiLarm を対称につけた双腕筋骨格ヒューマノイドを MusashiDarm と呼ぶ.

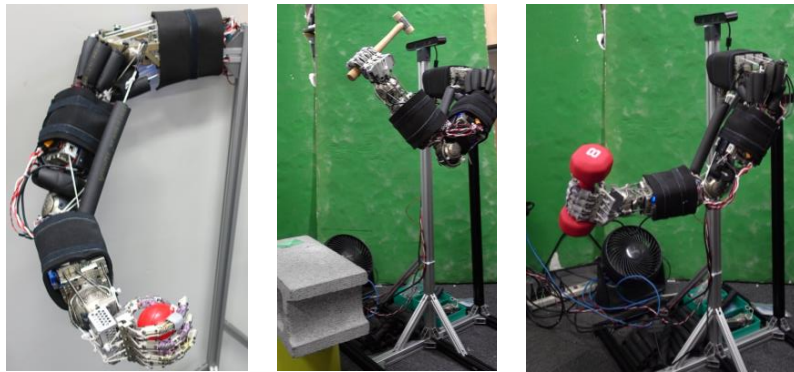


Fig. 4.22: Overview of MusashiLarm.

MusashiLarm の設計詳細を Fig. 4.23 に示す. 本研究で開発した筋骨格上肢は基本的に 4 つのリンク・3 つの関節によって構成されている. それぞれのリンク名を Scapula, Humerus, Forearm, Hand と呼ぶこととする. Scapula・Humerus リンクの構造体はアルミの汎用骨格 (Bone frame) によって構成されており, その周りにリンクごとに 5 つのセンサドライバ統合型筋モジュール (Sensor-driver integrated muscle module) が筋アタッチメント (Muscle attachment) を介して装着されている. Muscle attachment を介して, Bone frame と筋モジュール (Muscle module), また, Muscle module と Muscle module 同士を合体させることが可能である. また, 関節モジュール (Joint module) と Bone frame は Joint attachment を介して接続されており, アタッチメントの形を変えることで容易に構成を変化させることが可能である. Forearm リンクの構造は筋としても骨格としても用いることができるという利点を持った骨構造一体小型筋モジュール (Miniature bone-muscle module) 4 つによって構成されている. 筋としてだけでなく骨格になることができる利点を活かし, Bone frame は用いず, Muscle module のみで構成された非常に簡易な構成となっている. 手は切削ばねを用いた柔軟ハンド [116] である.

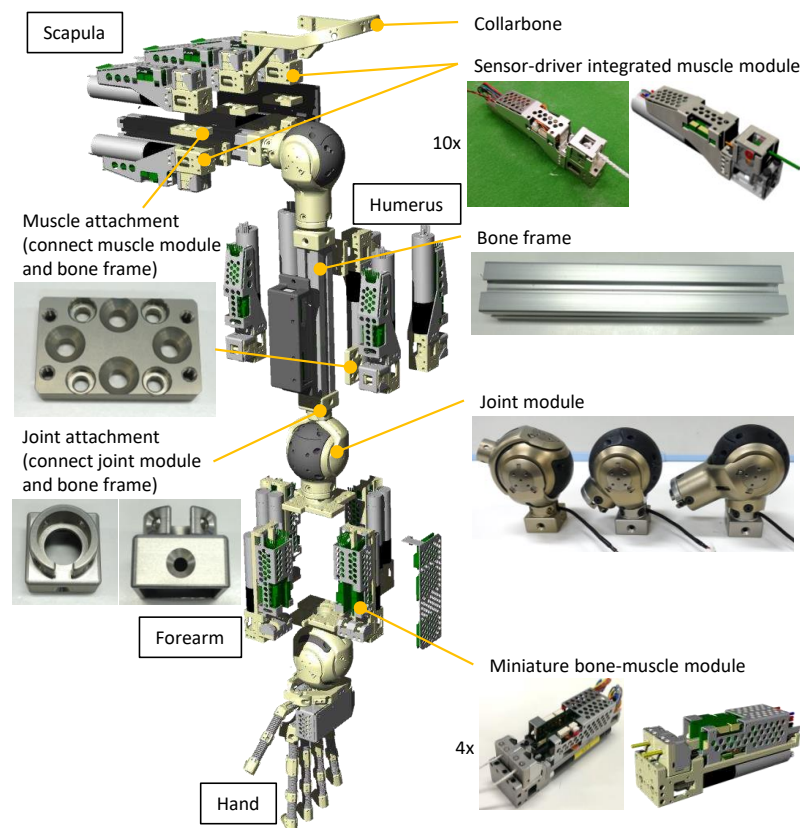


Fig. 4.23: Detailed modular design of MusashiLarm [87].

このように、これまで開発したモジュール群とアルミ汎用骨格、そしてそれらを接続する Muscle attachment・Joint attachment のみにより、簡易に構成・再構成可能な身体を構築することができる。

筋構成

MusashiLarm の筋配置を Fig. 4.24 に示す。Scapula・Humerus リンクにそれぞれ 5 本の筋, Forearm リンクに 4 つの Miniature bone-muscle module, つまり 8 本の筋が搭載されており、合計 18 本の筋が存在する。これらは人体の基本的な筋肉を模倣しており、多関節筋も有している。Miniature bone-muscle module はアクチュエータとして BLDC モータの 60W 128:1 を, Sensor-driver integrated muscle module は BLDC モータの 90W 29:1 を使用している。非線形弾性要素については Scapula と Humerus に配置された 10 本の筋にのみ適用されている。

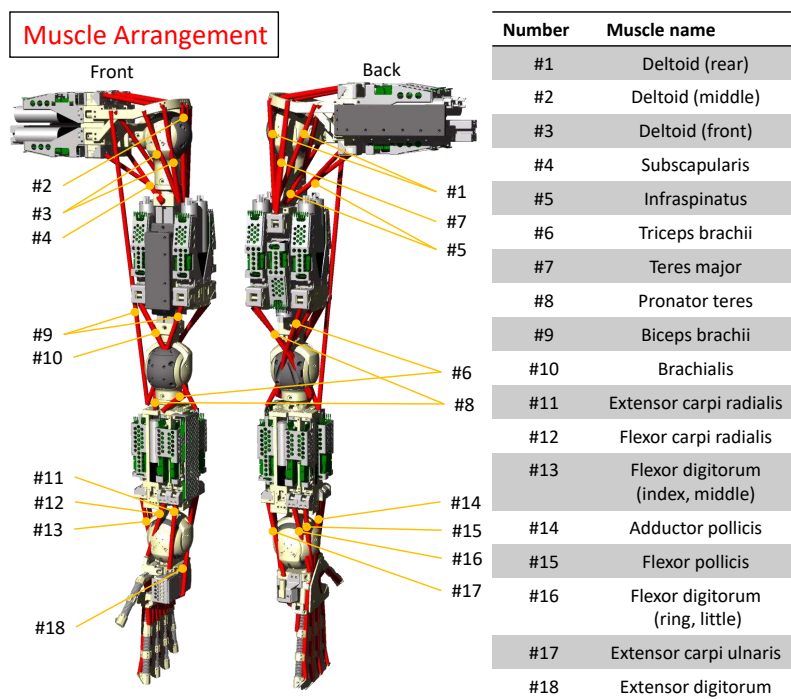


Fig. 4.24: Muscle arrangement of MusashiLarm [87].

回路構成

回路の構成は Fig. 4.25 のようになっており、新しい要素をすぐに繋げられる構成としている。全体のシステムは USB 通信によって行われており、体に配置された USB ハブにモータ制御のための Control Board, 手先感覚のためのロードセルの信号を増幅してまとめる Loadcell Board, 関節モジュール内のポテンショメータをまとめる Potentiometer Control Board が接続している。Control Board は臆悟郎 [86] から変更されており、全二重通信と半二重通信が両方できるような構成となっている。現状 Motor Driver が半二重通信のみ可能なため、通信量の限界から Control Board を介して Motor Driver は最大 3 つのみデージーチェーンで接続可能である。また、Loadcell Board は 12 個までのロードセルを読むことができ、Potentiometer Control Board は 4 つまでのポテンショメータを読むことができる。冗長なセンサ群として、Potentiometer Control Board はポテンショメータの他に自身に搭載された IMU (MPU9250, InvenSense, Inc.) の情報も上位に送っている。

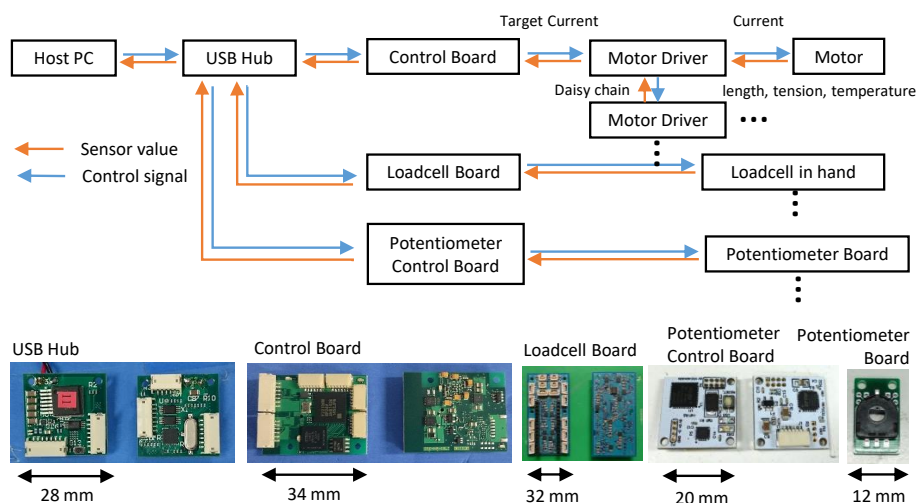


Fig. 4.25: Circuit configuration of MusashiLarm [87].

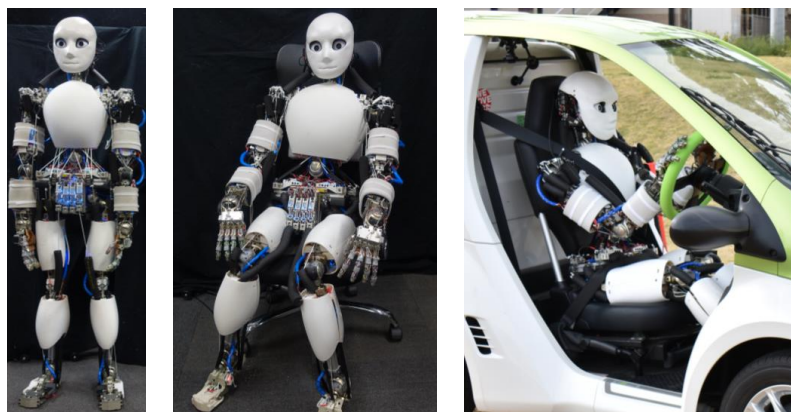


Fig. 4.26: Overview of the whole-body musculoskeletal humanoid Musashi.

4.6.2 Musashi

設計

開発した筋骨格ヒューマノイド Musashi [87] を Fig. 4.26 に示す. 筋骨格単腕ロボット MusashiLarm のモジュール化設計を全身にまで拡張し, 目や耳, 手, 足等の特殊構造を追加した構造を持つ. 身長は 159 cm, 体重は 40.1 kg である.

筋骨格ヒューマノイド Musashi の詳細な設計を Fig. 4.27 に示す. 左から, 全体像, 筋ワイヤを除いた身体, 筋ワイヤ・外装を除いた身体, 筋ワイヤ・外装・筋モジュールを除いた身体である.

基本的には MusashiLarm の構造設計をそのまま全身に拡張しているが, 下半身は筋構成が多少異なる.

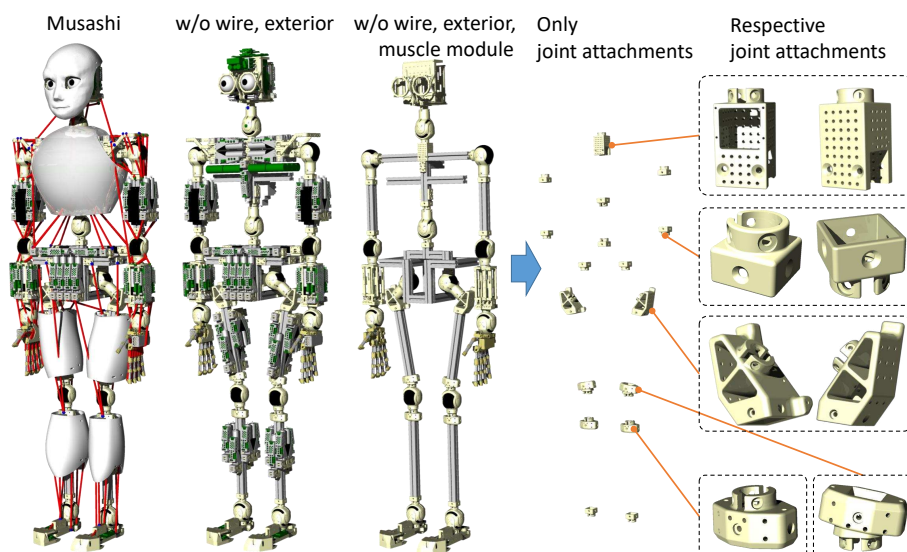


Fig. 4.27: Overview of the body structure of Musashi with several joint attachments [87].

る。下半身の筋モジュールのアクチュエータのギア比は全て 53:1 であり、非線形弾性要素は含まれていない。

また、筋モジュール・関節モジュール・汎用骨格等を除いた Joint attachment のみの身体構造を Fig. 4.27 の右図に示す。MusashiLarm から追加された Joint attachment は 4 種類であり、上半身と合わせて全 5 種類を作成している。よって、Muscle attachment を含め、全 6 種類のアタッチメントと関節モジュール・筋モジュール・筋ワイヤユニット・アルミ汎用骨格のみで全身の骨格を構成することができた。Fig. 4.27 の左図に示すように上腿・下腿・胸郭・顔は 3D プリンタパーツで覆われている。

特徴

Fig. 4.28 に示すように、筋骨格ヒューマノイド Musashi は 4.5 節で述べた様々な特徴的構造有している。

頭は可動眼球ユニット [112] を備え、視覚には解像度や焦点を変更できる多機能なカメラを有している。耳は人体模倣型両耳ユニットを備え、複雑な耳介構造による音源方向フィルタを学習することで、3次元音源方向推定が可能になる [158]。手は切削ばねによる柔軟で強靱な関節機構、冗長な接触センサ、指の剛性可変機構を持つ五指ハンド [116] である。最後に、足はコアとシェルによって構成された 6 軸力計測モジュールが踵とつま先についた全周触覚を備えた足構造を持つ [157]。

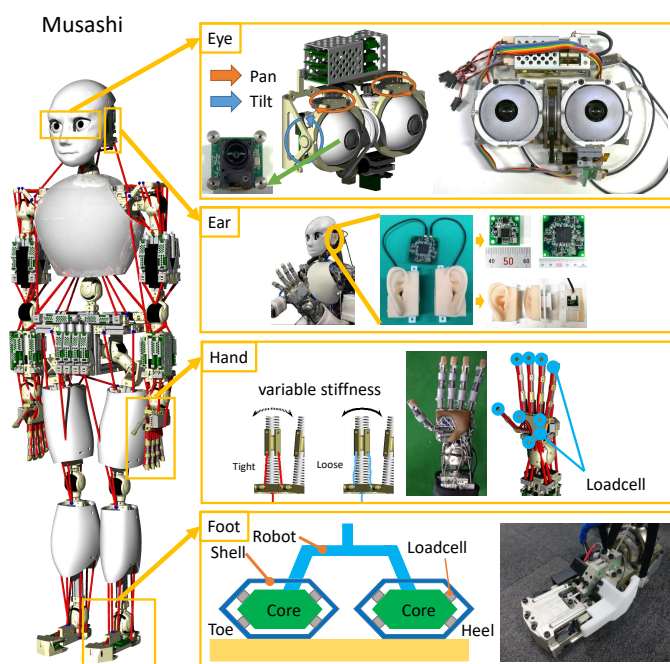


Fig. 4.28: Overview of the characteristic structures of Musashi: binocular eye unit, binaural ear unit, flexible hands with machined springs, and core-shell foot unit with all-around tactile sensor.

4.6.3 MusashiOLegs

開発した面状骨格間構造を有する筋骨格ヒューマノイド脚 MusashiOLegs [118] を Fig. 4.29 に示す. MusashiOLegs には腰から下の足首を除いた関節のみが実装されており, その自由度は 13 である. 身長はカメラの位置までで 151 cm, 体重は 27.1 kg である. 大殿筋や膝蓋靱帯, 側副靱帯や腸骨大腿靱帯が新たな面状骨格間構造によって構成されている点が新しい. それぞれの足裏には 4 つのロードセルが配置されており, 接触力を計測することが可能である. MusashiOLegs の筋配置を Fig. 4.30 に示す. 人体の下肢における主要な筋肉を模倣した筋配置をしている. 特に特徴的なのは, 腰関節から股関節までを跨ぐ多関節筋である腸腰筋が実装されている点である.

面状骨格間構造の詳細を Fig. 4.31 に示す. まず, 腸骨大腿靱帯は寛骨臼の縁から大腿骨転子までの間を大腿骨頭に螺旋状に巻き付く靱帯である. これを面状の高強度ナイロンである鎧布®(TORAY, Inc.) によって実装している. これにより, 複雑な股関節の伸展制限を行うことができる. 次に, 膝の側副靱帯は同様の鎧布® で実装されており, これが終末強制回旋構造 [119] を生み出す. 面状構造であるため, 膝の伸展時には膝がロックされるが, 屈曲時には側副靱帯が緩められ, 回旋方向への可動域が生まれる. 最後に, 大殿筋は臀部を覆い, 関節の伸展や外旋を担う重要な抗重力筋である. 線状の筋肉

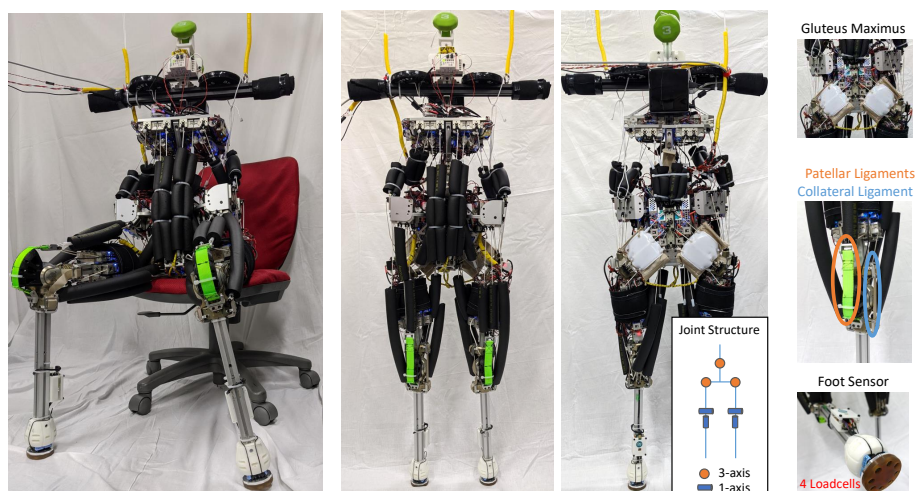


Fig. 4.29: The musculoskeletal humanoid MusashiOLegs.

の集まりでは、股関節の屈曲時に筋が股関節から逸脱してしまうが、これを面状骨格間構造によって実装することで、モーメントアームを保つことが可能である。また、柔軟素材により大殿筋を実装することで、環境接触に適した構造となる。加えて、この大殿筋には4.5.9節の臀部触覚センサが挿入されており、環境と臀部の接触力を測定することが可能である。

4.6.4 TWIMP

TWIMP (Two-Wheel Inverted Musculoskeletal Pendulum) [159] は上半身に筋骨格双腕 MusashiDarm, 下半身に倒立二輪を持つ筋骨格ヒューマノイドである (Fig. 4.32).

先行研究

これまでの先行研究を、筋骨格ヒューマノイド、倒立二輪、車輪とマニピュレータを組み合わせたタイプのロボットに分けて議論する。

これまで様々な筋骨格ヒューマノイドが開発されてきたが、これらは詳細な人体模倣を目指しており、人体における劣駆動性や柔軟性等を目指したものである。同時に、上肢の可変剛性制御や筋張力制御等は実用的なレベルまで模索されてきたのに対して、下肢の動きに関しては、筋張力から計算したZMPによるバランス制御 [120] が行われている一方、上半身を支えて安定して歩行するための柔軟な身体の精密な制御が困難であるため、未だに二脚での歩行による移動は達成されていない。そのため、

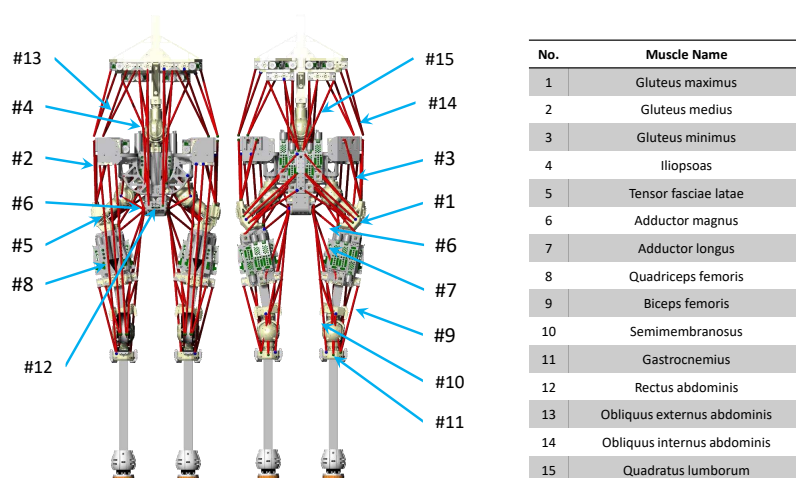


Fig. 4.30: The muscle arrangement of MusashiOLegs [118].

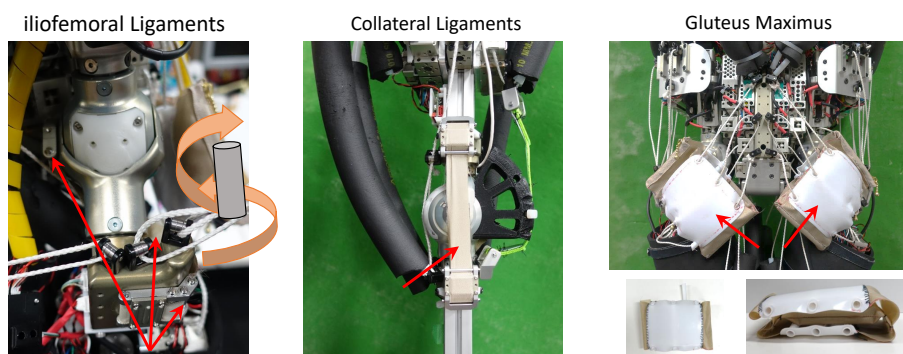


Fig. 4.31: The planar interskeletal structure of MusashiOLegs.

筋骨格ヒューマノイドは移動しながらのマニピュレーションについてはほとんど研究されることはなかった。

倒立二輪としては [160] や [161] が挙げられ、これまで盛んに研究されてきており、移動性や安定性に関して優れていると言える。同時に、台車型のロボットはより安定しており、姿勢安定化に力を要さないという利点がある。また、どちらもそのままでは基本的には倒れたら起き上がることはできないという問題がある。

車輪とマニピュレータを組み合わせたタイプのロボットとして代表的なのは PR2[149] や Fetch[150] 等のロボットであり、研究用として、多くの場所において活躍している。しかし、これらのロボットは足のフットプリントが大きいため狭隘な環境で作業することは難しく、一度倒れると自律的に立ち上がることができないため、足を台車ではなく倒立二輪としたロボットも多く開発されてきている。Boston

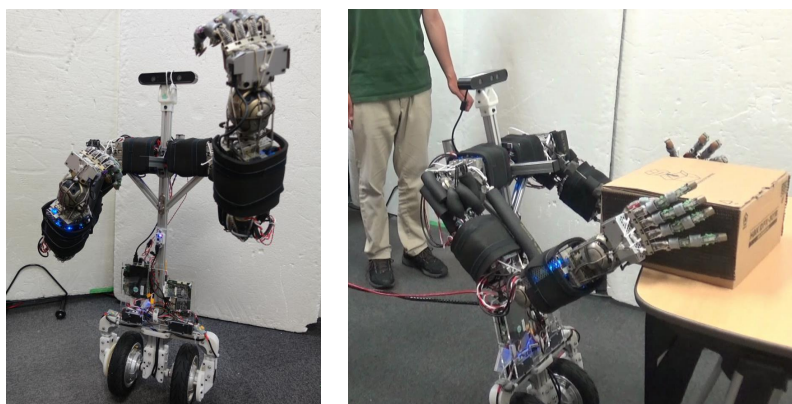


Fig. 4.32: Overview of TWIMP: Two-Wheel Inverted Musculoskeletal Pendulum.

Dynamics の Handle [162] は独立に可動する脚の先端に車輪を持ち、その自由度を用いて起伏のある地面でも常に両輪を接地させつつ安定した走行を行う。HITACHI の Emiew [163] は双腕と頭部を持つ人間に近い上半身と、移動性能に優れた倒立振子を組み合わせ、人間にとって親しみやすいナビゲーションロボットとして活躍した。しかし、どれもマニピュレータを環境と接触させながら動作することは少なく、倒立二輪の走行性能を重要視したロボットが多く開発されてきた。UBot-5 [164] は小さな体に二本の腕と倒立二輪を持ち、環境接触を伴いながらの学習型制御を行っているが、より日常生活等において実用的に用いることのできる大きさと、その大きな慣性による衝撃を受け止めることのできる柔らかな腕が必要である。

そこで、マニピュレータを環境と接触させ、また強い衝撃を加えるような動作に対しての利用を考え、それら性能に優れた筋骨格構造を持つ上肢と移動性能・フットプリントの小さな倒立二輪を組み合わせることでお互いの欠点を相殺し、利点をより強調することのできるロボットを提案した。また、学習等における研究用プラットフォームとしては、簡易に構成・再構成可能な設計が望ましく、その点でもその骨格のほとんどに汎用フレームを用いた本設計は有効である。

設計

全体の構成を Fig. 4.33 に示す。基本的には、筋骨格双腕 MusashiDarm と倒立二輪を汎用フレームにより合体させた身体となっており、自由にリンク長を変えることができる。車輪はトルクを重視することと、なるべく簡易な構成にすることで、再構成を容易にすべきである。そこで、モータと車輪が一体となったインホイールモータを使用しており、外側にアタッチメントをつけ、骨格は汎用フレームで構成することで非常に簡易な構成にすることに成功している。インホイールモータには外付けで

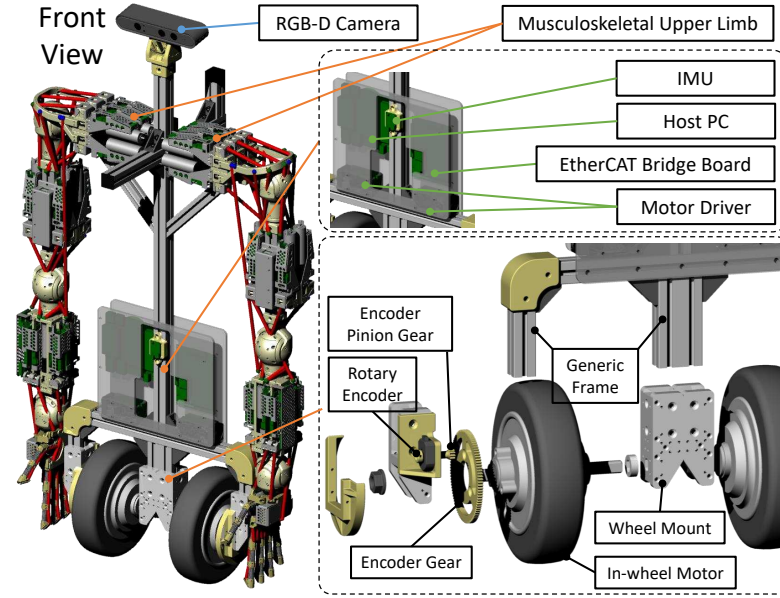


Fig. 4.33: Detailed Design of TWIMP [159].

エンコーダが、体幹の骨格には IMU が載っており、この二つを用いて倒立振子の姿勢・位置・角度に関する制御を行っている。頭には RGBD 視覚として Astra S (Orbbec 3D Technology International, Inc.) がついている。回路構成は Fig. 4.34 のようになっている。JAXON[8] と同様の大出力型ロボットの回路構成を足回りに対して適用し、筋骨格ヒューマノイド [87] における省スペースを重視した回路構成を組み合わせた構成となっており、Host PC (Intel NUC) や IMU 等は体幹部分に収納されている。転倒時に回路が破損することを防ぐため、上肢の胸部の前後にフレームを備え、転倒時に回路が直接地面と接触することを防いでいる。

倒立二輪制御

倒立二輪の制御は一般的な状態フィードバックによる方法を用いる。Fig. 4.35 のように、 θ を base の傾き、 ϕ を base からの車輪の回転、 m_w を車輪の質量、 m_b を base の質量、 I_w を車輪の慣性モーメント、 I_b を本体の慣性モーメント、 R を車輪の半径、 L を車輪と本体重心間の距離、 τ を車輪にかけるトルク、 $\mathbf{x} = [\theta \ \phi \ \dot{\theta} \ \dot{\phi}]^T$ を状態変数とすると、ラグランジュ方程式を解くことで状態方程式は以下

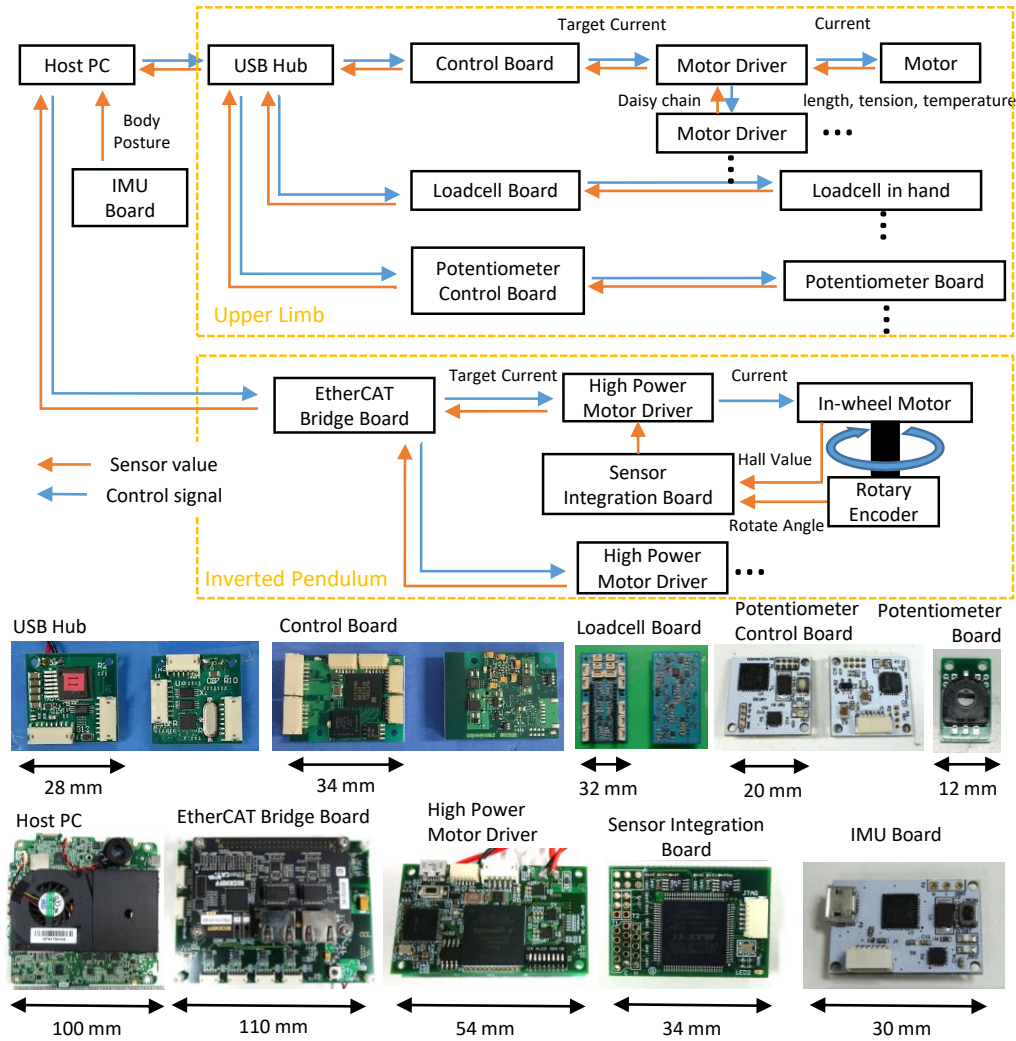


Fig. 4.34: Circuit configuration of TWIMP [159].

のようになる。

$$E\dot{\mathbf{x}} = A_0\mathbf{x} + B_0\tau \quad (4.4)$$

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu \quad (4.5)$$

$$A = E^{-1}A_0, B = E^{-1}B_0, u = \tau \quad (4.6)$$

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a+2b+c & a+b \\ 0 & 0 & a+b & a \end{bmatrix}, A_0 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (4.7)$$

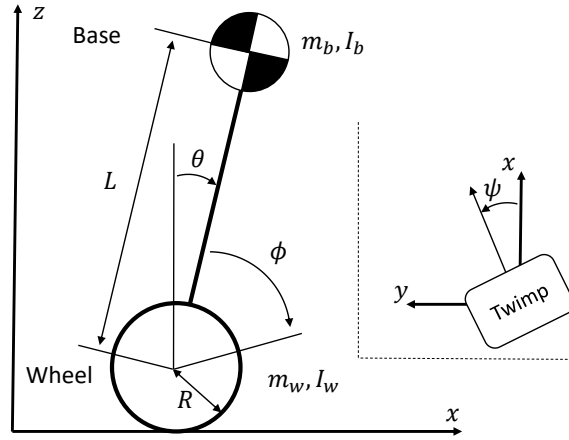


Fig. 4.35: Parameter configuration of two-wheel inverted pendulum [159].

$$\begin{cases} a = (m_b + m_w)r_w^2 + I_w \\ b = m_b r_w l \\ c = m_b l^2 + I_b \\ d = m_b g l \end{cases} \quad (4.8)$$

この状態方程式に対して $u = -Kx$ となるようなフィードバックゲインを最適レギュレータによってリカッチ方程式から求め、制御している。並進方向は ϕ を指令値との差分 $\phi^{ref} - \phi$ に変更することで制御し、回転方向 ψ に関しては θ, ϕ に対する大きな影響はないと考え、左右の車輪それぞれに逆転したトルクを、目標に対する差分に対して係数をかけた値を上乗せする P 制御を行っている。

$$u_l \leftarrow u_l - K_\psi(\psi^{ref} - \psi) \quad (4.9)$$

$$u_r \leftarrow u_r + K_\psi(\psi^{ref} - \psi) \quad (4.10)$$

ここで、 K_ψ は比例ゲイン、 $u_{\{l,r\}}$ は左右それぞれに対するトルク、 ψ^{ref} は回転方向に対する指令値である。状態 $\theta, \dot{\theta}, \phi, \dot{\phi}, \psi$ に関しては体幹についた IMU と車輪のエンコーダ値から求めている。また、姿勢の安定化が最も重要であるため、最適レギュレータの評価式 $J = \int (\mathbf{x}^T Q \mathbf{x} + u^T R u)$ において、 $Q = \begin{bmatrix} 500.0 & 1.0 & 500.0 & 0.2 \end{bmatrix}^T$, $R = \begin{bmatrix} 0.0001 \end{bmatrix}$ としており、 θ に対して ϕ の追従は非常に小さくなっている。

加えて、手が動作することにより、状態フィードバックのパラメータは大きく異なってくる。特に、手が動くことによる重心位置の変動は最も大きな問題であり、手を前に出した場合は姿勢を保つために体を反らせる必要が生じる。そのため、以下のように θ^{ref} を ϕ に応じて変化させて現在の重心状態

に適応していく必要がある。

$$\theta^{ref} \leftarrow \theta^{ref} + K_{adapt}\phi \quad (4.11)$$

ここで、 K_{adapt} は比例ゲインである。また、 ϕ に対しては不感帯を設けており、 θ^{ref} は大きく変更し過ぎると振動してしまうため、徐々に変更していくようになっている。

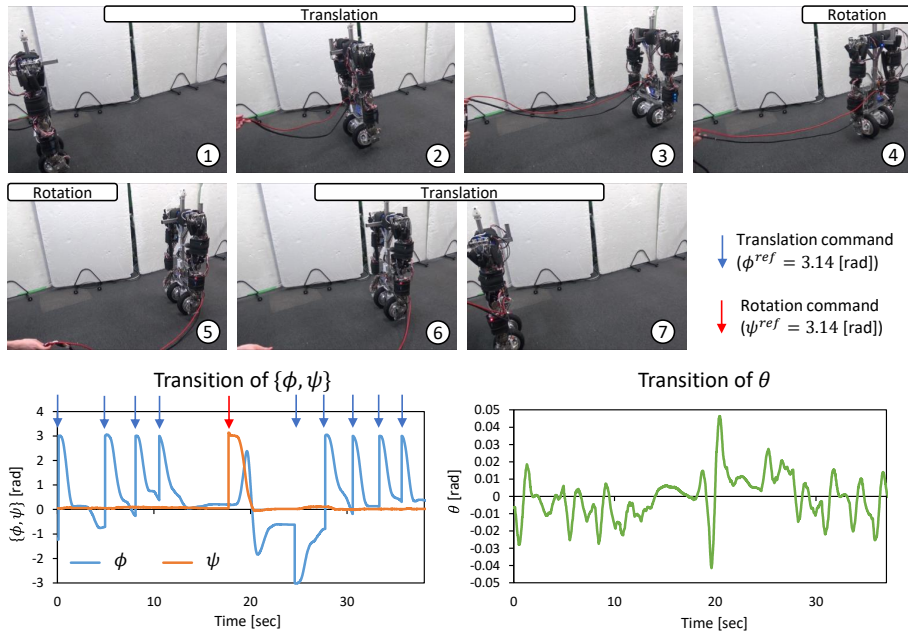


Fig. 4.36: The experiment of translational and rotational movements with the arms fixed to the initial pose [159].

倒立二輪に関する基礎実験

まず、腕を初期状態に固定して並進・回転に関する移動実験を試みる。初めに、 $\phi^{ref} = 3.14$ [rad] を何度か指令して並進し、 $\psi^{ref} = 3.14$ [rad] を指令して回転し、もう一度何度か $\phi^{ref} = 3.14$ [rad] を指令して戻る動作を行う。結果を Fig. 4.36 に示す。指令を送った直後は ϕ, ψ の値は 3.14 [rad] まで上昇し、その後、0 付近まで正しく移動することができている。また、姿勢角 θ はどの動作中も $-0.05 \sim 0.05$ rad に収まっていることがわかる。

次に、上肢を動作させた際に姿勢を安定に保てるかについて実験を行った。結果を Fig. 4.37 に示す。肩の pitch 軸を動かす際は倒立振子の重心位置が大きく変わるため、 ϕ が大きく移動している。その後、Eq. 4.11 によってその ϕ のずれから姿勢の指令値 θ^{ref} が変更され、オフセットは乗っているものの、 ϕ

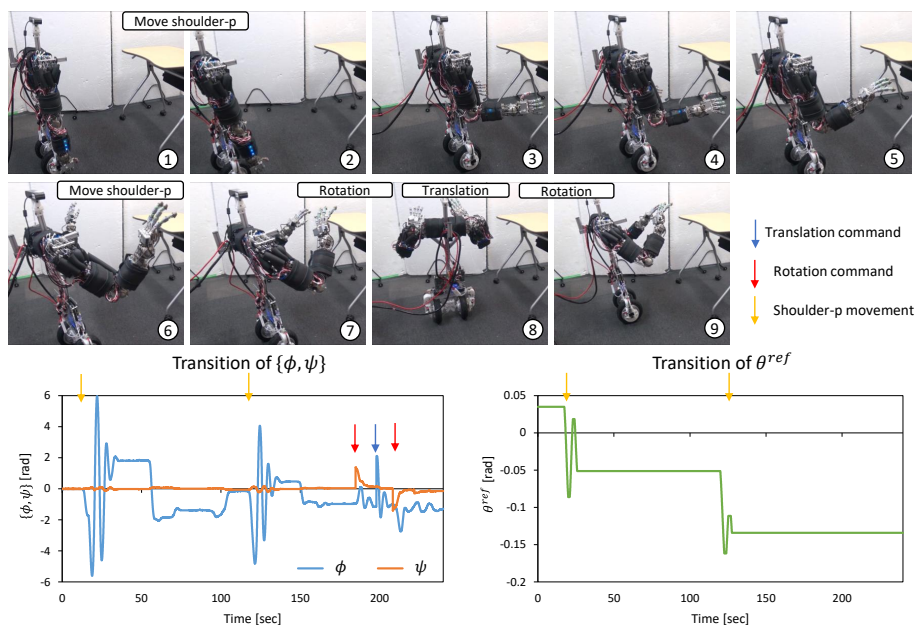


Fig. 4.37: Stabilization of TWIMP while moving its arms [159].

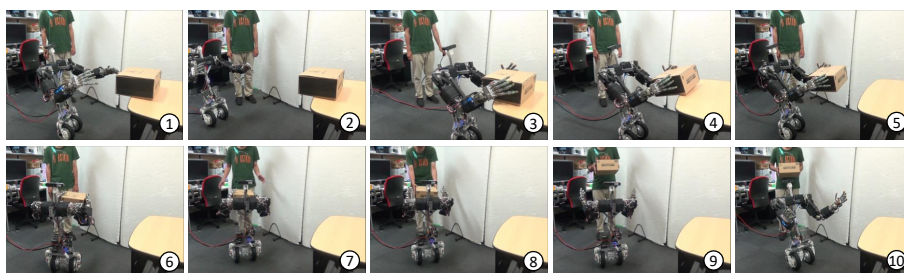


Fig. 4.38: The experiment of handing a box with both arms [159].

が正しく収束していることがわかる。また、腕を挙げた状態において回転、並進の指令を送った場合でも、 ϕ, ψ を正しく収束させ、安定的に動いていることがわかる。

最後に、ダンボール箱を双腕によって把持し、それを人に受け渡す実験を行う。まず机の上にあるダンボール箱を双腕によって把持し、回転して人の方向を向き、人が箱を持ったところで手を離す。結果を Fig. 4.38 に示す。筋骨格ヒューマノイド特有の柔らかな腕によってダンボールを横から押さえることによって把持し、 ϕ^{ref}, ψ^{ref} を指令することによって人の場所まで持って行くことに成功している。

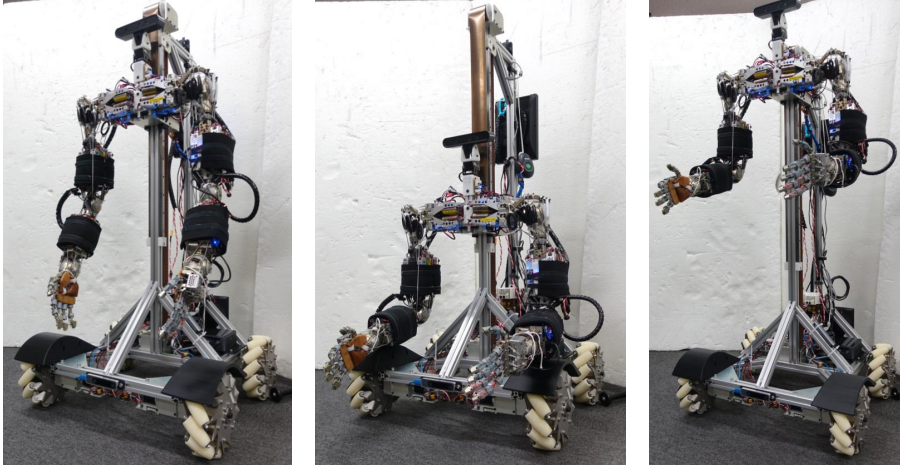


Fig. 4.39: The overview of the wheeled musculoskeletal humanoid Musashi-W.

4.6.5 Musashi-W

メカナム台車型筋骨格ヒューマノイド Musashi-W を Fig. 4.39 に示す. 上肢は筋骨格双腕 MusashiDarm, 下肢はメカナム台車であり, 台車に対して MusashiDarm が上下駆動する構造となっている. 身長は最大で 143 cm, 体重は 55.2 kg である. メカナム台車のアクチュエータは BLDC モータ $\phi 30$ 200W 14:1 を使用しており, カップリングを通して直径 203 mm のメカナムホイールが接続されている. MusashiDarm の頭には Dynamixel XM430-350 で構成された pan 軸と tilt 軸を通して RGBD カメラ Astra S (Orbbec 3D Technology International, Inc.) が接続されている. MusashiDarm 上下構造にはタイミングベルトとタイミングプーリを使用し, これをメカナム台車と同じアクチュエータで駆動している. MusashiDarm の重量を相殺するため, 荷重 160N の定荷重ばねを取り付けており, 静止時にはアクチュエータにほとんど負荷はかからない. なお, 回路構成は基本的に TWIMP と同様である.

メカナム台車制御

Fig. 4.40 にメカナム台車のパラメータを示す. y 方向のタイヤ間距離を $2a$, x 方向のタイヤ間距離を $2b$ とし, それぞれのタイヤ速度を $v_{\{1,2,3,4\}}$ とする. また, 台車の回転方向の角度を ψ とする. 車輪

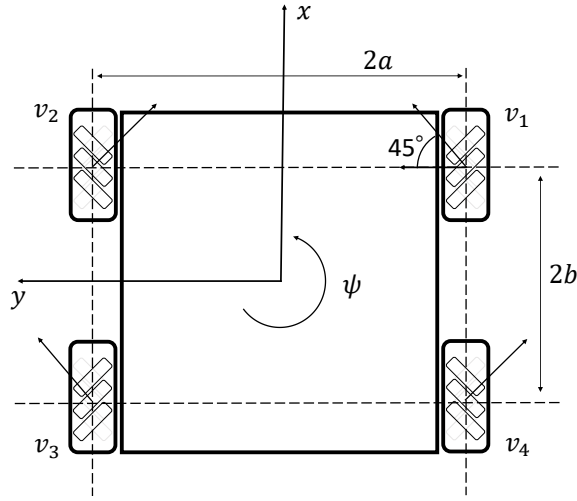


Fig. 4.40: Parameter configuration of the wheeled base of Musashi-W.

速度と台車の並進回転速度に以下の式が成り立つ.

$$\mathbf{v}_{wheel} = R\dot{\mathbf{x}} \quad (4.12)$$

$$\mathbf{v}_{wheel} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}, R = \begin{pmatrix} 1 & 1 & a+b \\ 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & a+b \end{pmatrix}, \dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{pmatrix} \quad (4.13)$$

よって, 指令速度 $\dot{\mathbf{x}}$ に対して, $R\dot{\mathbf{x}}$ を各車輪に速度指令することで, 所望の方向に動作することができる. また, オドメトリの際は $\mathbf{x} \leftarrow \mathbf{x} + R^+ \mathbf{v}_{wheel}$ を繰り返すことで, 現在の \mathbf{x} を推定することが可能である.

4.6.6 Kengoro

筋骨格ヒューマノイド Kengoro を Fig. 4.41 に示す [86]. Kengoro は Musashi の一世代前の筋骨格ヒューマノイドである. 詳細な人体模倣を目指して開発されており, 身長 167 cm, 体重 56.5 kg であり, 116 本の筋を持つ. Fig. 4.42 に Kengoro の上半身の筋配置を示す. 幾何モデルから分かるように, 鎖骨と肩甲骨を持ち, 上腕は開放型球関節により肩甲骨に接続している. これらの肩複合体に多数の多関節筋が配置され, 人間の主要な筋肉を模倣している. なお, Musashi のように関節角度を測定する機構・非線形弾性要素はない.

Kengoro に特徴的な構造を Fig. 4.43 に示す. まず, 特徴的な胸郭と背骨の構造を有している. 胸郭は CFRP により構成され, 背骨は主に切削バネによって構成されているため, 全体が外力により適応

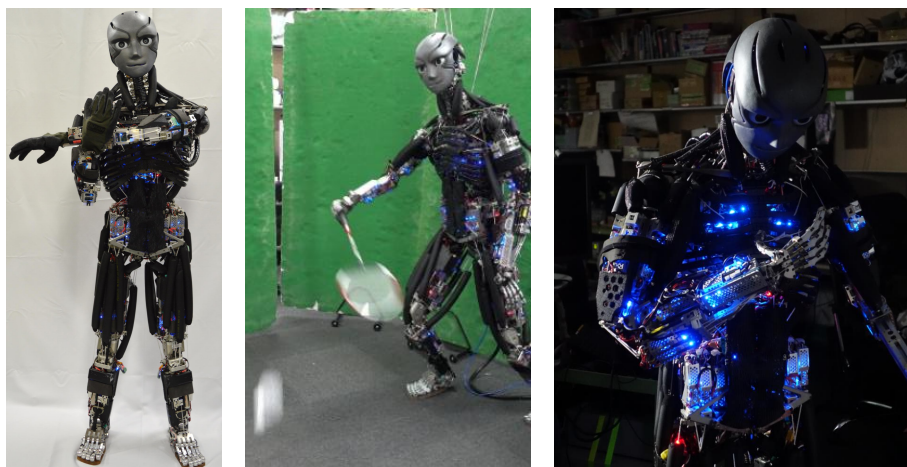


Fig. 4.41: The overview of the musculoskeletal humanoid Kengoro.

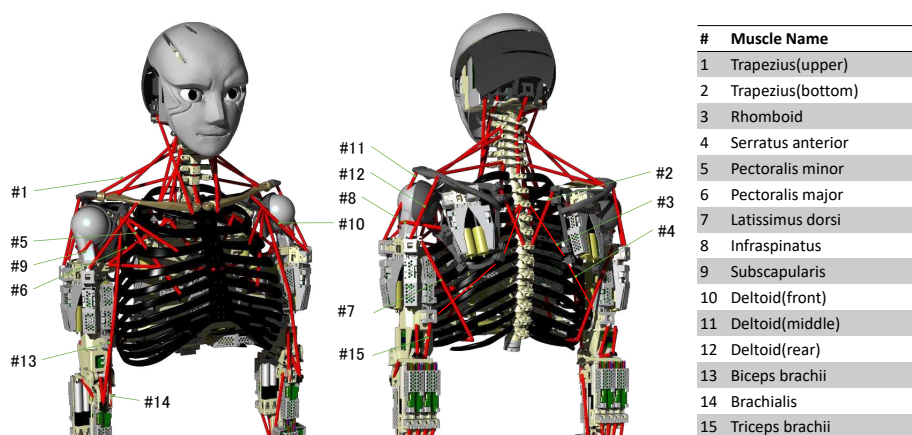


Fig. 4.42: The muscle arrangement of the left arm of Kengoro [86].

的に曲がる構造となっている。次に、上腕部に汗をかく構造を有している。上腕の骨は多孔質金属によって構成され、その周りにモータが接着されている。この多孔質金属内に水を充填することで、その気化熱によりモータを冷やすことが可能である。次に、前腕の橈骨尺骨構造を有している。橈骨と尺骨の2本の細長い骨によって構成された人間の前腕を模倣している。また、手は切削バネにより構成された力強く柔軟な指を持つ。最後に、5本の指により構成された足構造を持つ。リンクによって組み立てられた5本の足や足の付け根におけるバネにより、環境形状に柔軟に対応することができる。

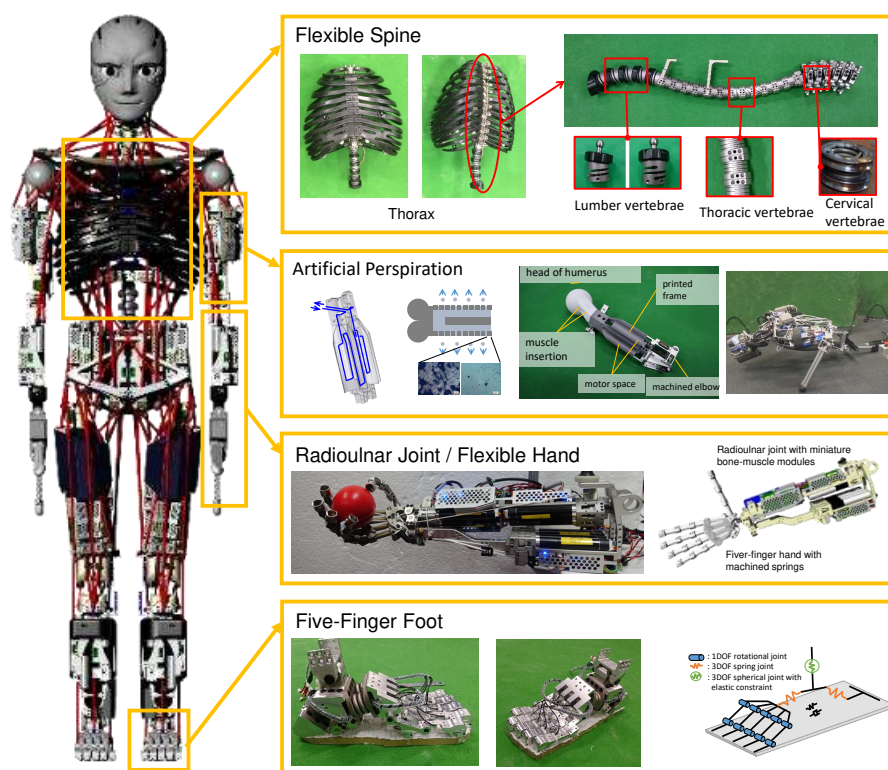


Fig. 4.43: The characteristic structures of Kengoro.

4.7 筋破断を補償する冗長性最大化に基づく筋配置最適化

本節では、筋破断を補償する冗長性最大化に基づく筋配置最適化手法 [165] について述べる。

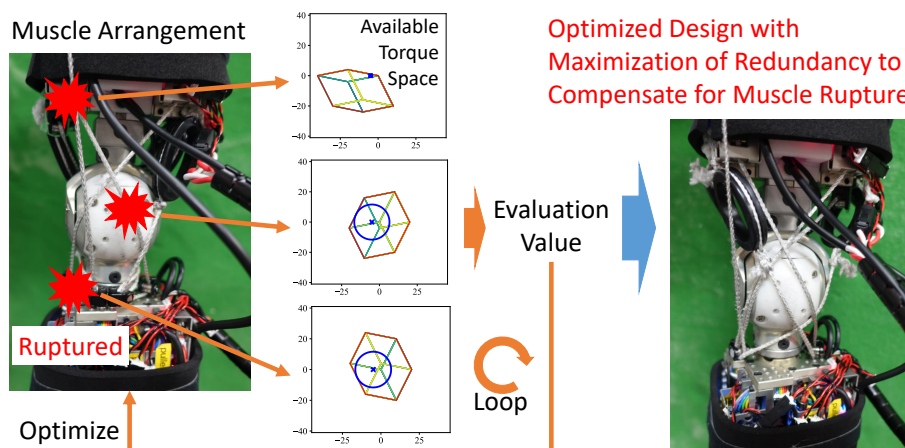


Fig. 4.44: The concept of this study [165]. By calculating the radius of the hypersphere (blue circle) inscribed in the available torque space (red polygon) when each muscle is broken, an evaluation value is calculated, and the design (muscle Jacobian) is optimized by a genetic algorithm so that the robot can continue to move even when one muscle is broken.

4.7.1 概要と先行研究

筋骨格ヒューマノイドは様々な生物規範型の特徴を持つが、冗長な筋配置は様々な利点・欠点を持つ特徴的な構造である。これまで、その利点を活かし、非線形性要素と冗長性を利用した可変剛性制御 [88, 125], 1 関節筋だけでなく 2 関節筋を用いたバランシングと関節協調 [166] 等が行われている。また、拮抗による高張力を削減する方法 [167, 168], 最も遅い筋により関節速度が制限される問題を解決する方法 [169] 等、その欠点を補う手法も様々開発されている。

本節では、この冗長な筋配置の一つの特徴である、筋が一本切れても冗長な筋肉を使って動き続けられるという利点に焦点を当てる。この特徴はこれまで多くは議論されていない。関連研究としては、関節トルク制御において切れた筋を使わずに筋張力を算出する方法 [123], 筋長制御においてオンライン学習により筋が切れても動き続ける方法 [125] が開発されている。人間においても、一部の筋が麻痺した場合における feasible force set の変化についての考察が行われている [170]。しかし、これまで筋が切れた時に、そもそも身体を思ったように動かせるかどうかを判断することや、その指標を元に設計を最適化すること等は行われていない。

そこで本節では, 筋破断を補償する冗長性を最大限に利用するための設計方針を最適化によって明らかにする. そのコンセプトは Fig. 4.44 に示す通りであり, 筋が切れたときにどう発揮可能関節トルク空間が変化するかどうかを評価し, 最適化のループを回す.

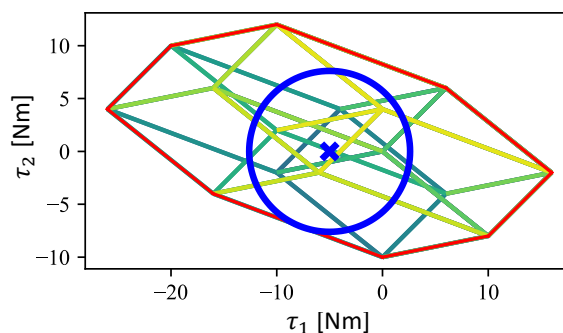


Fig. 4.45: Hypersphere (blue circle) inscribed in available torque space (red polygon). The line with a gradient from green to yellow is the line where each edge of the available muscle tension space is transformed into the joint torque space [165].

4.7.2 筋破断を補償する冗長性最大化に基づく筋配置最適化手法

筋骨格ヒューマノイド

筋骨格ヒューマノイドにも種類が存在する. アクチュエータについては, 電気モータとプーリによりワイヤを巻き取る方式 [171], 空気圧による人工筋肉を用いる方式 [172], Twisted and coiled polymer (TCP) を用いる方式 [173] 等が存在する. また, 筋のモーメントアームが一定であるモデル化が容易な方式, 人間のようにモーメントアームが変化する, より複雑でモデル化が困難な方式が存在する. 本節の実機で扱うロボットは, この中でも電気モータを用いており, 人間のようにモーメントアームが変化する複雑な筋骨格構造を持つ. 一方, モーメントアームが変化する場合はモデル化が困難なため, 数値的な解析が難しい. そのため, 本節では関節角度限界範囲の中心から大きく動かなければモーメントアームはある程度一定と捉えることができると考え, モーメントアーム一定として解析を行う.

RITS の計算

本節では冗長性の評価値を計算するために必要な値 (RITS) の導出を行う。筋骨格構造における基本的な式をもう一度以下に示す。

$$\mathbf{l} = \mathbf{h}(\boldsymbol{\theta}) \quad (4.14)$$

$$d\mathbf{l} = \mathbf{G}d\boldsymbol{\theta} \quad (4.15)$$

$$\boldsymbol{\tau} = -\mathbf{G}^T \mathbf{f} \quad (4.16)$$

ここで、 \mathbf{l} は筋長、 \mathbf{f} は筋張力、 $\boldsymbol{\theta}$ は関節角度、 $\boldsymbol{\tau}$ は関節トルク、 \mathbf{h} は関節角度から筋長へのマッピング、 \mathbf{G} は筋長ヤコビアンを表す。また、 $\{\mathbf{l}, \mathbf{f}\}$ は M 次元のベクトル (M は筋の数を表す)、 $\{\boldsymbol{\theta}, \boldsymbol{\tau}\}$ は N 次元のベクトル (N は関節の自由度数を表す) である。モーメントアームが一定でない場合、 \mathbf{G} は $\mathbf{G}(\boldsymbol{\theta})$ となり、本節のように一定と仮定する場合は定数行列となる。

ここで本節の目的は、発揮可能関節トルク空間に内接する超球の半径 (Radius of hypersphere Inscribed in available Torque Space, RITS) を求めることである。これは、例えばトルク空間を 2 次元とすると、Fig. 4.45 に示す青い円の半径と同等である。赤い線は発揮可能トルク空間の境界を表す。これは [174, 175] で用いられている MIV (Maximum Isotropic Value) と同様の考え方であるが、本節ではこれを関節トルクについて考える点や原点が発揮可能関節トルク空間内に存在するかどうかを判定する点等で異なる。また、発揮可能トルク空間の size や shape に関する議論は [176] に詳しい。RITS は全関節方向に対してどの程度力を発揮することができるかの指標となるため、筋が切れた状態でも、この値が 0 よりも大きければ身体を動かすことが可能となる。RITS r は数式で表すと以下のように計算される。

$$\text{maximize} \quad r \quad (4.17)$$

$$\text{subject to} \quad R \subseteq T \quad (4.18)$$

$$R := \{\boldsymbol{\tau} \in \mathbb{R}^N \mid |\boldsymbol{\tau} - \boldsymbol{\tau}_g| \leq r\} \quad (4.19)$$

$$T := \{\boldsymbol{\tau} \in \mathbb{R}^N \mid \exists \mathbf{f} \in F, \boldsymbol{\tau} = -\mathbf{G}^T \mathbf{f}\} \quad (4.20)$$

$$F := \{\mathbf{f} \in \mathbb{R}^M \mid \mathbf{f}^{\min} \leq \mathbf{f} \leq \mathbf{f}^{\max}\} \quad (4.21)$$

ここで、 $\mathbf{f}^{\{\min, \max\}}$ は筋張力の最小・最大値、 $\boldsymbol{\tau}_g$ は重力や摩擦等に抗って出さなければならない関節トルクを表す。本節では、この $\boldsymbol{\tau}_g$ は一定とみなしている。

この r を求める方法を以下に示す。まず、Eq. 4.21 において、この \mathbf{f} は M 次元の超立方体 F と捉えることができる。そして、この \mathbf{f} が Eq. 4.16 によって、 N 次元の $\boldsymbol{\tau}$ の超多面体 T へと変換されると考

えることができる. ここで, 凸な超多面体を一次変換により超多面体に変換してもこれは凸である. そのため, この r は, この N 次元超多面体内に内接する超球の半径と考えることができる. その計算アルゴリズムは Alg. 1 のように書ける.

Algorithm 1 Calculation of RITS r [165]

```

1: function CALCRITS( $G, \tau_g$ )
2:    $included = \text{True}$ 
3:    $p = []$ 
4:   for  $v_F$  in all vertices of  $F$  do
5:     push  $-G^T v_F$  to  $p$ 
6:   end for
7:    $C = \text{CalcConvexHull}(p)$ 
8:    $c = \text{CalcCenter}(C)$ 
9:    $r = 1e9$ 
10:  for  $s_C$  in all simplices of  $C$  do
11:     $d_1 = \text{CalcDistanceWithSign}(\tau_g, s_C)$ 
12:     $d_2 = \text{CalcDistanceWithSign}(c, s_C)$ 
13:    if  $d_1 d_2 < 0$  then
14:       $included = \text{False}$ 
15:    end if
16:     $r = \min(|d_1|, r)$ 
17:  end for
18:  if not  $included$  then
19:     $r = 0$ 
20:  end if
21:  return  $r$ 
22: end function

```

まず, f が張る超立方体 F の全頂点を取り出し, それを τ の空間 T へと射影する. そして, その射影された全長点の凸包 C を計算する (CalcConvexHull). この C 内に頂点 τ_g が含まれなければ内接超球は存在しないため, これを判定する必要がある. まず, 絶対に C 内に存在する, C に含まれる全頂点の平均である中心点 c を求める (CalcCenter). 次に, C 内の全超平面について, τ_g からの距離と c からの距離を符号付きで求める (CalcDistanceWithSign). これは, 超平面の法線 a を求め, $(a^T \tau_g - 1)/\|a\|$ を計算することを意味する. この二つの距離 d_1, d_2 の掛け算が正であるとき, この c と τ_g は超平面に対して同じ側に存在する. これを全超平面に対して行い, 全て c と同じ側にあった場合, C 内に τ_g が存在することになる. 存在した場合, τ_g から C によって得られた超平面への距離 $|d_1|$ を求め, この最小値が内接する超球の半径となる.

冗長性の評価

与えられた設計 (G) について, RITS を用いて筋が切れても動き続けられるかどうかを表す指標を計算する. G の i 行目, つまり i 番目の筋が切れ, そのモーメントアームが全て 0 となったものを G_i とする. 本節では以下の値 E を評価値として用いる.

$$r_0 = \text{CalcRITS}(G, \tau_g) \quad (4.22)$$

$$r_i = \text{CalcRITS}(G_i, \tau_g) \quad (1 \leq i \leq M) \quad (4.23)$$

$$E_{\text{value}} = \sum_{i=0}^M r_i \quad (4.24)$$

$$E_{\text{count}} = \sum_{i=0}^M \text{Integer}(r_i > 0) \quad (4.25)$$

$$E = \begin{cases} E_{\text{value}} & (E_{\text{count}} \geq M_{\min} + 1) \\ 0 & (\text{otherwise}) \end{cases} \quad (4.26)$$

ここで, M_{\min} は切れても問題のない筋の数の下限を表し (筋は一回に一本のみが切れることを想定している), $\text{Integer}(x)$ は x が真であれば 1, 偽であれば 0 を返す関数である. つまり $E_{\text{count}} - 1$ は切れても問題のない筋の個数を表し, これが指定した M_{\min} より多ければ, G とそれぞれの筋を切ったときの G_i に関する RITS の合計値が E となる.

設計最適化と具体的実装

得られた評価値 E をもとに, 筋が一本切れても残りの筋により関節を動かすことができるよう, 設計 (G) を最適化する. これには様々な方法が考えられるが, 本節では遺伝的アルゴリズムを用いた. 本節では具体的な数値も含め実験セットアップを説明する. ライブラリとして [177] を用い, 50%の確率で関数 `cxBlend` により交叉, 20%の確率で関数 `mutGaussian` により突然変異を行う. 個体選択は関数 `selTournament` で行い, トーナメントサイズを 3 とする. 個体数は 200, 世代数は 50 とする. また, 筋長やコピアンについてはそれぞれの値の最小値と最大値を -0.1, 0.1 [m] とし, 交叉や突然変異の度に範囲内に収まるように `crop` を行う. また, わかりやすいよう小数点以下は 2 桁までに制限して計算を行う. 筋張力については $F^{\min} = 0$ [N], $F^{\max} = 200$ [N] とした. 本節では, 関節の自由度数 N , 筋数 M , 切断可能な筋数の下限 M_{\min} , 重力や摩擦等に抗って出すべきトルク τ_g を変化させながら結果を考察する.

Table 4.1: The optimized designs of 1-DOF tendon robot. The values show $10G^T$ [165].

	$\tau_g = -5$	$\tau_g = 0$
$M = 3$	$\begin{pmatrix} -1 & 1 & 1 \end{pmatrix}$	No Solution
$M = 4$	$\begin{pmatrix} -1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & 1 & 1 \end{pmatrix}$
$M = 5$	$\begin{pmatrix} -1 & -1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & 1 & 1 & 1 \end{pmatrix}$
$M = 6$	$\begin{pmatrix} -1 & -1 & -1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & -1 & 1 & 1 & 1 \end{pmatrix}$

4.7.3 実験

1自由度シミュレーション

$N = 1$ のロボットについてシミュレーションを行う。パラメータを $M = \{3, 4, 5, 6\}$, $\tau_g = \{0, -5\}$ に変化させたときに最適化された結果を Table 4.1 に示す。なお、見やすさのため、10 倍した $10G^T$ を表示している。 M_{min} は変化させても $\tau_g = 0, M = 3$ の時以外は同じ結果であったため、 $M_{min} = M$ と一定としている。

まず、 $\tau_g = 0$ の場合については、 $M = 3$ では解が得られていない。これは、必ず正か負のモーメントアームの筋が1本だけになってしまうため、それが破断してしまうと片側に対してトルクが発生できなくなってしまうためである。また、 $M \geq 4$ のときは、 M が偶数であれば正と負のモーメントアームの数が一致している設計が最適であることがわかる。

これに対して、 $\tau_g = -5$ のように一定のトルクがかかる（例えば肘を-90度に曲げた）状態では、 $M = 3$ のときも解が出ている。これは、もし負のモーメントアームの筋が切れたとしても τ_g によってトルクを確保できるためである。また、 $M = 4$ のときについてはわかりやすいように Fig. 4.46 に $\tau_g = \{-5, 0\}$ のときの最適設計を示す。 $\tau_g = -5$ の場合には正のモーメントアームの筋が3本、負が1本であり、 $\tau_g = 0$ のときは両者とも2本であった。これは、 $\tau_g = -5$ の場合は τ_g により正のトルクが稼げるため、負のモーメントアームより正のモーメントアームの筋を多くしたほうがより発揮可能トルク空間が稼げるということである。正のモーメントアームの筋を切ると、 $M = 3$ の場合の最適設計と同じになる点も興味深い。

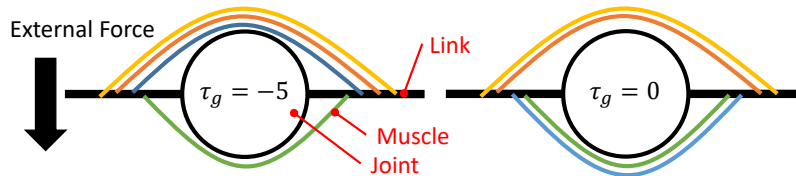
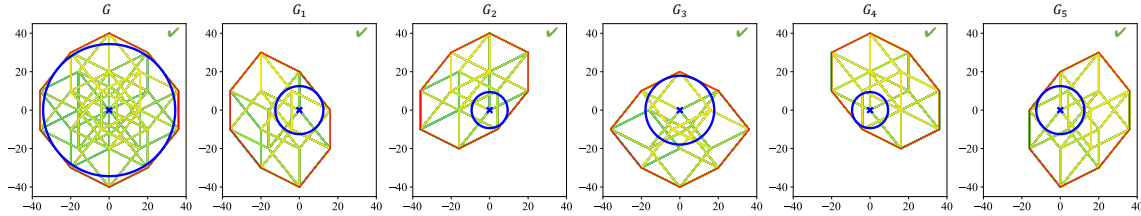
Fig. 4.46: The optimized design when $M = 4$ and $\tau_g = \{-5, 0\}$ [165].

Table 4.2: The optimized design of 2-DOF tendon robot [165]. The values show $10G^T$.

	$\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$	$\tau_g^T = \begin{pmatrix} 0 & 0 \end{pmatrix}$
$M = 4$ ($M_{min} = 4$)	$\begin{pmatrix} -0.5 & -0.5 & 1 & 1 \\ -1 & 1 & -0.2 & 0.2 \end{pmatrix}$	No Solution
$M = 4$ ($M_{min} = 3$)	$\begin{pmatrix} -1 & 0.7 & 0.8 & 0.8 \\ 0 & 1 & -1 & 1 \end{pmatrix}$	No Solution
$M = 5$ ($M_{min} = 5$)	$\begin{pmatrix} -1 & 0.2 & 0.6 & 1 & 1 \\ -0.4 & 1 & 1 & -1 & -0.9 \end{pmatrix}$	$\begin{pmatrix} -1 & -0.8 & 0 & 0.8 & 1 \\ -0.5 & 1 & -1 & 1 & 0.5 \end{pmatrix}$
$M = 5$ ($M_{min} = 4$)	$\begin{pmatrix} -1 & -1 & 0.7 & 0.8 & 0.9 \\ 0 & 0.1 & -1 & 1 & -1 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 & 0.5 & 1 & 1 \\ -0.4 & 0.4 & -1 & -1 & 1 \end{pmatrix}$

Fig. 4.47: The available torque space and RITS of the optimized design when $\tau_g^T = \begin{pmatrix} 0 & 0 \end{pmatrix}$, $M = 5$, and $M_{min} = 5$ [165].

2 自由度シミュレーション

$N = 2$ のロボットについてシミュレーションを行う。パラメータを $M = \{4, 5\}$, $\tau_g = \left\{ \begin{pmatrix} -5 & 0 \end{pmatrix}^T, \begin{pmatrix} 0 & 0 \end{pmatrix}^T \right\}$, $M_{min} = \{M, M - 1\}$ に変化させたときに最適化された結果を Table 4.2 に示す。なお、同様に見やすさのため、10 倍した $10G^T$ を表示している。

まず、 $\tau_g^T = \begin{pmatrix} 0 & 0 \end{pmatrix}$ の場合は、 $M = 4$ のときは $M_{min} = \{4, 3\}$ について解が見つからない。そのため、2 自由度動かし、かつどの筋が一本切れても問題ない状態にするためには、必ず 5 本以上の筋が必要となる。そして、 $M = 5$, $M_{min} = 5$ のときの最適設計の発揮可能トルク空間と RITS, 筋を 1 本ずつ切ったときのそれらの変化を Fig. 4.47 に示す。最適設計では原点 τ_g を中心に大きな円が描かれていることがわかる。また、一本ずつ筋を切った際の τ_g は発揮可能トルク空間内に内包され、関節を任意の方向に動かすことができることがわかる。

次に、 $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$ の場合は、 $M = 4$ についても解が得られている。1 自由度のときとは違い、全てのモーメントアームが 1 か -1 ではなく、-0.5 や 0.2 等の値も含まれていることにも着目したい。 $M = 4$, $M_{min} = 4$ のときの最適設計 (A) の発揮可能トルク空間と RITS, 筋を 1 本ずつ切ったときのそ

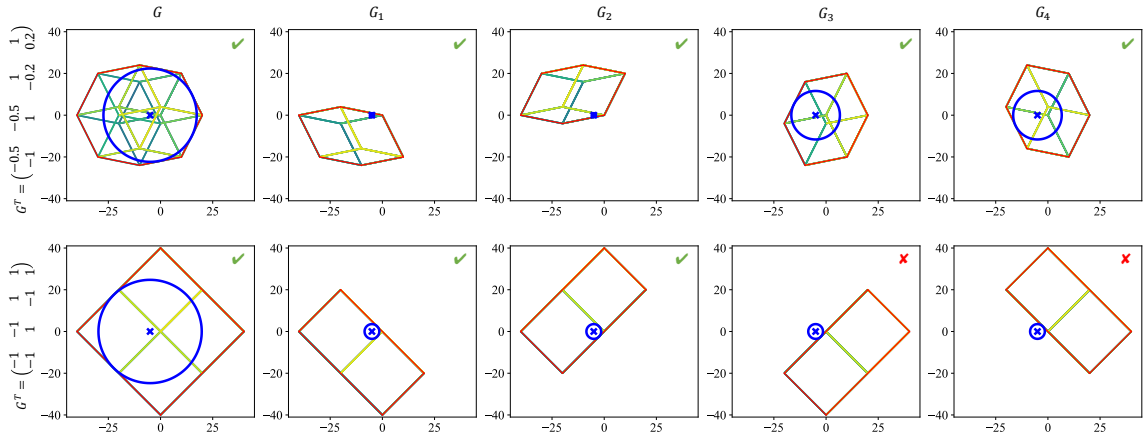


Fig. 4.48: The available torque space and RITS of the optimized design when $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$, $M = 4$, and $M_{min} = 4$, and when absolute value of each moment arm of A is maximized with the same sign [165].

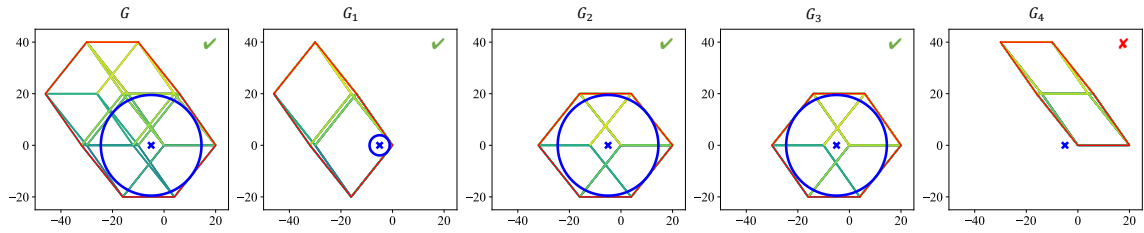


Fig. 4.49: The available torque space and RITS of the optimized design when $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$, $M = 4$, and $M_{min} = 3$ [165].

これらの変化を Fig. 4.48 の上段に示す。また、-0.5 や 0.2 を -1, 1 のように、符号をそのままに絶対値を最大にした場合の設計 (A') についても Fig. 4.48 の下段に示す。なお、それぞれの図の右上にあるチェックマークは、発揮可能トルク空間内に τ_g が存在するかどうかを表している。A の設計では、どの筋が切れても τ_g が発揮可能トルク空間内に存在し、動き続けることができる。なお、筋 1 と 2 が切れた際は RITS が非常に小さいため、これを解消するためには、最適化の際に RITS の大きさにも制約をかけることが考えられる。しかし、通常考えられる、最大のモーメントアームをどの関節に対しても均等に配置する設計 A' では、筋 3 と 4 が切れた際には任意の方向にトルクが出せなくなってしまうことがわかる。つまり、モーメントアームは全て最大に設定するのではなく、場所に応じて小さな値を設定する必要があることがわかる。

最後に $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$, $M = 4$, $M_{min} = 3$ のときの結果を Fig. 4.49 に示す。これに限らないが、 $M_{min} = M - 1$ のときは、最適化された設計の発揮可能トルク空間がある一方向に伸びた形になっていることがわかる。そして、2 つの筋 2 と 4 については、筋を切っても RITS が変化しない。また、

$M_{min} = M - 1$ であるため, ある一つの筋については切れると完全に動かせなくなってしまうということもわかる.

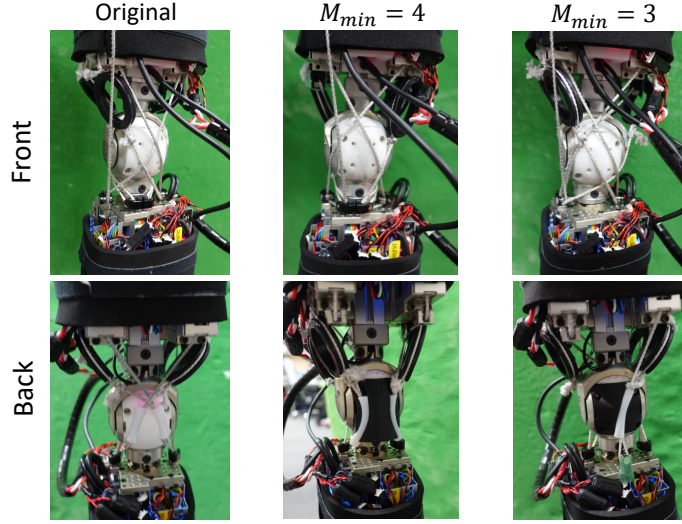


Fig. 4.50: The original design and optimized designs with $M_{min} = \{4, 3\}$ of the elbow of Musashi [165].

Table 4.3: The human-made designs of the elbow of Musashi with reference to the optimal designs obtained in simulation [165]. The values show $10G^T$.

original	$\begin{pmatrix} -0.47 & -0.26 & 0.43 & 0.42 \\ -0.17 & 0.20 & -0.15 & 0.24 \end{pmatrix}$
$M = 4, M_{min} = 4$	$\begin{pmatrix} -0.18 & -0.29 & 0.61 & 0.43 \\ -0.10 & 0.20 & -0.043 & 0.036 \end{pmatrix}$
$M = 4, M_{min} = 3$	$\begin{pmatrix} -0.47 & 0.50 & 0.66 & 0.73 \\ 0.047 & -0.19 & -0.30 & 0.18 \end{pmatrix}$

筋骨格ヒューマノイド

$N = 2$ のシミュレーションについて, Table 4.2 の $M = 4$, $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$ の状態を考える. 筋骨格ヒューマノイド Musashi の左肘には 4 つの筋と elbow-pitch, elbow-yaw の 2 つの関節が存在する. そのため, 肘を -60 度程度曲げたとき (θ_0) の状態は, 大体 $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$ の力が重力補償に必要であり, 前節の最適化の条件と似た状態になっている. そこで, シミュレーションから得られた G を参考に, 肘周りの筋経路を修正し, 実機においてどのような性能差が生まれるかについて検証する. 筋骨格ヒューマノイド Musashi は筋経路がプーリー等により正確に設計できるわけではなく, 人間と同じように球関節に巻きつく形であるため, 人間の設計・解釈性に多少依存することに留意されたい.

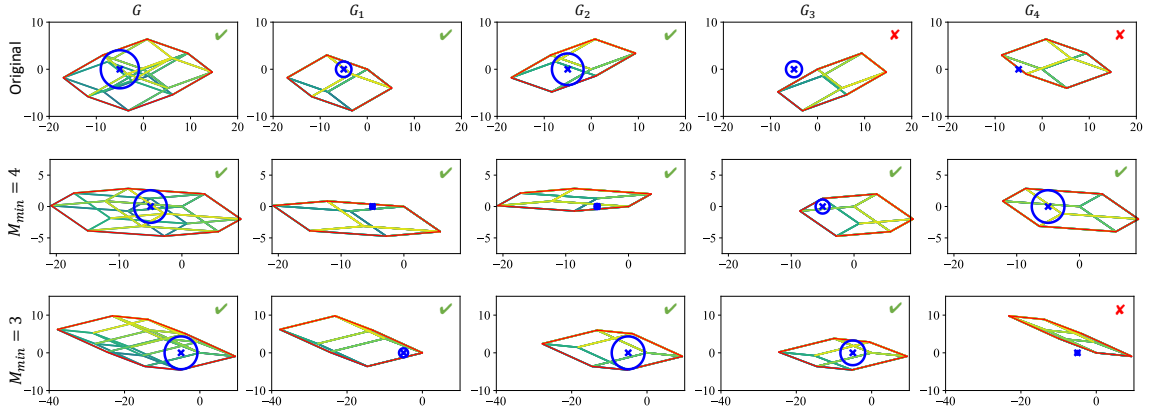


Fig. 4.51: The available torque space and RITS of the original design of Musashi and the designs based on the optimization results of $M_{min} = \{4, 3\}$ in Table 4.2 [165].

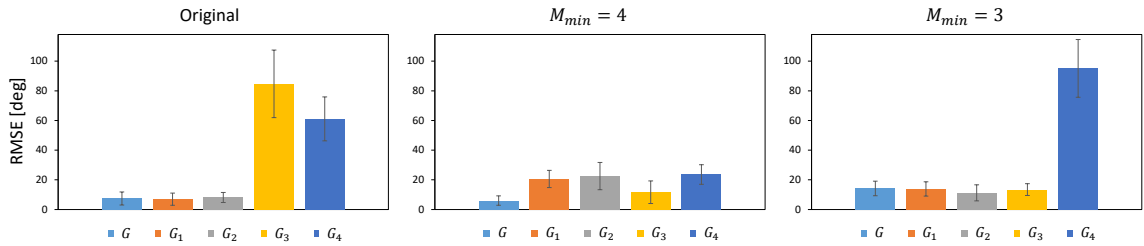


Fig. 4.52: The average and standard deviation of RMSE between the commanded and measured joint angles when using the original design and the optimized designs with $M_{min} = \{4, 3\}$ [165].

設計した筋経路を Fig. 4.50 に示す。左から、これまで使われてきた筋経路 [87], $M_{min} = 4$ の最適設計解, $M_{min} = 3$ の最適設計解を参考にしたものである。なお、この Original は Fig. 4.48 の下段の設計に似通っている。しかし、図からではモーメントアームはわからないため、[178] の手法を用いて関節・筋張力・筋長の関係を学習し、得られた θ_0 における筋長ヤコビアンを Table 4.3 に示す。なお、見やすさのため同様に $10G^T$ を表示している。Original は elbow-p, elbow-y についてそれぞれの筋が同じようなモーメントアームを有しているのに対して、 $M_{min} = 4$ の設計解を参考にした筋配置では一部のモーメントアームを小さく設定することで筋張力の干渉を減らしている。また、 $M_{min} = 3$ の設計解を参考にした筋配置では、elbow-p に対する正負のモーメントアームを持つ筋を 2 つずつ配置するのではなく、正のモーメントアームを持つ筋を 3 つにして偏りを持たせている。このときのモーメントアーム G から得られる発揮可能トルク空間と RITS, 筋を 1 本ずつ切ったときのそれらの変化を Fig. 4.51 に示す。elbow-yaw (図の y 軸) に関するモーメントアームが設計の都合上得にくいため、Fig. 4.48 や Fig. 4.49 よりも縦方向が縮んでいる。Fig. 4.51 の上段と Fig. 4.48 の下段、Fig. 4.51 の中段と Fig. 4.48

の上段, Fig. 4.51 の下段と Fig. 4.49 が対応しており, それらの発揮可能トルク空間の形, RITS, 任意方向に力を発揮可能かどうか似通っていることが読み取れる.

これら得られた筋配置の状態で, θ_0 周辺の elbow-p, elbow-y の関節角度を 6 つ決め, 実機に [178] を用いて制御入力として指令することを 3 回繰り返す. それぞれの設計について, 筋が一本も切れていない状態と, それぞれの筋が一本ずつ切れた状態 (電流を 0 とした) についてこの実験を行い, 指令関節角度と測定された関節角度の RMSE (Root Mean Squared Error) の平均と分散を計算する. その結果を Fig. 4.52 に示す. Original では 3 本目, 4 本目を切った際に RMSE が大きく上昇し上手く身体を動かしておらず, Fig. 4.51 の上段と同じ結果となった. $M_{min} = 4$ で最適化された設計については, 筋を切ることで少し RMSE は上昇するものの, どの筋を切っても身体の運動を維持することに成功した. $M_{min} = 3$ で最適化された設計については, 4 本目の筋を切った際は RMSE が大きく上昇し上手く身体を動かしておらず, Fig. 4.51 の下段と同じ結果となった. ゆえに, 実機においても本手法が有効であることがわかった.

4.7.4 議論

本節の実験から得られた結果について考察する. まず, 1-DOF のシミュレーションに関する実験から, 1 自由度ロボットで $\tau_g = 0$ の場合に冗長性の効果を最大化するには, 正と負のモーメントアームの筋を出来る限り同じ本数にするべきであることがわかった. この際, モーメントアームは全ての筋について設定した最大値である方が良い. また, $\tau_g = -5$ のように一方方向に余分に力をかける必要がある状態においては, その方向に筋を多めに配置することで, どの筋が一本切れても RITS を大きく取ることができる.

次に, 2-DOF のシミュレーションに関する実験から, 2 自由度ロボットで $\tau_g^T = \begin{pmatrix} 0 & 0 \end{pmatrix}$ の場合には, どの筋が切れても任意方向に関節を動かせるようにするためには, 5 本以上の筋が必要であることがわかった. $\tau_g^T = \begin{pmatrix} -5 & 0 \end{pmatrix}$ のように一歩方向に余分に力をかける必要がある状態においては, 4 本の筋でも問題ないが, 一部の筋が切れた際の RITS が非常に小さくなってしまう場合がある. また, 1 自由度の場合と違い, 全モーメントアームの絶対値を同じように最大にしてしまうと一部の筋が切れた時に釣り合いが保てず関節が動かせなくなってしまう. そのため, 一部のモーメントアームを半分やそれ以下に小さくすることが冗長性を活かすためには重要となる. また, $M_{min} = M - 1$ のように制約を緩和すると, その一本に大きな役割を担わせ, 残りの筋は切れても RITS を大きく保てるような設計が導出される. これは逆に, その重要な一本さえ守れば良いため, この筋を積極的に利用した冗長性の利用も考えられる.

最後に, Musashi の実機に関する実験から, シミュレーションで得られた最適設計解を模倣することで, 実機においても同様の性能が得られることがわかった. 実機においてはモデル化が難しい摩擦の影響等があるため, RITS が小さい場合は指令関節角度と実現された関節角度の差が多少大きくなる場合があるが, 概ね特性はシミュレーションと一致していた. よって, シミュレーションで得られた結果を元に設計を変更することで, 筋が一本切れても動き続けるロバストな身体を作ることができる考える.

本手法の限界について述べる. まず, 本手法は関節が3自由度以上の一般の系に対しても適用可能であるが, 評価値 E の計算量は RITS の計算における超立方体の全頂点列挙の計算量 $O(2^M)$ に依存するため, 筋数に従って指数関数的に爆発する. これは, face-span 変換を利用しても同じである. 関節が増えるに従って筋数も増やす必要があるため, 現状 $N = 3, M = 7$ 程度までが限界であり, 今後高速化の必要がある. また, 本手法はモーメントアームを一定と近似したうえで最適化を行い, モーメントアームが一定でない人間のような関節の設計にそれを応用している. そのため, N が増えるほど人間の解釈に依るところが大きくなってしまうため, 本手法では $N = 2$ に留めている. モーメントアームが θ によって徐々に変化する系に対しても一般に適用可能な方法にしていく必要がある.

4.8 本章のまとめ

本章では軸駆動型ロボット PR2, 台車型ロボット Fetch, 低剛性樹脂製ロボット KXR, 筋骨格ヒューマノイドについて, 柔軟性と冗長性に焦点を当てながらそのハードウェア構成を述べた. 既存の軸駆動型ロボットや台車型ロボットにも時間的柔軟性や空間的柔軟性, 制御的冗長性や感覚的冗長性があり, 身体-道具-環境間の相関複雑性と時間的変容に対応する必要がある.

また, 構成法が確立されていない筋骨格ヒューマノイドについては, その柔軟性と冗長性を活かすため, そして学習制御のプラットフォームとして利用するための設計要件について考えた. 設計コンポーネントとして, 筋モジュール・関節モジュール・筋経路点ユニット・非線形弾性ユニットを開発した. 筋モジュールは時間的柔軟性と制御的冗長性を, 関節モジュールは時間的柔軟性と感覚的冗長性を, 筋経路点ユニットは時間的柔軟性を, 非線形弾性要素は空間的柔軟性を担保することができる. また, 多数の冗長な接触センサと可変剛性機能を有する柔軟ハンド・輻輳による物体距離認識機能と高解像度カメラを有した可動眼球ユニット・足全周の触覚を持ったコア-シェル型足ユニット・人体模倣構造により三次元音源定位が可能な両耳ユニット・臀部と環境の柔軟な接触力を測定可能な臀部触覚センサの設計開発についても述べた. これらの組み合わせにより, 筋骨格ヒューマノイド MusashiLarm, Musashi, MusashiOLegs, TWIMP, Musashi-W, Kengoro が開発された.

最後に, 筋骨格ヒューマノイドの冗長な筋配置の特徴である, 筋が破断しても身体を動かし続けられるという利点を最大限に活かすための解析手法・身体設計最適化について提案した. 筋が一本ずつ切れた際に発揮可能関節トルク空間が張る超多面体に内接する超球の半径を最大化するという問題を設定し, その手法について述べた. シミュレーション実験から, 全モーメントアームを同じように配置するのではなく, 一部のモーメントアームを小さくすることで, どの筋が切れても動き続けられるような設計が可能となる. また, 最適化の評価関数次第では, 一本に負荷を集中させ, その他の筋はどれが切れても動ける, というような偏った設計にすることも可能であった. 本手法を実機に適用した結果, シミュレーションと同様の結果が示され, その有効性が確認された.

第5章

身体図式学習を支える反射制御

本章の概要を Fig. 5.1 に示す. 本章ではまず, 5.1 節において, 反射制御の要件として, 空間的柔軟性と制御的冗長性の欠点であるモデル化誤差や拮抗関係による内力上昇・温度上昇・速度制限等に短い周期で対応する必要について述べる. 次に, 5.2 節において, 温度推定パラメータのオンライン学習, 動的な最大出力制限, パラメータ変化に基づく異常検知を行い, 温度上昇に対応する手法について述べる. 次に, 5.3 節において, 姿勢に影響のない範囲で不必要な拮抗筋から徐々に弛緩させることで, 内力緩和動作, 環境を使い身体を休める動作が可能な筋弛緩制御について述べる. 次に, 5.4 節では, 筋骨格系に特有な筋の拮抗関係における, 冗長な多数の筋の中で最も速度の遅い筋に関節速度が制限されてしまう問題を解決する, 拮抗筋抑制と拮抗筋予見伸長反射について述べる. 次に, 5.5 節において, モデル化誤差による拮抗筋間の内力上昇により身体の可動範囲が制限されてしまう問題を解決可能な, 人間の相反性神経支配に基づく拮抗筋抑制制御について述べる. 最後に, 5.6 節において, 危険回避, 姿勢保持, 能動的活用等の応用が可能な, 人間の伸長反射のロボットにおける実装とその有効性について述べる. これらのうち, 5.2 節は全てのロボットに適用可能であり, 5.3 節-5.6 節は筋骨格構造にのみ適用可能である.

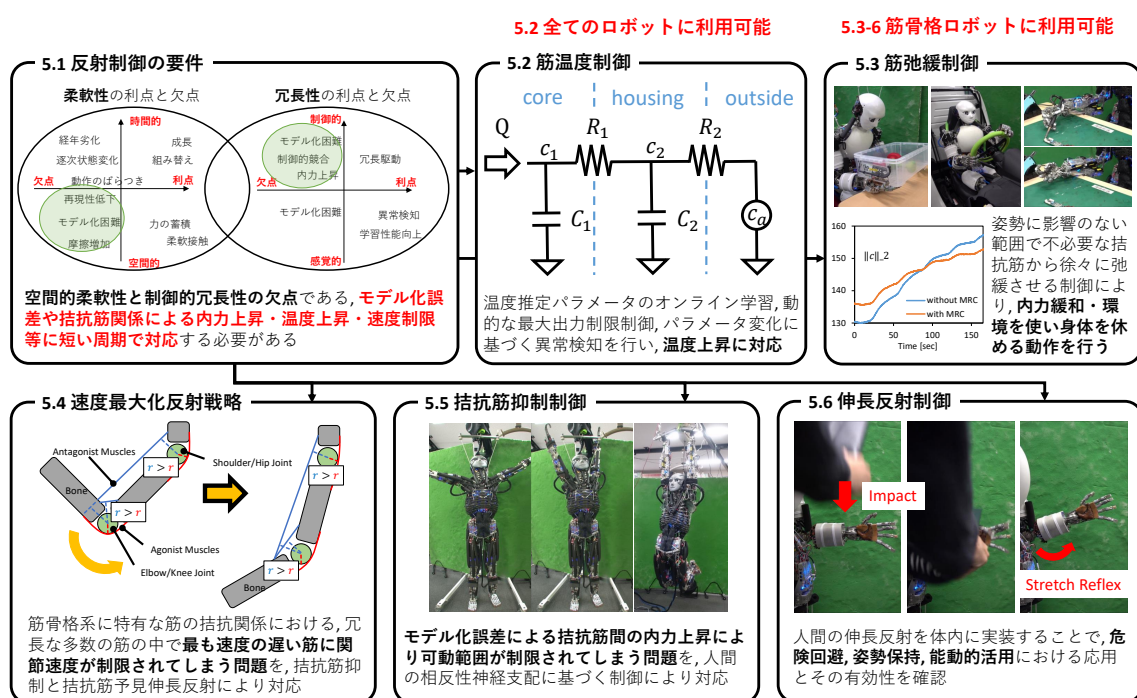


Fig. 5.1: The overview of this chapter.

5.1 反射制御の要件

反射とは、特定の刺激に対する反応の中でも、無意識的なものを言う。また、反射は意識的な行動に比べるとより速い周期で実行される。ここでは、柔軟性と冗長性の欠点補完の工学的な観点、そして人間に備わる生物学的な反射の観点から反射制御を考える。本研究は柔軟性と冗長性を持つロボットの身体図式学習に焦点を当てるため、前者が基本である。一方、後者の生物学的反射に関する知見からも反射制御を考えることで、柔軟性と冗長性を支える有用な反射制御を開発する。

5.1.1 柔軟性と冗長性の欠点補完

Fig. 5.2 に、柔軟性と冗長性の欠点を示す。本研究における反射制御は主に、関節や筋単体、それら同士の間には存在する比較的短い周期の低レイヤ制御である。そのため、時間的柔軟性や感覚的冗長性のような、長期的な変化、関節や筋以外の感覚との統合は扱わない。つまり、空間的柔軟性の欠点、制御的冗長性の欠点を補完するような反射制御を考える。

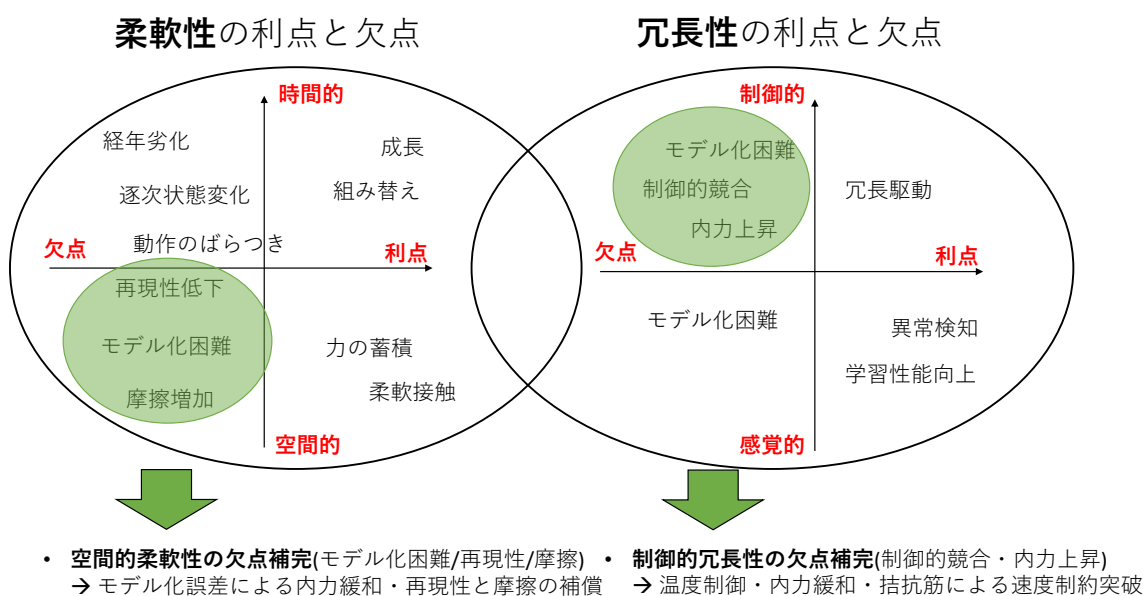


Fig. 5.2: The reflex controls compensating with the disadvantages of the flexibility and redundancy.

空間的柔軟性によって、軸駆動型のロボットに比べ、筋骨格型のロボットはモデル化が困難になる。また、筋骨格型の場合、柔軟な筋と骨格の間の摩擦は増加し、この影響で筋制御にヒステリシスが生まることが多い。よって、動かし方や外力等、様々な要因によって身体の状態が変化し、再現性が取り

にくいという問題がある。これら空間的柔軟性に対しては、このモデル化誤差によって起こる筋内力の緩和、そして再現性や摩擦等の補償が必要になる。

また制御的冗長性によって、冗長な制御入力により閉ループが構築され、モデル誤差と合わせて内力の過度な上昇が起こりうる。これは軸駆動型でも起こりうるが、関節と筋が常に閉ループを構成する筋骨格型に起こりやすい。これにより、モータの温度が過度に上昇する場合もある。また、筋骨格型の場合は動作に対して主動筋と拮抗筋が存在し、これらの関節に対するモーメントアームがそれぞれ大きく異なるため、動作速度が、最もモーメントアームが大きく遅い筋により制限されてしまうという問題がある。これら制御的冗長性に対しては、内力緩和、モータ温度の上昇を防ぐ制御、拮抗筋による速度制約を突破するための制御が必要になる。

以下に必要な反射制御をまとめる。

- モータ温度上昇を抑制する反射制御
- 筋内力緩和を行う反射制御
- 拮抗筋による速度制約を突破する反射制御
- 動作の再現性や摩擦等を補償する反射制御

モータの温度制御は様々なロボットに対して有用である一方、それ以外の反射制御については主に筋骨格型特有の問題を解決するものである。

5.1.2 人間の反射制御とロボットへの適用

人間に備わる生物的な反射機構について考える。なお、ここでは主に筋骨格型ロボットについて考察する。反射には様々なものがあるが、この中でも、筋単体、または筋同士の間が存在する最も基本的な3つの反射を以下に並べる (Fig. 5.3)。

- 伸長反射

急激な筋の伸びを検知する筋紡錘からくる求心性神経が、その同じ筋の α 運動ニューロンに興奮性接続することによって成り立つ。急激な筋の伸長に抵抗して筋が収縮することで、筋の過伸長を抑制する。

- ゴルジ腱反射

筋の持続的な伸びにより反応するゴルジ腱器官が、Ib群繊維、抑制性介在ニューロンを経て脊髄の α 運動ニューロンに接続することによって成り立つ。持続的な筋の伸長が加わると筋の収縮が抑制されることで、筋の過剰伸長による断裂を防ぐ。

- 相反性神経支配

筋紡錘からの Ia 求心性神経が主動筋の運動ニューロンを興奮させると同時に、抑制性のニューロンを介して拮抗筋の α 運動ニューロンを抑制するような神経回路が用意されることによって成り立つ。主動筋に対して拮抗筋が抑制されることで、身体の内力が高まらずスムーズに身体を動かすことができる。

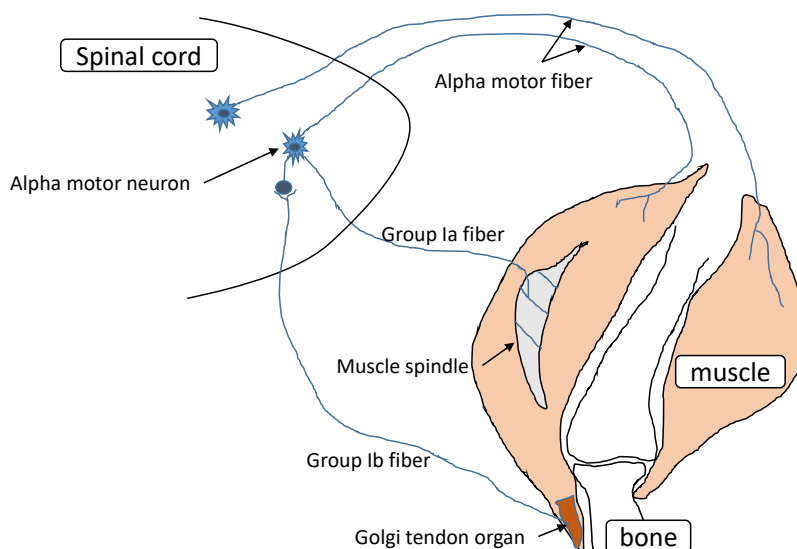


Fig. 5.3: The mechanism of human reflex.

この中でも、伸長反射は急激な筋の伸びによる過伸長を、ゴルジ腱反射は持続的な筋の伸びによる過伸長を抑制している。一方、人間の筋には直列弾性要素と並列弾性要素があると言われており、伸びすぎると筋を痛めてしまうのに対して、ロボットの筋はプーリによって巻き出されるため、これは考えなくて良い。よって、急激な筋の伸びによる過伸長はどのロボットでも危険であるため考慮することが大事であるが、持続的な筋の伸びによる過伸長を考える必要はない。そこで本研究では、主に伸長反射と相反性神経支配の筋骨格ヒューマノイドへの適用について考える。

ここで、人間の反射や筋のメカニズムと、5.1.1 節に述べた考えるべき反射制御と見比べる。筋内力緩和を行う反射制御、拮抗筋による速度制約を突破する反射制御は、主動筋に対して拮抗筋を緩めることになるため、相反性神経支配と似た反射構造を持つ。また、モータ温度上昇を抑制する制御は、人間の筋の最大張力が疲労によって低下する筋自体のメカニズムと似ている。動作の再現性や摩擦等を補償する反射制御は、後の実験から、伸長反射がその一部を補償することができていることがわかっている。

5.1.3 反射制御システムの概要

これらの議論から, 本研究では以下の 5 つの反射制御を実装する. なお, モータ温度制御については筋骨格型を例として実装するため筋温度制御と呼ぶが, 通常のどのロボットに対しても利用可能である.

- **筋温度制御**
モータハウジング温度からモータコア温度を推定し, これを定格以下に収まるように最大筋張力を制御する.
- **筋弛緩制御**
動作停止時, 姿勢維持に必要な筋を弛緩させることで内力を減少させる.
- **速度最大化反射戦略**
拮抗筋を抑制または予見伸長させることで, 拮抗筋による動作の妨げを防ぎ, 関節速度を最大化する.
- **拮抗筋抑制制御**
主動筋の動きに合わせて拮抗筋を抑制することで, モデル誤差による内力を減らし, 広可動域の動作を実現する.
- **伸長反射制御**
急激な筋の伸長の際に筋を収縮させることで, 筋の保護作用・ヒステリシスによる姿勢変化の低減・能動的誘発を使った摩擦の影響低減を実現する.

これらの反射制御を実装したシステムの概要を Fig. 5.4 に示す.

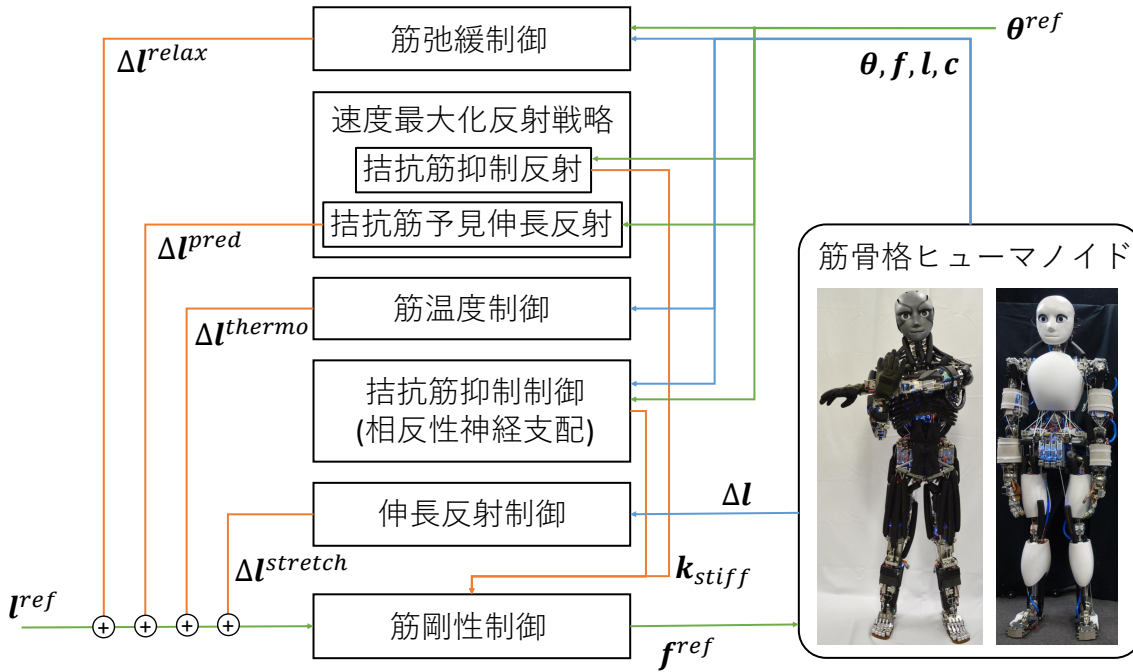


Fig. 5.4: Components of the developed reflex controls.

5.2 筋温度制御

5.2.1 概要と先行研究

モータ温度の管理は、ロボットの継続的な動作において極めて重要である。特に、等身大ヒューマノイド [78, 179] は、人間と同じような身長・体重を保つため、全ての動作に十分なモータを搭載することは難しい。そのため、関節トルクを制限し、温度を定格内に収める必要がある。

これまで、様々な方面からモータの温度管理を行う手法が考案されてきた。ハードウェアの工夫によりモータ温度を管理する手法として、モータを水冷することで温度の上昇を抑え、一瞬の大電流によりジャンプが可能なヒューマノイドが開発されている [180]。この他にも、ヒートシンクや空冷、相転移物質を用いたモータの温度管理方法も提案されている [181]。しかし、等身大ヒューマノイドが人間と同じような身長・体重を保つためには、ハードウェアの工夫には限界がある。

関節トルクを抑える制御として一般的な方法は、最適化の際に関節トルクを最小化する [182]、または最適化の際の不等号制約として関節トルクの最大値を設定する [183]、といったものである。しかし、これらは直接的に温度を扱っているわけではないため、温度が定格より小さくなっていることを保証するものではない。また最適化の際のコスト関数として温度を考慮した例として、モータハウジング温度の推定と温度緩和のための重心移動制御 [184] が存在するが、コスト関数は温度が定格よりも小さくなっていることを厳密に保証するものではない。

これに対して、浦田らはモータハウジング部の温度と電流からモータ内部のコア温度を推定し、コアの温度が定格よりも大きくならないような、ごく微小な時間流し続けられる最大の電流を計算して電流制限を行う手法を開発している [185]。また、熊谷らはモータハウジングの温度推定により 120 秒間発揮し続けられる最大の関節トルクによって制限を施した制御 [73] を提案した。

しかし、これまでに提案されたソフトウェアによる温度制御にはいくつかの問題点がある。まず、温度推定のパラメータはいくつか存在するが、このうち、外部温度が一定という仮定は往々にして間違っている。特に、屋外で夏と冬に実験するのでは、温度上昇の仕方は全く異なる。また、熱が溜まりやすいような部位のモータでは、外部温度は動的に変化する。その上、モータに関する熱抵抗や端子間抵抗も、劣悪なモータの使い方や経年劣化等によって徐々に変化していくものであり、これも一定とは限らない。[73] では温度推定パラメータの同定を行ってはいるが、オフラインの実行であり、また、温度モデルを簡易化しているためモータハウジング温度しかわからない。次に、温度制御の方法にも問題がある。これまで述べた手法では、ある一定期間同じ電流/関節トルクを発揮したときに、定格温度を上回らない最大の電流/関節トルクを使ってそれらを制限していた。[185] はごく微小な時間を考えて

いるため、ジャンプのような瞬発的な動作には適用可能だが、通常の動作では突然最大電流値が下がりロボットが停止してしまう。[73] では電流/関節トルクを動的に変えていないため、本当に出力可能な最大値よりも小さな値を最大値として使用することになってしまう。

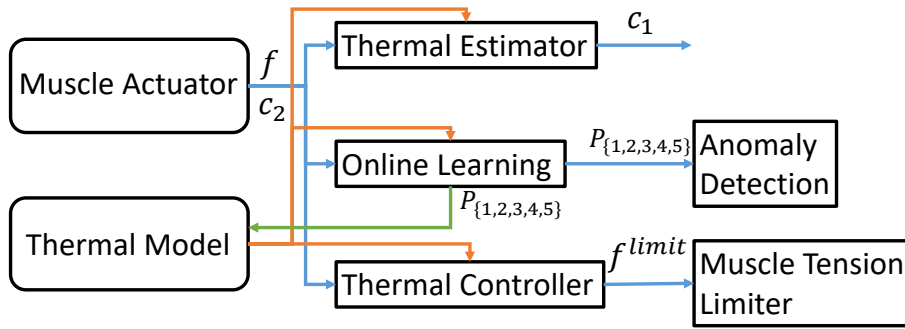


Fig. 5.5: System overview of thermal management [186].

そこで本節では、Fig. 5.5 に示すような、モータコア温度推定のパラメータのオンライン学習、またそのモデルを用いた最適化による動的な最大出力制限の制御を行う [186]。また、パラメータ変化を用いたモータの異常検知等を行う。本節では、本制御を筋骨格ヒューマノイドの全身の筋に適用することで、その有効性を確かめる。筋骨格ヒューマノイドは、人体を詳細に模倣しているがゆえ、多数の筋を有し、重量やプロポーションの制限が厳しく、空冷や水冷が現状では難しい（汗をかく手法も開発されている [114]）。そのため、ハードウェアの改造が難しく、ソフトウェアによる温度制限が適している。本節で扱う筋骨格ヒューマノイドは、筋が空気圧等ではなく、モータとプーリによって紐を巻き取る方式のものについて考える。本手法を軸駆動型ヒューマノイドに適用する際は、筋張力を関節トルクに変換して同様に適用することが可能である。

5.2.2 筋温度制御システム

まず、モータのコア温度を推定する際の基本的な2抵抗モデルについて述べる。その後、それらのパラメータを変数とするモデルを構築し、温度推定・パラメータのオンライン学習・異常検知・温度制限制御・筋張力制限制御について述べていく。

基本温度モデル

本手法では、[185] で述べられたものと同じ、Fig. 5.6 の2抵抗モデルをモータの温度モデルとして使用する。なお、本節では温度の単位は $^{\circ}\text{C}$ 、筋張力の単位は N 、筋長の単位は mm とする。モータコア

とハウジングに熱容量 C_1, C_2 , モータコアとモータハウジング間, モータハウジングと外気温間に熱抵抗 R_1, R_2 を仮定する. このとき, モータコア温度 c_1 , モータハウジング温度 c_2 , 外気温 c_a の間には以下のような関係が存在する.

$$C_1 \frac{dc_1}{dt} = Q - \frac{c_1 - c_2}{R_1} \quad (5.1)$$

$$C_2 \frac{dc_2}{dt} = \frac{c_1 - c_2}{R_1} - \frac{c_2 - c_a}{R_2} \quad (5.2)$$

$$Q = R_e i^2 \quad (5.3)$$

ここで, R_e は巻線電気抵抗, i は電流値である. 加えて, 本節の趣旨に沿うように, このモデルの熱量 Q を電流ではなく筋張力で表したい場合は, 以下の式を用いて電流 i を筋張力 f に変換する.

$$D_{pulley} f = E_{gear} D_{gear} E_{motor} K_t i \quad (5.4)$$

ここで, K_t はトルク定数, $E_{\{motor, gear\}}$ はモータまたはギアの伝達効率, D_{gear} はギア比, D_{pulley} はプーリの半径を表す. よって, $K = R_e (D_{pulley} / (E_{gear} D_{gear} E_{motor} K_t))^2$ として Eq. 5.1, Eq. 5.2 を整理すると, 以下ようになる.

$$\dot{c}_1 = \frac{K}{C_1} f^2 - \frac{c_1 - c_2}{R_1 C_1} \quad (5.5)$$

$$\dot{c}_2 = \frac{c_1 - c_2}{R_1 C_2} - \frac{c_2 - c_a}{R_2 C_2} \quad (5.6)$$

ここで, $\dot{c}_{\{1,2\}}$ は $dc_{\{1,2\}}/dt$ を表す. これら全てのパラメータはモータのデータシートから得ることができ, この漸化式 Eq. 5.5, Eq. 5.6 を離散的に繰り返すことで, モータコア温度 c_1 を得ることが出来る. またよくある構成として, モータのハウジングに温度センサを取り付けて c_2 を得ることができる場合, Eq. 5.5 のみを繰り返すことで c_1 を得ることができる.

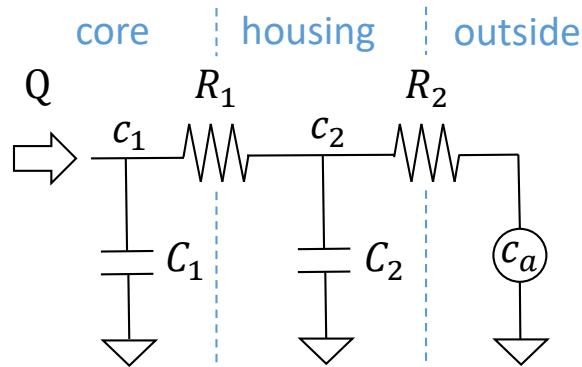


Fig. 5.6: Overview of a basic two-resistor thermal model [186].

しかし、実際のパラメータはデータシートから得られるものとは大きく異なる。これは、モータに接続する金属等への放熱や外気温の違い、モータの経年劣化や故障など、様々な要因が考えられる。その例として、新品のモータと半年間用いた回りの悪いモータを比べた。一定の 200 N を 90 秒間加えた際のモータハウジングの温度上昇を Fig. 5.7 に示す。温度上昇は 90 秒後で約 15 度程度異なっていることがわかる。モータコア温度はさらにセンシティブなため、より大きな差が生まれていると考えられる。よって、温度モデルは常に学習し続けられていくべきであると考え、その方法を以下で提案する。

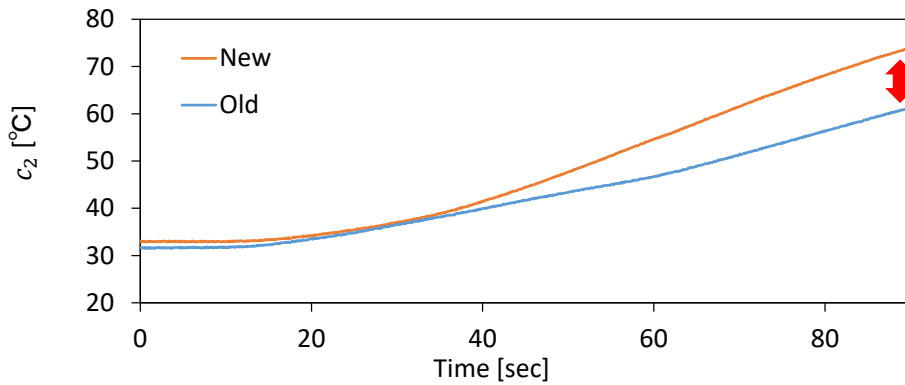


Fig. 5.7: Difference of transition of c_2 when applying $f = 200$ [N] between new and old motors [186].

提案温度モデル

本手法では、Eq. 5.5, Eq. 5.6 におけるパラメータを変数としておき、以下のようなモデルを作成する。

$$\dot{c}_1 = W_1 \exp(P_1) f^2 - \frac{c_1 - c_2}{W_2 \exp(P_2)} \quad (5.7)$$

$$\dot{c}_2 = \frac{c_1 - c_2}{W_3 \exp(P_3)} - \frac{c_2 - W_5(1 + P_5)}{W_4 \exp(P_4)} \quad (5.8)$$

ここで、 $W_1 = K/C_1$, $W_2 = R_1 C_1$, $W_3 = R_1 C_2$, $W_4 = R_2 C_2$, $W_5 = c_a$ とおいた。 $P_{\{1,2,3,4,5\}}$ は本モデルで学習されるパラメータであり、 $\exp(\cdot)$ は指数関数である。 $P_{\{1,2,3,4\}}$ を指数関数に通したのは、 $W_{\{1,2,3,4\}} \exp(P_{\{1,2,3,4\}})$ を必ず正とするためである。 $P_{\{1,2,3,4,5\}} = 0$ のとき、このモデルは Eq. 5.5, Eq. 5.6 と一致する。これらのパラメータは、 P_1 は電流からの熱量を決める係数、 P_2 はモータコアからモータハウジングに逃げる熱の時定数、 P_3 はモータコアからモータハウジングに流入する熱の時定数、 P_4 はモータハウジングから外部に逃げる熱の時定数、 P_5 は外気温度を表す係数、と解釈が可能である。

このモデルを関数 h_1, h_2 を使って簡易に表すと、以下のようになる。

$$\dot{c}_1 = h_1(f, c_1, c_2) \quad (5.9)$$

$$\dot{c}_2 = h_2(c_1, c_2) \quad (5.10)$$

本手法では、 $P_{\{1,2,3,4,5\}}$ をパラメータとして、Eq. 5.9, Eq. 5.10 を使って温度推定、温度制限制御を行っていくと同時に、これらパラメータをオンラインで実機に合うように更新していく。

モータコア温度推定

本手法では、一般的な構成として、モータのハウジング部に温度センサが取り付けられたような構成を考える。この場合、モータコア温度推定の方法は非常に単純である。温度推定の時間間隔 Δt_{est} を決め、以下のように c_1 を推定する。

$$c_{1,t+1} = c_{1,t} + h_1(f_t, c_{1,t}, c_{2,t})\Delta t_{est} \quad (5.11)$$

ここで、 $\{c_1, c_2, f\}_t$ は現在のタイムステップ t における $\{c_1, c_2, f\}$ を表す。

本節では、 $\Delta t_{est} = 0.02$ [sec] としている。

温度モデルのオンライン学習

モータのハウジング部に取り付けられた温度センサの値 c_2 と筋張力 f を教師として、温度モデルのパラメータ $P_{\{1,2,3,4,5\}}$ を更新していく。

まず、教師となるデータを蓄積する。データの蓄積間隔 Δt_{data} 、データ列数 N_{seq} 、バッチ数 N_{batch} を設定する。 Δt_{data} の間隔で連続する N_{seq} 個の $\{c_1, c_2, f\}$ のデータ $\{c_1, c_2, f\}^{data}$ を蓄積する。ここで、 c_1 は直接得られないため、前節で得られた推定値を蓄積している。この N_{seq} 個のデータ列を貯めていき、それらが N_{batch} 個得られたところで学習を実行し始める。これら N_{batch} 個のデータ集合を使って学習が終わったら、その内最初のデータを削除し、また N_{seq} 個のデータ列が集まりデータ集合が N_{batch} 個になったところでまた学習を実行することをオンラインで繰り返す。

次に、オンライン学習について説明する。モータコア温度推定には Eq. 5.9 のみしか用いなかったが、Eq. 5.10 によって、同時に c_2 を推定することが可能である。そこで、推定された c_2 と、実際に得られた c_2 を比較することで、パラメータ $P_{\{1,2,3,4,5\}}$ を更新していく。蓄積されたデータ列 $\{c_1, c_2, f\}_{[k, k+N_{seq}-1]}^{data}$ のうち、はじめに $c_{1,k}^{data}, c_{2,k}^{data}, f_{[k, k+N_{seq}-1]}^{data}$ を取り出す。その後、 $c_{1,k} = c_{1,k}^{data}, c_{2,k} = c_{2,k}^{data}, f_{[k, k+N_{seq}-1]} =$

$f_{[k,k+N_{seq}-1]}^{data}$ として、以下の漸化式を $N_{seq} - 1$ 回繰り返す.

$$c_{1,k+1} = c_{1,k} + h_1(f_k, c_{1,k}, c_{2,k})\Delta t_{data} \quad (5.12)$$

$$c_{2,k+1} = c_{2,k} + h_2(c_{1,k}, c_{2,k})\Delta t_{data} \quad (5.13)$$

すると、 $c_{1,[k+1,k+N_{seq}-1]}$, $c_{2,[k+1,k+N_{seq}-1]}$ が得られる. ここで、実機からデータ列として得られた $c_{2,[k+1,k+N_{seq}-1]}^{data}$ と現在のパラメータによって推定された $c_{2,[k+1,k+N_{seq}-1]}$ を以下の式で比較する.

$$L_{update} = \text{MSE}(c_{2,[k+1,k+N_{seq}-1]}, c_{2,[k+1,k+N_{seq}-1]}^{data}) \quad (5.14)$$

ここで、MSE は Mean Squared Error を表す. そして、この誤差からパラメータ $P_{\{1,2,3,4,5\}}$ を通時的誤差逆伝播法 [141] を用いて以下のように更新する.

$$P_{\{1,2,3,4,5\}} \leftarrow P_{\{1,2,3,4,5\}} - \alpha \frac{\partial L_{update}}{\partial P_{\{1,2,3,4,5\}}} \quad (5.15)$$

ここで、 α は学習率である. これは言ってみれば、ニューラルネットワークのパラメータを直接温度推定のパラメータとして、最急降下法によって更新していることに相当する. このとき、実際には N_{batch} 個のデータから計算された勾配の平均を持ってパラメータを更新する. また、このとき Gradient Clipping [187] を用いて、勾配のノルムの最大値を D_{clip} に設定している.

この手法の問題点として、 $c_{1,k}^{data}$ が実測値ではなく推定値であることが挙げられる. しかし、 c_2 の変化は c_1 の初期値に対してそこまで敏感ではなく、後の実験からも本手法により正確にパラメータが更新できることがわかる.

本節では、 $\Delta t_{data} = 1.0$ [sec], $N_{seq} = 30$, $N_{batch} = 10$, $\alpha = 0.02$, $D_{clip} = 5.0$ としている.

モータの異常検知

本手法では、オンラインで学習されたパラメータ $P_{\{1,2,3,4,5\}}$ が物理的な意味を持つため、この変化を用いて異常検知を行うことができる. これは、物理的な意味として解釈が難しい重みを学習するニューラルネットワークとの大きな違いである. 異常検知は単純に、以下の値 g が閾値 D_{detect} を超えたら異常と見なす.

$$g = \text{RMSE}([P_1, P_2, P_3, P_4]^T, [P_1^{init}, P_2^{init}, P_3^{init}, P_4^{init}]^T) \quad (5.16)$$

ここで、 $P_{\{1,2,3,4\}}^{init}$ は動作を開始したときの $P_{\{1,2,3,4\}}$, RMSE は Root Mean Squared Error を表す. P_5 は外気温のパラメータであり常に変わる可能性があるため、これは g から抜いている. 初めてオンライ

ン学習を実行する際は $P_{\{1,2,3,4\}} = 0$ から始めるため、 $P_{\{1,2,3,4\}}$ は大きく変化する可能性がある。そのため、一度パラメータがしっかりと学習されてから、異常検知は行うと良い。また、もう一つの最も単純な異常検知の方法は、この4つのパラメータを表示して監視することである。

本節では、 $D_{detect} = 1.0$ としている。

モータコア温度制御

モータコア温度制限制御は、モータコア温度 c_1 が設定した最大値 c_1^{max} に達するような滑らかな筋張力 f^{limit} の時系列を求め、その値を持って筋張力を制限する制御である。これは、ある時間間隔 $\Delta t_{control}$ とそれを展開する回数 $N_{control}$ を決め、 $\Delta t_{control} N_{control}$ 秒間なるべく速く c_1 が c_1^{max} を達成するための滑らかな f の時系列を最適化することに相当する。まず、現在推定されているモータコア温度を $c_{1,t}^{cur}$ 、温度センサから得られたハウジング温度を $c_{2,t}^{cur}$ とする。また、最適化する前の f の時系列 $f_{[k,k+N_{control}-1]}^{limit}$ を決定する。これらから、 $c_{1,k} = c_{1,k}^{cur}$ 、 $c_{2,k} = c_{2,k}^{cur}$ 、 $f_{[k,k+N_{control}-1]} = f_{[k,k+N_{control}-1]}^{limit}$ として、以下の漸化式を $N_{control} - 1$ 回繰り返す。

$$c_{1,k+1} = c_{1,k} + h_1(f_k, c_{1,k}, c_{2,k}) \Delta t_{control} \quad (5.17)$$

$$c_{2,k+1} = c_{2,k} + h_2(c_{1,k}, c_{2,k}) \Delta t_{control} \quad (5.18)$$

すると、 $c_{1,[k+1,k+N_{control}-1]}$ 、 $c_{2,[k+1,k+N_{control}-1]}$ が得られる。ここで、 c_1 の最大値 c_1^{max} を $N_{control}$ 個並べた $c_{1,[k+1,k+N_{control}-1]}^{max}$ と、推定された $c_{1,[k+1,k+N_{control}-1]}$ の誤差を以下の式で比較する。

$$L_{control} = \text{MSE}(c_{1,[k+1,k+N_{control}-1]}, c_{1,[k+1,k+N_{control}-1]}^{max}) + W_{control} \text{MSE}(0, f_{[k,k+N_{control}-1]}^{limit}) \quad (5.19)$$

ここで、 $W_{control}$ は右辺第一項と、 f^{limit} を最小化するような右辺第二項をバランスさせるための重みである。 f^{limit} の最小化項を加えることで、以降の $L_{control}$ を用いて最適化される f^{limit} の遷移が滑らかになり、最適化の際の安定性が増す。隣り合うタイムステップ同士の f^{limit} を最小化する方が本節の趣旨には合うが、実験からその項を入れると不安定になることがわかっている。この誤差から $f_{[k,k+N_{control}-1]}^{limit}$ を通時的誤差逆伝播法 [141] を用いて以下のように更新する。

$$f_{[k,k+N_{control}-1]}^{limit} \leftarrow f_{[k,k+N_{control}-1]}^{limit} - \beta \frac{\partial L_{control}}{\partial f_{[k,k+N_{control}-1]}^{limit}} \quad (5.20)$$

ここで、 β は学習率である。

実際には安全対策として筋張力の最大値と最小値 $f^{\{min,max\}}$ を決め、それよりも f^{limit} が大きくならないように制限している。また、最初に説明した $f_{[k,k+N_{control}-1]}^{limit}$ の初期値は、前ステップで最適化

された値 $f_{[k-1,k+N_{control}-2]}^{limit}$ をシフトして最終項を複製した $[f_{[k,k+N_{control}-2]}^{limit,T}, f_{k+N_{control}-2}^{limit}]^T$ としている。この最適化が初めて、つまり前ステップで最適化された値がない場合は f^{max} が $N_{control}$ 個並んだものを用いる。

本節では、 $\Delta t_{control} = 1.0$ [sec], $c_1^{max} = 80$ [°C], $N_{control} = 30$, $W_{control} = 0.001$, $\beta = 30$ [N], $f^{min} = 10$ [N], $f^{max} = 300$ [N] としている。

筋張力制限制御

得られた f_k^{limit} を現在の最大筋張力に設定し、筋張力制限をかける。筋張力制御 [188] であれば設定する筋張力最大値をこの値に設定するだけであるが、筋長制御の場合は以下のように、振動しないように筋長を緩め $l^{ref} + \Delta l_k$ をロボットに指令する。

$$\begin{aligned} & \text{if } f > f_k^{limit} \\ & \quad \Delta l_k = \Delta l_{k-1} + \min(D_{gain}d - \Delta l_{k-1}, \Delta l_{plus}d) \end{aligned} \quad (5.21)$$

$$\begin{aligned} & \text{else} \\ & \quad \Delta l_k = \Delta l_{k-1} + \max(0 - \Delta l_{k-1}, -\Delta l_{minus}d) \end{aligned} \quad (5.22)$$

$$d = |f - f_k^{limit}| \quad (5.23)$$

ここで、 $|\cdot|$ は絶対値、 l^{ref} は実機に送る筋長指令、 Δl_k はタイムステップ k における弛緩度、 $\Delta l_{\{minus, plus\}}$ はマイナス方向またはプラス方向に対する一ステップの筋長変化量を決める係数、 D_{gain} は最大弛緩量を決める係数である。この Δl_k は Fig. 5.4 における Δl_{thermo} に該当する。つまり、 $\Delta l_{minus}d$, $\Delta l_{plus}d$ で制限をかけながら、筋張力が最大値を越えないように筋を弛緩・緊張させている。この制御により、無理な筋長が送られたり力がかかったりしても、モータコア温度が c_1^{max} を越えないように、勝手に筋が弛緩するようになる。

本節では、 $\Delta l_{minus} = 0.001$, $\Delta l_{plus} = 0.003$, $D_{gain} = 2.0$ とし、本制御は 8 msec 周期で行う。

5.2.3 実験

一つの筋アクチュエータにおけるシミュレーション、実機実験を行い、最後に筋骨格ヒューマノイドの複数の筋における実機実験を行う。

本節で用いる Musashi [87] の筋アクチュエータ [152, 115] について簡易的に説明する。筋のモータは Maxon の brushless DC motor である、EC-4pole 22 90W with 29:1 gear と EC 16 60W with 128:1 を

用いており、肩・肘に用いられるのは前者、手首・指に用いられるのは後者である。これまで述べたように、モータハウジングに温度センサを有している。また、モータの先についたプーリに筋が巻かれており、筋張力を測定可能な筋張力測定ユニットから筋が出て行く。また、電流制御可能なモータドライバがカバーで覆われモータ上部に接続されている。

ここで、デフォルトのパラメータについて記載する。EC-4pole 22 90W with 29:1 に関しては、 $C_1 = 2.10$ [J/K], $C_2 = 29.0$ [J/K], $R_1 = 1.20$ [K/W], $R_2 = 10.3$ [K/W], $K = 2.97E - 4$ [J/N²s] である。また、EC 16 60W with 128:1 に関しては、 $C_1 = 1.19$ [J/K], $C_2 = 12.2$ [J/K], $R_1 = 4.30$ [K/W], $R_2 = 39.5$ [K/W], $K = 4.50E - 5$ [J/N²s] である。 c_a については、本節では 30°C としている。

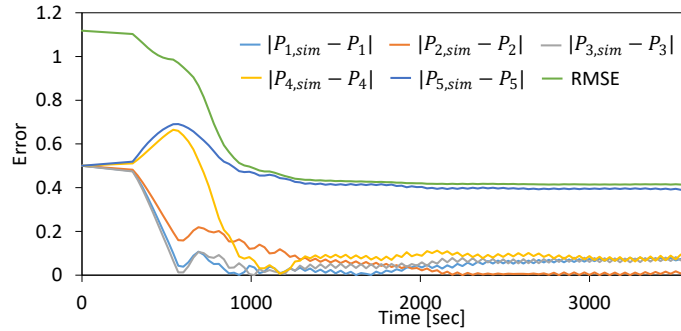


Fig. 5.8: Transition of errors in thermal model parameters between simulator and current model during online learning [186].

1 筋によるシミュレーション実験

まずオンライン学習について実験を行う。アクチュエータのパラメータは EC-4pole 22 90W with 29:1 を $P_{1,sim} = 0.5$, $P_{2,sim} = 0.5$, $P_{3,sim} = -0.5$, $P_{4,sim} = -0.5$, $P_{5,sim} = 0.5$ として、変更したものをシミュレータとして用いる。なお、初期の c_1, c_2 はともに 30°C とする。1 秒ごとに $f \leftarrow f + \text{Rand}(-50.0, 50.0)$ ($\text{Rand}(a, b)$ は $[a, b]$ のランダムな値である、また、 f は最小値 10 N, 最大値 200 N として制限をかけている) で f を更新しながら、その f をこのシミュレータに対して指令する。 c_2, f のみを観測データとしてシミュレータから得ることができる。これに対して現在、デフォルトの $P_{\{1,2,3,4,5\}} = 0$ のモデルを持っているとする。このモデルを使って、観測された f と c_2 から c_1 を推定しておく (同様に c_1 の初期値は 30°C とする), 5.2.2 節に示したように、 (c_1, c_2, f) からオンライン学習を実行する。これにより、現在のモデルをシミュレータに近づけていく。シミュレータと現在のモデルのそれぞれのパラメータの差の絶対値 $|P_{\{1,2,3,4,5\},sim} - P_{\{1,2,3,4,5\}}|$ とそれら 5 つのパラメータの RMSE の遷移を 3600 秒にわたって Fig. 5.8 に示す。それぞれのパラメータが徐々にシミュレータの値に近づいていくことがわか

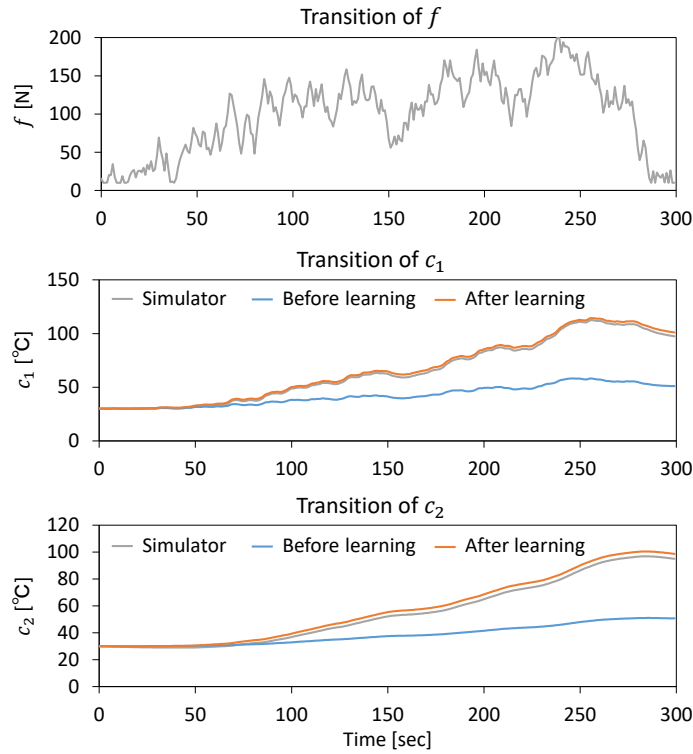


Fig. 5.9: Transition of (f, c_1, c_2) of the simulator and the estimated c_1 and c_2 using the model before and after online learning [186].

り、約 1200 秒付近で収束している。 P_5 の誤差が下がり切っていないが、これは P_5 を更新するのに優れた c_2 の温度変化が得られなかったためであると考ええる。

また、3600 秒間オンライン学習を実行した後に、上記と同様に f をシミュレータ・学習前のモデル・学習後のモデルに与え、そのときの f, c_1, c_2 の遷移を Fig. 5.9 に示す。 c_1, c_2 について、学習後のモデルによる温度推定が学習前に比べてシミュレータに近づいていることがわかる。教師として c_2 のみを用いて学習が行われているが、 c_1 もシミュレータに近づいており、本手法が有効であることがわかる。

次に、オンライン学習の定量的な評価を行う。まず、デフォルトの $P_{\{1,2,3,4,5\}}$ からランダムに変化を加えた $P_{\{1,2,3,4,5\},sim}$ を 10 個作成する。このとき、デフォルトの値と変化を加えた値の RMSE が 0.5 になるようにする。5.2.3 節と同じ実験を 10 個のパラメータについて実行したときの、0, 10, 20, 30, 40, 50, 60 分後の $|P_{\{1,2,3,4,5\},sim} - P_{\{1,2,3,4,5\}}|$ とそれら 5 つのパラメータの RMSE の平均と分散を Fig. 5.10 に示す。ここで、5.2.2 節で指摘した、 c_1 の初期値に実機値ではなく推定値しか得られないという問題についても考える。つまり、5.2.2 節における c_1^{data} を、実機値としたときと、推定値としたときとの差についても Fig. 5.10 に示す。グラフから、 c_1^{data} は実機値を用いた方が推定値を用いた場合より誤差が

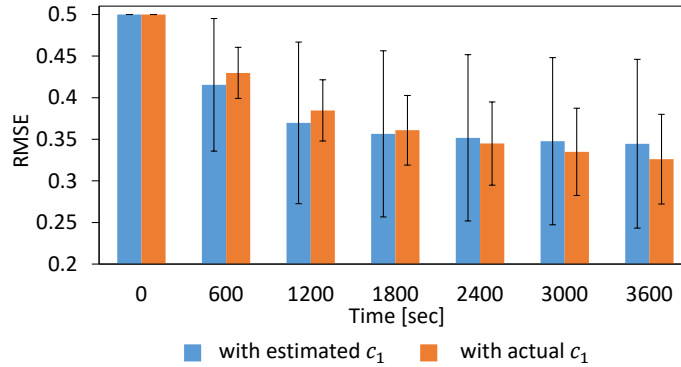


Fig. 5.10: Quantitative evaluation of online learning [186]. The graph shows the transition of RMSE of thermal model parameters between the current model and the simulator with 10 modified parameters during online learning using the estimated or actual c_1 .

速く減っていき、分散も少ない。しかし、分散はある程度あるものの、推定値を用いても徐々にモデルのパラメータがシミュレータに近づいていることがわかる。よって、本手法が有効であることが示された。

次に、異常検知手法について実験を行う。シミュレータのパラメータをデフォルトの $P_{\{1,2,3,4,5\},sim} = 0$ とし、オンライン学習を実行する。このとき、常に観測できる値が $c_2 = 30$ で一定になってしまう状態、または、観測できる f は更新される値だが、実際にはシミュレータに対して $f = 200$ が常にかかっている状態、の2つを考える。前者は温度センサが壊れている状態、後者はギアや筋が引っ掛かり、筋が思ったように動かない状態を想定している。このときのオンライン学習によるパラメータ誤差 $|P_{\{1,2,3,4\},sim} - P_{\{1,2,3,4\}}|$ の変化、異常検知の値 g の遷移を Fig. 5.11 の上図に示す。また、Fig. 5.11 の下図に、 $P_{\{1,2,3,4\}}$ の変化も同様に示す。前者の場合はオンライン学習が開始してから約400秒、後者は600秒で異常が検知されていることがわかる。実際の $P_{\{1,2,3,4\}}$ の値を見ると、前者では P_2 を小さく（モータコアの熱容量を下げる）してモータコアからモータハウジングに逃げる熱の量を増やし、 P_3 を上げる（モータハウジングの熱容量を上げる）ことでモータコアからの熱の流入を減らしており、モータハウジング温度を30度で一定に保つための変化が起きていることがわかる。また後者では、 P_4 を増やしてモータハウジングの熱が外部に流れるのを防ぎ、 $f = 200$ で一定のため常に高い c_2 が観測され続ける状態を再現している。このように、温度センサが壊れた、ギアが壊れてモータが回らない、モータが焼けて動かずに温度のみが上昇していく、等の異常を検知でき、かつその状況をパラメータの解釈性から読み取ることができる可能性がある。

最後に、モータコア温度制御について実験を行う。5.2.3 節の学習前のモデル・学習後のモデルにつ

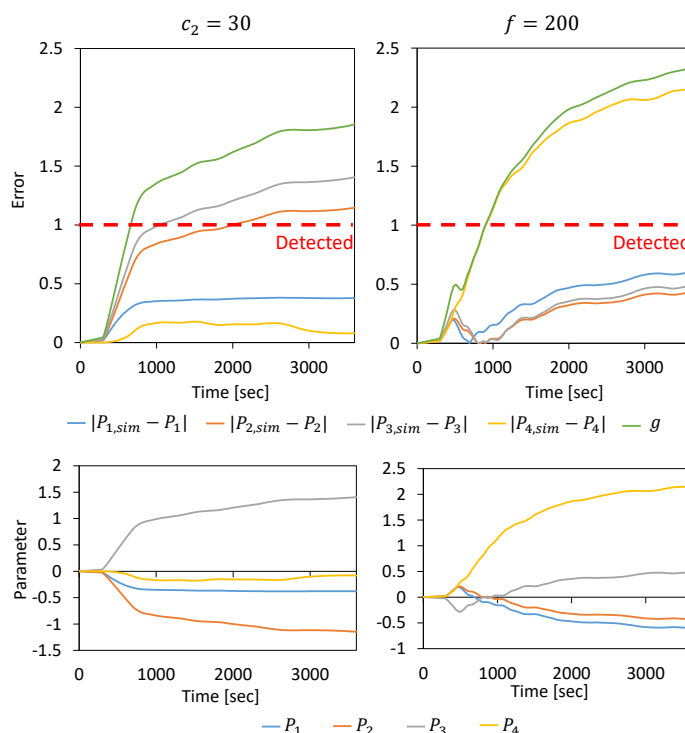


Fig. 5.11: Anomaly detection by learning thermal model parameters [186]. The graph shows the transition of errors in thermal model parameters and g during online learning.

いて、 c_1, c_2 が現在両者とも 60, または 75 のときに、どのような $f_{[k, k+N_{control}-1]}^{limit}$ が計算されるのかを観察する。また、実際に計算された $f_{[k, k+N_{control}-1]}^{limit}$ をシミュレータに入力したときの c_1 の遷移を学習前と学習後のモデルで比較する。Fig. 5.12 に示すように、初期温度が 60, 75 の両者で同様に、最初は高く、徐々に値が落ちて最終的になだらかで一定となるような f^{limit} が計算されていることがわかる。また、 c_1 の遷移は、学習後のモデルでは正しく最大値に設定した c_1^{max} を実現できているのに対して、学習前のモデルでは c_1 が c_1^{max} を大きく超えてしまっていることがわかる。よって、温度制限制御を行うにあたって温度モデルのオンライン学習は有効であり、また、本手法によってモータコアの温度を c_1^{max} 以下に常に保ち続けることができることがわかった。

1 筋による実機実験

一本の実際の筋アクチュエータを使って、オンライン学習を評価する。筋の末端の位置を固定したうえで、10 秒かけて $\text{Rand}(-16, 0)$ [mm] の筋長を送って筋を緊張させ、10 秒かけて戻すことを繰り返す。その際の筋張力変化、同時にオンライン学習を実行した際の温度モデルパラメータの推移を Fig. 5.13

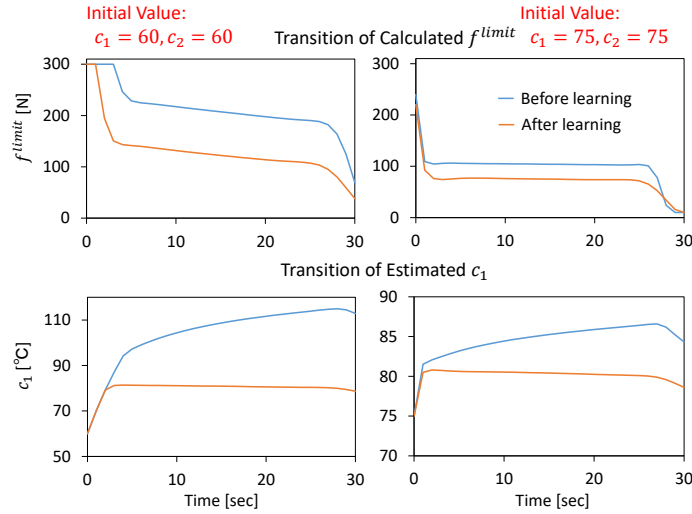


Fig. 5.12: Calculation of the optimized muscle tension to control c_1 and its evaluation [186]. The graph shows the transition of the calculated f^{limit} and observed c_1 with initial c_1 and c_2 of 60 or 75.

に示す。その後、もう一度同様に筋長を変化させたときに、実機から得られた c_2 、学習前と学習後におけるモデルを使って推定した c_2 を Fig. 5.13 に比較する。学習後は、学習前に比べて大きく推定値が実機に近づいていることがわかる。 c_1 の実機値は得ることができないため比較は難しいが、5.2.3 節の Fig. 5.9 から c_2 の推定が近づくことで c_1 の推定もより正しくなっていると考えられる。そのため、実機においても本手法が適用可能なことがわかった。

学習されたモデルを用いて、5.2.2 節、5.2.2 節を用いて、 c_1 を常に c_1^{max} 以下に保つことができるかを検証する。筋長として -16 mm を送り、約 200 N の筋張力を発生させる。これと同時に筋張力制限制御によって、筋張力を制限させる。その際の f 、計算された f^{limit} 、筋弛緩量 Δl_{thermo} 、推定された c_1 、実測値 c_2 の遷移を Fig. 5.14 に示す。高い筋張力をかけ続けることで c_1 は徐々に上がっていき、約 70 sec の時点で計算された f^{limit} の値が f^{max} よりも下がり始める。そして、約 90 sec のところで f が f^{limit} を越え、 Δl_{thermo} が徐々に増え、筋張力が抑制されている。この制御により、 c_1 がぴったりと c_1^{max} を実現し続けていることがわかる。モータコア温度を c_1^{max} 以下に抑えるためには f^{max} を 100 N 程度に抑える必要があるが、本手法により、通常はより大きな力を出し、モータコア温度を見ながら出力を下げていくことができることがわかった。

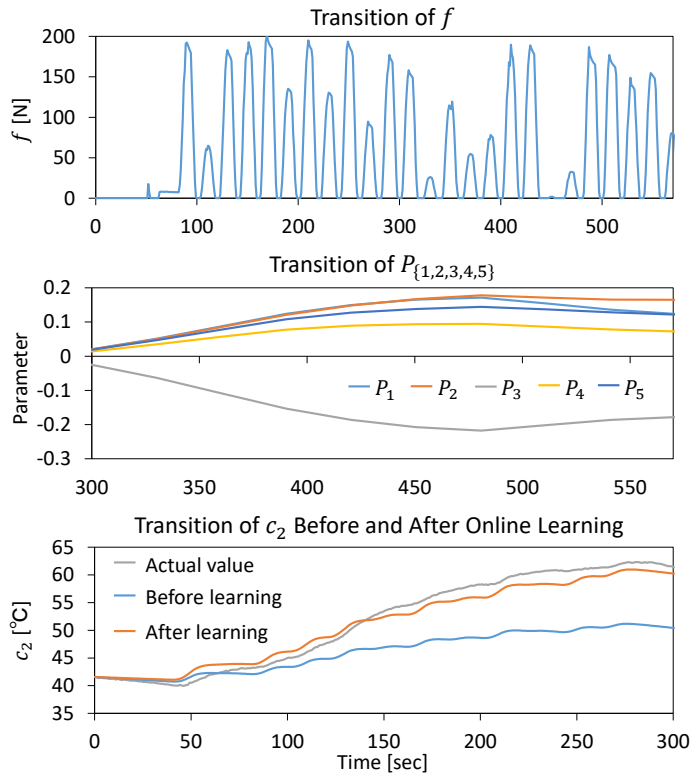


Fig. 5.13: Online learning experiment with one actual muscle actuator [186]. The upper graph shows the transition of the applied f during online learning, the middle graph shows the transition of $P_{\{1,2,3,4,5\}}$, and the lower graph shows the transition of the estimated c_2 before and after online learning.

筋骨格ヒューマノイドへの適用

筋骨格ヒューマノイド Musashi [87] の左腕の筋に対して本手法を適用し、実際のロボットの動きの中でどのように本手法が寄与するかを考察する。オンライン学習・モータコア温度推定・筋張力制限制御を走らせながら、Fig. 5.15 のようにランダムな関節角度を送り続けることを行う。この際に、[125] の可変剛性制御を用い、高い剛性を実現する一方で大きな力が常にかかり続けるような状態で実験を行う。ここで重要なのは、 c_1 を制限するだけでなく、推定されたモータコア温度を Fig. 5.16 に示すように、色をつけて常に監視することである。オンライン学習によってより正確になったモータコア温度を監視し、実験中にいち早く異常に気づけるようにすることが重要である。また、この際の f, c_1 の推移を、肩・肘に寄与する 10 本の筋#1-#10 に関して、Fig. 5.17 に示す。主に高い筋張力を発しているのは#1, #2, #6, #8 である。動作初期は最大で 350 N 程度の強い力がかかることがあるが、 c_1 が上がるにつれ、最大筋張力が 250 N 程度まで筋張力が抑えられるように変化している。特に、#1, #2, #6 の筋

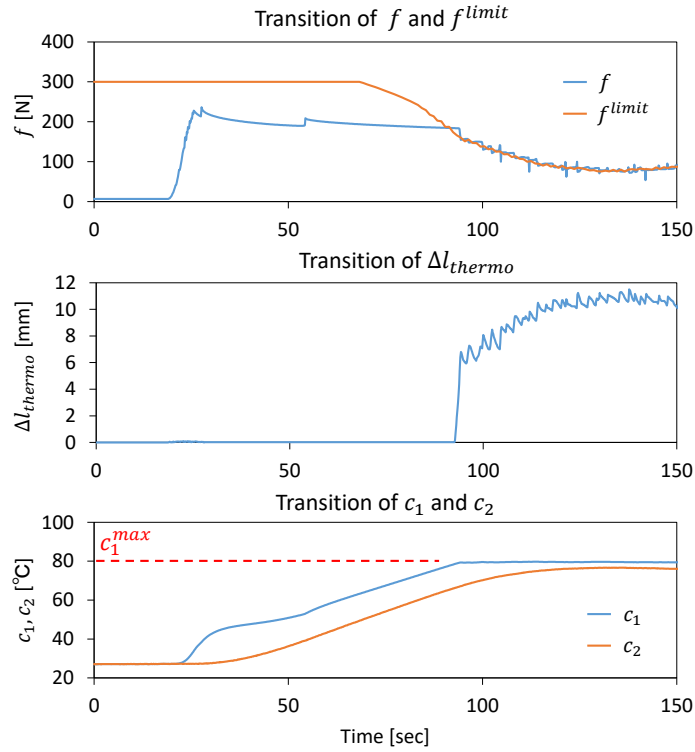


Fig. 5.14: Thermal control experiment with one actual muscle actuator [186]. The upper graph shows the transition of f and calculated f^{limit} , the middle graph shows the transition of Δl_{thermo} , and the lower graph shows the transition of c_1 and c_2 .

は c_1 が 80 度付近までいっては戻るような動きをしており、筋張力制限制御により、継続的な動作が可能となっていることがわかる。このように、本手法を複数のアクチュエータに対して適用し、実際の動作の中でも有効であることが示された。

最後に、例えば Musashi の両腕 36 自由度に本手法を適用した場合、温度推定は 5 msec、温度制御は 200 msec、オンライン学習は 200 msec 程度の時間が必要になる。本手法のプログラムは全て Python で書かれているため、コンパイル言語で書き直す、または N_{seq} や $N_{control}$, N_{batch} などを変更することで、より速い Hz で動作させることが可能である。

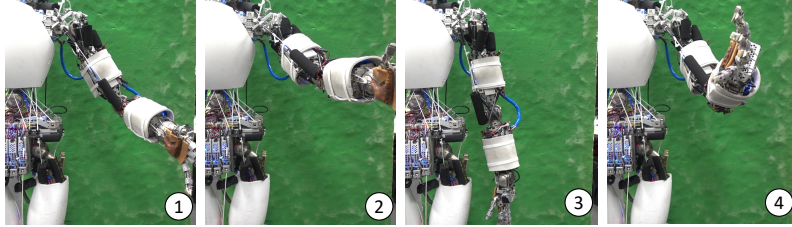


Fig. 5.15: Random movements of the left arm of the musculoskeletal humanoid Musashi [186].

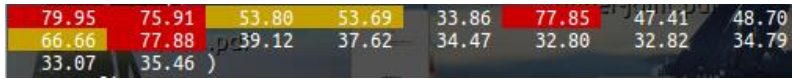


Fig. 5.16: Monitoring of c_1 with colors for manual thermal management [186]. Each value shows c_1 included in the left arm of Musashi. When the value is colored red, the temperature is high (>70 °C), and when the value is colored yellow, the temperature is slightly high (>50 °C).

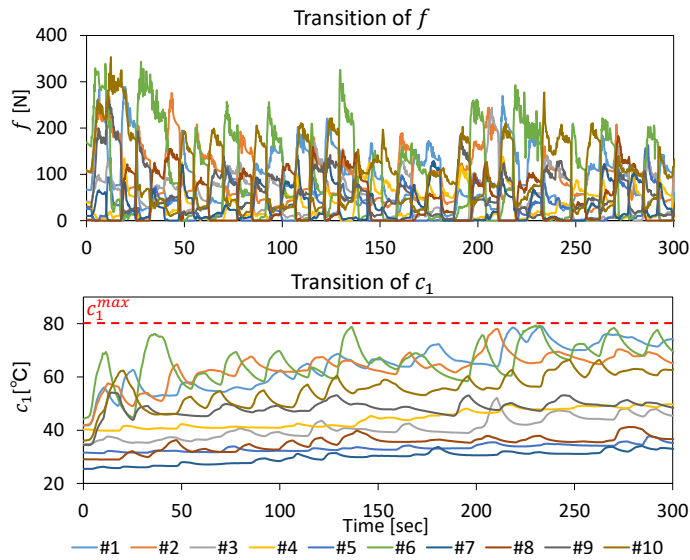


Fig. 5.17: Transition of f and c_1 during random movements of the left arm of Musashi [186].

5.3 筋弛緩制御

5.3.1 概要と先行研究

筋骨格ヒューマノイドは、筋の冗長性や非線形弾性要素による可変剛性制御、特異点のない球関節や劣駆動な背骨・指等、生物規範型の様々な利点を有することはこれまでも述べた。また、その劣駆動性・柔軟で冗長な腱・衝撃に強い球関節等の特徴から、通常の軸駆動型ヒューマノイドに比べ、環境接触下における動作実現に適している (Fig. 5.18)。しかし、筋骨格ヒューマノイドにおいて内力を抑えて継続的に環境接触行動を行うことは難しい。これには、2つの問題が起因していると考える。

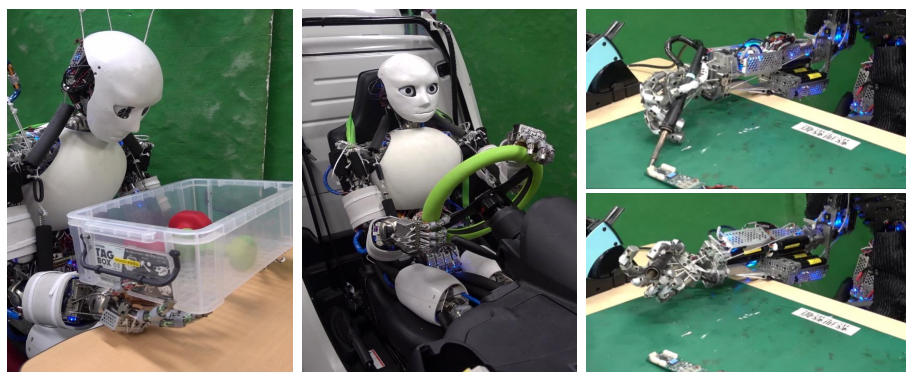


Fig. 5.18: Motions with environmental contact by musculoskeletal humanoids [189].

1 つめに、筋骨格ヒューマノイドはその複雑な身体構造ゆえにモデル化が難しく、実機と幾何モデルの間に大きな誤差が生じる点である。そのため、筋の拮抗関係の誤差により内力がたまり、筋温度が急上昇したり、ロボットが破損したりする可能性がある。これに関しては様々な解決方法が提案されている [128, 100, 124]。一方、これらの手法は内力をある程度解消することが可能であるが、ヒステリシス等のモデル化困難な項は依然として残り、それらを解決するために反射型の手法も開発されている。浅野らは実機センサデータから筋張力を分配することで、筋の負荷を和らげる手法を開発している [190]。河原塚らは主動筋に対して拮抗筋を緩めることでモデル誤差を許容する拮抗筋抑制制御を開発している [167]。しかし、前者は負荷を分散する筋群を手動で決める必要があり、後者は指令関節角度と関節角度推定値との差に敏感であるため静的動作において上手く機能しない等、様々な問題を抱える。

2 つめに、筋骨格ヒューマノイドは身体の正確な位置決めが難しく、環境との接触により内力が発生したり、逆に環境を上手く利用したりすることが難しい点である。前者に関しては、筋骨格ヒューマノイドの腱が持つ弾性、そして非線形弾性が効果を示し [154, 191]、環境認識や動作に誤差があっても、

それを柔軟に許容可能である。しかし同時に、長時間の内力は筋に対して負荷となり、必要のない内力は除去すべきである。また、環境を積極的に利用した動作研究は乏しい。

本節では、これら2つの問題を解決する手法として、筋弛緩制御を提案する [189]。これは、姿勢に影響を及ぼさない範囲で徐々に筋を弛緩させていくというシンプルな制御である。本手法を用いることで、1つめの問題である拮抗関係における筋内力を抑え継続的な動作が可能となると同時に、2つめの問題である環境との接触による内力の抑制、そして環境を使って身体を休める動作等が可能となる。

5.3.2 筋弛緩制御システム

全体システム

全体のシステムを Fig. 5.19 に示す。

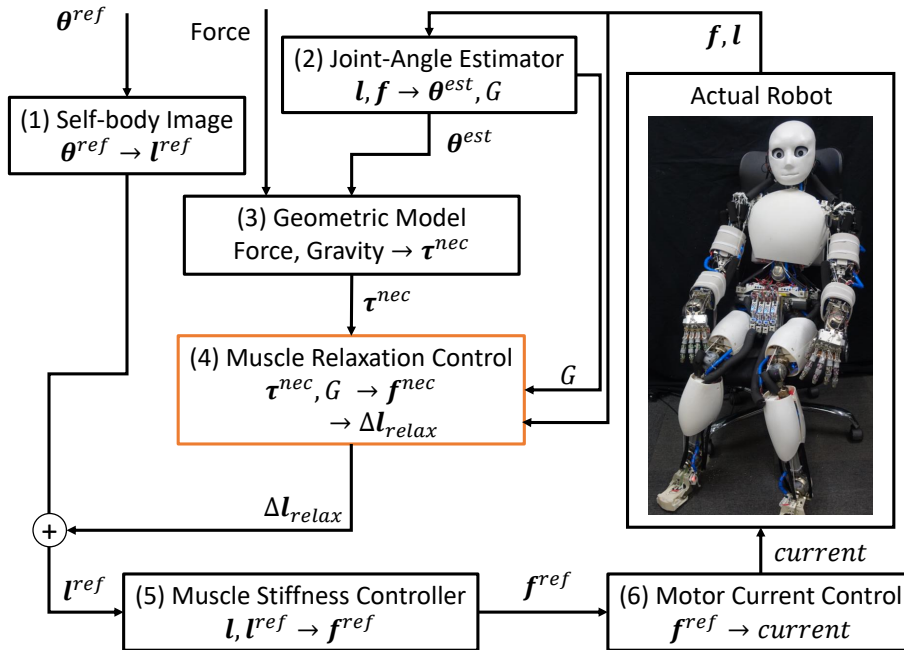


Fig. 5.19: The whole system of the muscle relaxation control [189].

まず、(1)では2.5.2節の自己身体モデル [124] を用いて指令関節角度 θ^{ref} を指令筋長 l^{ref} に変換する (ここでは Eq. 2.20 のみ考えている)。幾何モデルよりも正確に θ^{ref} が実現でき、筋内力の高まりも抑えられるものの、筋の摩擦・ヒステリシス等の問題により完全には解決されない。

次に、(2)では2.5.2節の自己身体モデル [124] を用いた関節角度 θ^{est} の推定・筋長やコピアン G の導出を行う。

次に, (3) では現在のタスクにおける必要な力や関節角度推定値 θ^{est} , 重力等から, 現在必要なトルク τ^{nec} を計算する. ここでは, 関節の構造に関する幾何モデルを既知として計算を行う.

次章で説明する (4) の筋弛緩制御において得られた筋弛緩項 Δl_{relax} から, l^{ref} を以下のように更新する.

$$l^{ref} \leftarrow l^{ref} + \Delta l_{relax} \quad (5.24)$$

(5) では, Eq. 2.4 の筋剛性制御 [121] により指令筋張力 f^{ref} が計算され, (6) において最終的に電流が実機のモータに指令される.

この中で, (2), (3) は 20 msec 周期, (1), (4), (5) は 8 msec 周期, (6) は 1 msec 周期で行われている.

筋弛緩制御アルゴリズム

筋弛緩制御は非常に単純な制御構造をしている.

まず, (2), (3) から得られた G, τ^{nec} から, 現在必要と考えられる筋張力 f^{nec} を, 以下のような二次計画法を解くことで計算する.

$$\text{minimize } \mathbf{x}^T W_1 \mathbf{x} + (G^T \mathbf{x} + \tau^{nec})^T W_2 (G^T \mathbf{x} + \tau^{nec}) \quad (5.25)$$

$$\text{subject to } \mathbf{x} \geq f^{min} \quad (5.26)$$

ここで, W_1, W_2 は重みを表し, 本節ではそれぞれ $I \times 1.0E - 5, I$ とする (I は単位行列を表す). 計算された \mathbf{x} を $f^{nec} = \mathbf{x}$ とする. これは, 通常は $\tau^{nec} = -G^T \mathbf{x}$ とするのが一般的であるが, G の誤差から解が求まらない可能性があるため, より誤差を許容できるこの形を採用した.

次に, f^{nec} の値から筋を昇順にソートし, f^{nec} の小さい, つまり本タスクに必要な筋 i から順に見ていく. 本手順を Fig. 5.20 に示す. 現在の実機の筋張力センサ値 f_i が $f_i < f^{min}$ の場合は次の筋 $i+1$ へ移動する. もし $f_i > f^{min}$ の場合は, Δl_i の値を確認する. もし $\Delta l_i + \Delta l_+ > \Delta l^{max}$ ならば, 次の筋 $i+1$ へ移動する. $\Delta l_i + \Delta l_+ < \Delta l^{max}$ ならば, $\Delta l_i \leftarrow \min(\Delta l_i + \Delta l_+, \Delta l^{max})$ とし, Δl_i を更新する. 一度でも Δl_i の更新に成功すればループを抜け, 成功しない場合は最後の筋まで上記の工程を繰り返す.

これが基本的な筋弛緩制御の動きであるが, その作動に関してはいくつかの条件が入る. まず, 筋弛緩制御は筋長指令 l^{ref} が変動しない, つまり静止状態においてのみ動作する. l^{ref} が変動している, つまり動作中に関しては, 逆に f^{nec} が高い筋から順に, $\Delta l_i \leftarrow \max(\Delta l_i - \Delta l_-, 0)$ のように, Δl_i を元に戻していく. これも同様に, 一度でも Δl_i の更新に成功すればループを抜け, 成功しない場合は最後の筋まで上記の工程を繰り返す. また, l^{ref} が静止した瞬間の関節角度を θ^{init} , 現在の関節角度を θ とし

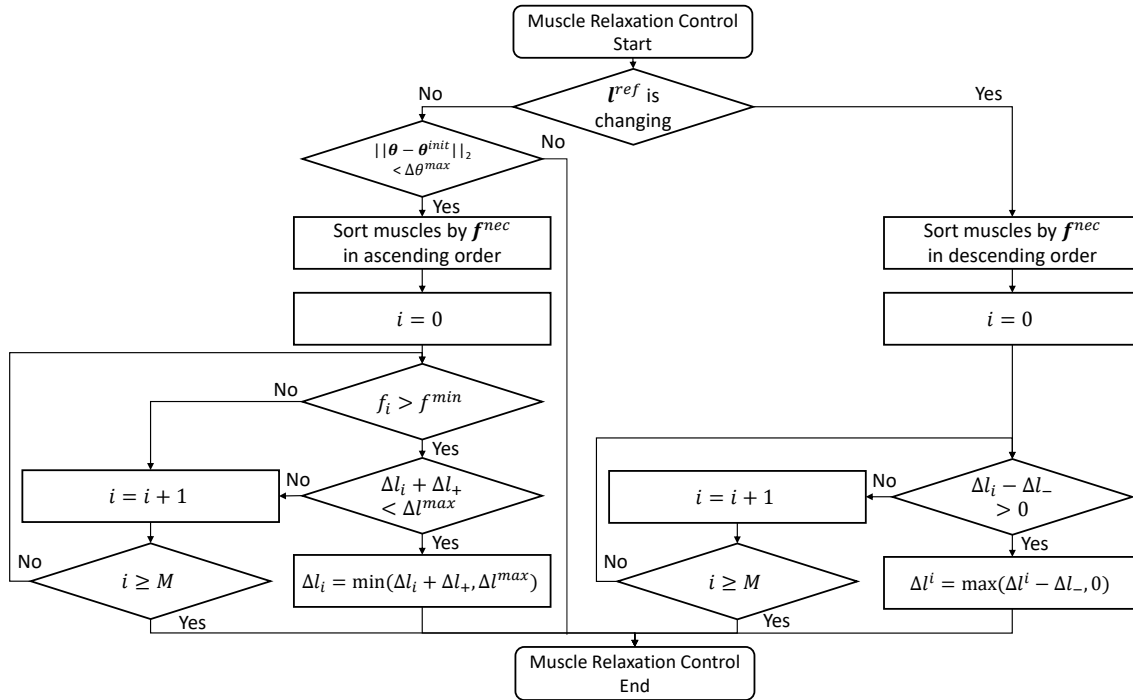


Fig. 5.20: Flow chart of muscle relaxation control during one time step [189].

たときに, $\|\theta - \theta^{init}\|_2 > \Delta\theta^{max}$ となった場合には筋弛緩制御を停止する. これは, 筋の弛緩によって姿勢に大きな影響を及ぼさないようにするためである. θ は関節角度推定値 θ^{est} を用いても良いし, 関節角度センサが備わっている場合はその値, 備わっていない場合は視覚と筋長変化から算出した値 $\theta^{est'}$ を用いても良い.

本節では, $f^{min} = 30$ [N], $\Delta l_- = \Delta l_+ = 0.03$ [mm], $\Delta l^{max} = 2.0$ [mm], $\Delta\theta^{max} = 0.1$ [rad] とする.

筋弛緩制御の特徴

筋弛緩制御は簡単に言うと, 姿勢を崩さない範囲内で, 必要のない筋から順に筋を弛緩させていくことに相当する. 筋骨格ヒューマノイドは主動筋と拮抗筋を有し, その動作を主に遂行する筋が主動筋, それに追従する筋が拮抗筋である. つまり, なるべく必要のない拮抗筋から緩み, 最終的には主動筋も緩んでいくことになる.

筋弛緩制御を実行することで, 通常の動作においては, 拮抗筋が弛緩することで, 内力の高まりを抑えることができる. 拮抗筋が弛緩し終わり, 主動筋が緩む際には, 姿勢の崩れが起きるため, 主動筋の弛緩は途中で止まる. 一方, 柵を掴む等の環境に身体を拘束させて行う動作の際は, 主動筋を弛緩させ

でもある程度までは姿勢に崩れが起きることはない。よって、拮抗筋だけでなく主動筋も緩んでいき、身体の内力だけでなく環境との接触の内力も小さくなっていく。

基本的な動作や環境接触には広く使えるものの、筋弛緩制御の適用範囲はいくつかの制約を受ける。筋弛緩制御は内力により剛性を制御する可変剛性制御と併用することはできない。また、動作しながら環境に力を発揮する場合は良いものの、動作せず一定の力を継続的に環境に発揮したい場合等には向いていない。しかし同時に、継続的な強い力を必要としないテーブル拭きや電車のつり革に掴まる動作、ハンドル操作や柵に掴まる動作等、多くの環境接触動作に適用可能である。

5.3.3 実験

本手法を用いて、いくつかの状況に応じた筋弛緩制御の動きと効果について実験・考察し、有効的な使い方について説明する。単腕・双腕の実験についても左腕の肩と肘を合わせた結果のみ示す。

本節では筋骨格ヒューマノイド Musashi [87] を用いる。本節では主に、肩の 3 自由度、肘の 2 自由度を用いて実験を行うこととし、関節角度は $\theta = (\theta_{S-r}, \theta_{S-p}, \theta_{S-y}, \theta_{E-p}, \theta_{E-y})$ のように表す (S は shoulder, E は elbow, $ropy$ はそれぞれ roll, pitch, yaw を表す)。また、肩と肘の 5 自由度は筋を 10 本含み、その筋配置を Fig. 4.24 に示す。

基本動作実験

基本的な動作群を行い、その際の筋弛緩制御の動きについて考察する。Fig. 5.21 のように関節角度をランダムに指定して 3 秒間で送り、3 秒間静止する動作を複数回行う。その際の ΔI_{relax} , θ , $\|f\|_2$ の遷移を考察する。

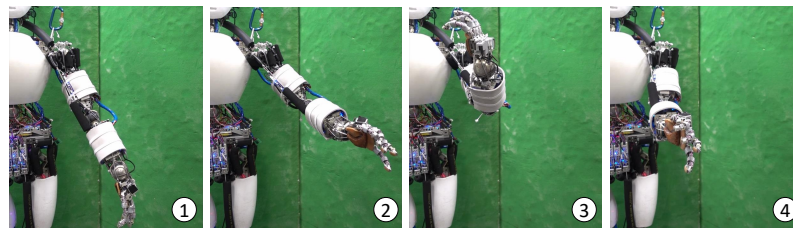


Fig. 5.21: The experiment of basic movements with and without muscle relaxation control [189].

まず、筋弛緩制御を入れた時の ΔI_{relax} の遷移を Fig. 5.22 に示す。静止した際には MRC が働いて ΔI_{relax} が上昇していき、動作している際には ΔI_{relax} が減少していく様子がわかる。 ΔI_{relax} が大きく変化する筋は動作によって異なり、現在最も重要でないと考えられる筋に対して MRC が働いている。

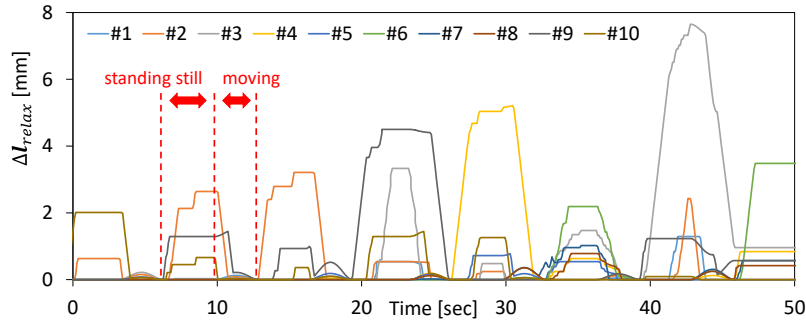


Fig. 5.22: The transition of Δl_{relax} during basic movements with muscle relaxation control [189].

次に、筋弛緩制御を入れた際と入れなかった際における $\|f\|_2$ の遷移の違いを Fig. 5.23 の上図に示す。筋弛緩制御を入れた場合は、入れなかった場合に比べて静止時に大きく筋張力が下がっていることがわかる。これは、モデル誤差や摩擦による無駄な筋内力が解放されたためであると考えられる。

最後に、筋弛緩制御を入れた際と入れなかった際における θ の遷移を Fig. 5.23 の下図に示す。両者に大きな違いはなく、筋弛緩制御を入れることによって関節角度の追従性はほとんど変わらないことがわかる。よって、このような基本動作の際には、タスクに影響を与えずに、筋内力を大きく減らすことが可能である。

重量物体把持実験

ダンベルのような重量物体を把持した際の筋弛緩制御の働きについて考察する。Fig. 5.24 のように、約 3.6 kg のダンベルを持ち上げ、その状態で静止する。このときの筋弛緩制御を入れた場合と入れない場合における $\|f\|_2$ の遷移を Fig. 5.25 の上図に、最も大きな筋張力を出していた 2 本の主動筋の温度遷移を Fig. 5.25 の下図に示す。筋弛緩制御を入れない場合に比べて、入れた場合は大きく筋張力を抑制できていることがわかる。ここで、筋弛緩制御を入れない場合でも徐々に筋張力が下がっているのは、[125] における安全機構が、温度が 60 度を越えたところで動作し始めているためである。筋弛緩制御を入れた場合は温度が 60 度を越えないため、ほぼ一定の低い筋張力を発揮している。筋弛緩制御を入れない場合は筋温度が急上昇し、80 度を越えたところで動作を終了している。

筋弛緩制御を入れた方が筋張力を抑制し継続的に動作ができるものの、大きな力が加わる場合は関節角度の誤差を $\Delta\theta^{max}$ 限界まで許容するため、大きく関節角度誤差が発生していることがわかる。この $\Delta\theta^{max}$ を小さく設定することで関節角度誤差は小さくできるものの、筋張力抑制の効果は薄れてしまい、ここに本制御のトレードオフが存在していることがわかった。

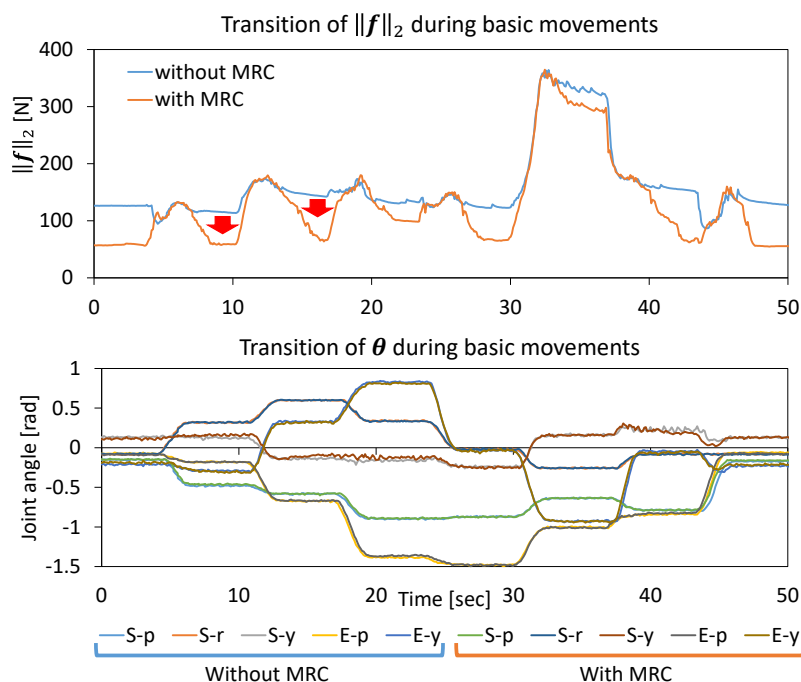


Fig. 5.23: The transition of $\|f\|_2$ and θ during basic movements with and without muscle relaxation control [189].

机上での上腕接触実験

テーブルに手を置いて休む動作の際の筋弛緩制御の働きについて考察する。重量物体を運ぶ際に、一度テーブルの上に手を置いて身体を休め、また動作を再開する等の動作を想定している。Fig. 5.26のように、約 5 kg の箱を抱え、手をテーブルの上におろして身体を休める。この際、認識やモデル誤差等から、環境とロボットの間で内力が働きすぎず、身体をテーブルに預ける動作が正確にできるとは限らない。本実験では、 $S-p$ と $E-p$ のみを動作させ、身体を休める際に $(\theta_{S-p}, \theta_{E-p})$ がそれぞれ、1: (-45, -80), 2: (-45, -85), 3: (-45, -90), 4: (-50, -90), 5: (-50, -95) になるように動作させた。これは、Fig. 5.27 の下図のように、休めたい身体部位とテーブルという環境の距離が 1 から 5 に進むにつれて、めり込んだ状態から離れる状態へと移っていくような遷移である。

筋弛緩制御を入れた場合と入れない場合における、1 から 5 の状態の $\|f\|_2$ を Fig. 5.27 の上図に示す。筋弛緩制御を入れた場合は入れない場合に比べて、無駄な拮抗筋による内力を抑制することができているため、より筋張力の値が小さくなっていることがわかる。また、両者とも 2 の場合に最も筋張力が小さく、1 のように身体がテーブルにめり込むような姿勢を指令すると筋張力が高まる。そして、5 のように身体が環境から離れていくと、身体を環境に預けることができずに筋張力が増加していく。

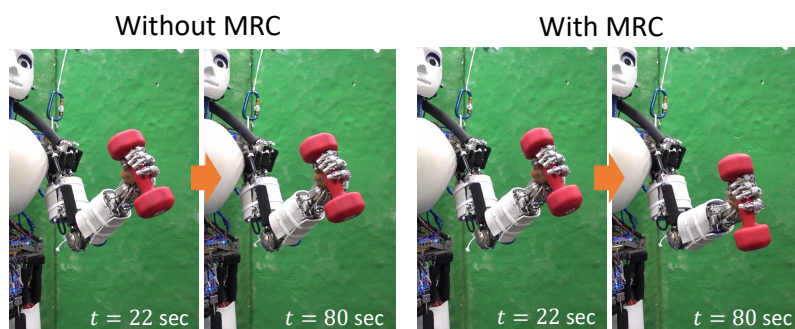


Fig. 5.24: The experiment of grasping a heavy dumbbell [189].

また, 筋弛緩制御を入れた場合と入れない場合における $\|f\|_2$ の差に着目する. その差は 2 や 3 のときに最も小さく, 1 や 5 のときに増加する傾向にある. これは, 筋弛緩制御を入れない場合は身体を上手く環境に預けられるかどうかが敏感に変化するのに対して, 筋弛緩制御を入れることによって, 身体と環境の間の誤差を吸収して, 身体を上手く環境に預けることが可能となるからであると考えられる. 身体が環境にめり込むような姿勢を送った場合には, 主動筋を弛緩させても身体姿勢に変化が起きないため, 最大限に身体を環境に預けることができる. 逆に, 身体が環境から離れる方向に誤差がある場合には, $\Delta\theta^{max}$ の分だけなるべく環境に身体を預けるような動作が生まれるのである.

ハンドル操作実験

ハンドル操作の際の筋弛緩制御の働きについて考察する. ハンドルは環境に固定され, 回転方向のみの動きを持っている. ハンドルに対して逆運動学を解き, Fig. 5.28 のように双腕で -45 度から 45 度まで回転させる操作を 5 回行う. この際, 45 度回転させる動作は 5 sec かけて行い, ハンドルが 0 deg のときは 5 sec, 45 または -45 deg のときは 3 sec 静止する.

筋弛緩制御を入れた場合と入れない場合における, $\|f\|_2$ の遷移を Fig. 5.29 の上図に示す. 筋弛緩制御を入れた場合は入れない場合に比べて, 筋張力を抑制することができていることがわかる. また, $\|c\|_2$ の遷移を Fig. 5.29 の下図に示す. 筋弛緩制御を入れることで, 筋温度を抑制し, 継続的なハンドル操作を行うことが可能となる.

ハンドル操作は常にハンドルと身体が拘束されるため, 拮抗筋だけでなく主動筋まで弛緩させることが可能である. 実際の運転の際も常に身体を動かしているとは限らず, ほとんどの場合はハンドルに手をかけただけの状態である. このように, ハンドル操作をはじめとして, つり革を掴んだりキーボードを触るときに前腕を机に接触させたり, 多くの動作において本制御が有効であると考えられる.

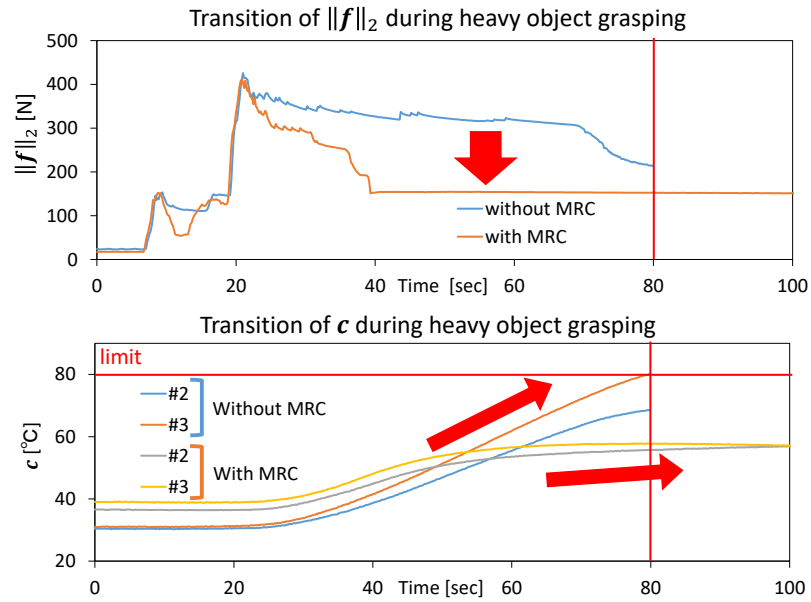


Fig. 5.25: The transition of $\|f\|_2$ and c while grasping a heavy dumbbell with and without muscle relaxation control [189].

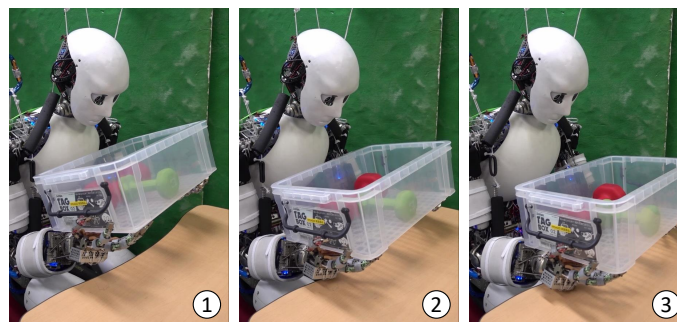


Fig. 5.26: The experiment of resting the arms on the desk while carrying a heavy box [189].

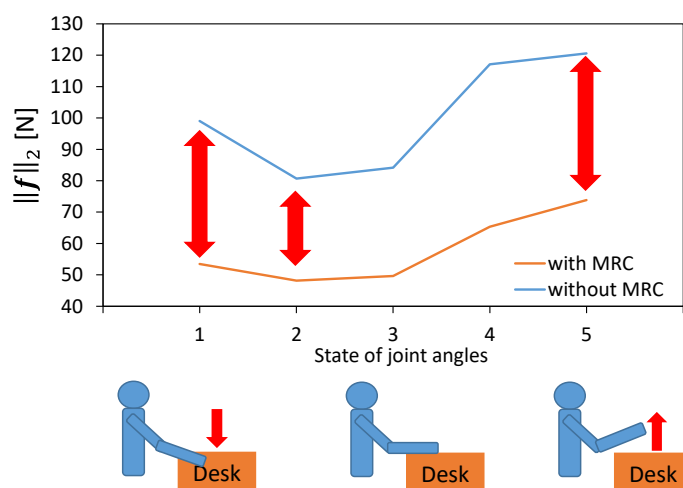


Fig. 5.27: $\|f\|_2$ while resting the arms on the desk with and without muscle relaxation control, regarding each joint angle state 1 – 5 [189].

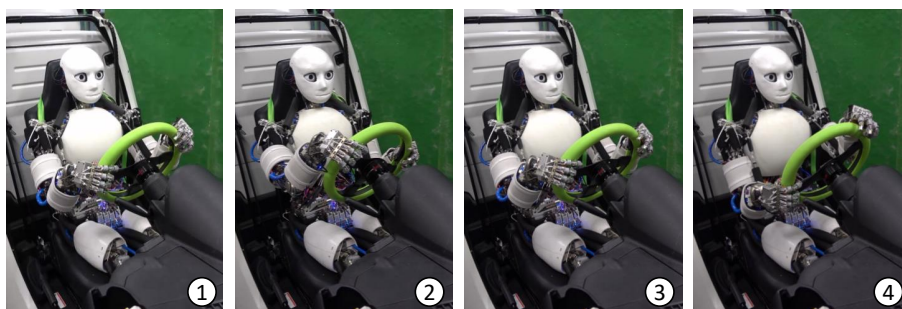


Fig. 5.28: The experiment of handle operation [189].

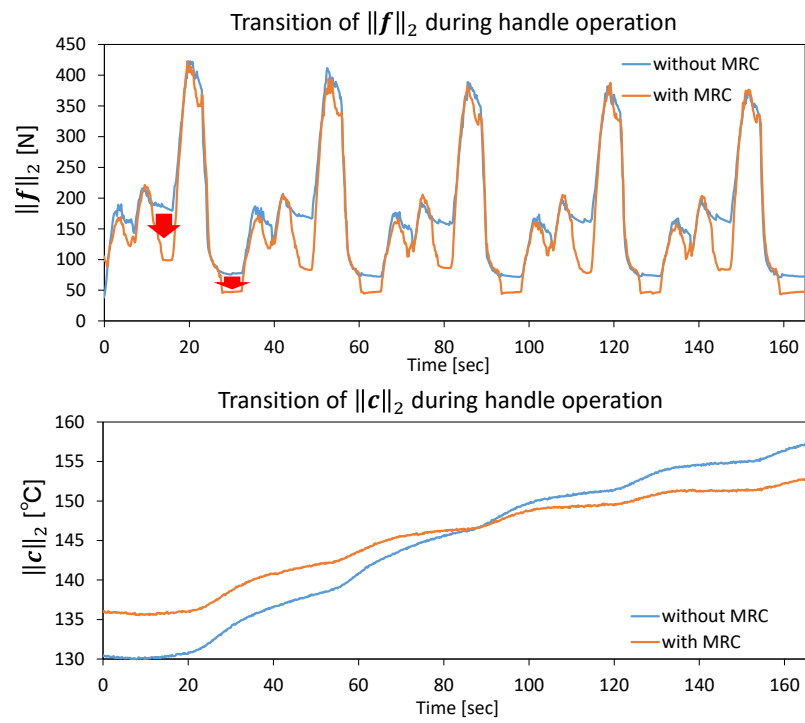


Fig. 5.29: The transition of $\|f\|_2$ and $\|c\|_2$ during handle operation with and without muscle relaxation control [189].

5.4 速度最大化反射戦略

5.4.1 概要と先行研究

筋骨格ヒューマノイドは様々な生物規範型構造の利点を持つが、その中でも最も重要な特徴の一つが冗長な筋配置である。これにより、筋が一本切れても動き続ける冗長駆動 [125] や、非線形弾性要素と合わせた可変剛性制御 [125, 108] が可能となる。この冗長な筋の拮抗配置はこれら多くの利点が存在する一方、モデル化の誤差によって、拮抗筋間で大きな内力や緩みが発生するという問題があった。この問題に対して、これまで拮抗筋抑制制御 [167] や拮抗修正制御 [168]、筋弛緩制御 [189] 等による内力・緩みの緩和が行われてきている。

そして本節は、新たに発見したもう一つの問題点を扱うものである。これは、冗長な多数の筋の関節に対する速度が様々であるため、最も速度の遅い筋に関節角速度が規定されてしまうという現象である。本節ではこの問題を解決し、規定された最大関節角速度よりも速い速度を出すための手法提案を行う [169]。これは言い換えれば、問題点ではなく、これまで最大と思われていた規定された関節角速度よりも速い速度を出すことができるという、利点と考えることもできる。

これまで関節角速度を速くするためには、[192] のようなソフトウェアにおける最適化が行われてきた。しかし、これらは拮抗腱駆動のハードウェアの特性を活かすことはできない。また、可変剛性機能を持つ軸駆動型ロボットにおいては、その弾性変化を用いた速度最大化が成されている [193, 194]。本節では、冗長な筋を有する筋骨格構造の特性を用いて、最大関節角速度を超えるシンプルな手法を提案する。

5.4.2 冗長腱駆動構造の問題点

速い関節角速度を出す動作として、手を振り下ろす動作を考えてみる。例として、机を叩いたり、ハンマーを振り下ろしたりするような動作を考えてもらえると良い。このとき、手を振り上げたとき関節角度 θ^{start} と振り下ろす目標の関節角度 θ^{end} を決める。手を振り下ろしている最中の現在の関節角度を θ 、このときの関節角速度 (現在関節角度から目標関節角度方向へのベクトル) を $\dot{\theta}$ とすると、現在の動作方向に関する筋のモーメントアーム \mathbf{r} は以下ようになる。

$$\mathbf{r}(\theta) = G(\theta)\dot{\theta}/\|\dot{\theta}\|_2 \quad (5.27)$$

ここで、 $\|\cdot\|_2$ は L2 ノルムを表す。 \mathbf{r} が正なら拮抗筋、 \mathbf{r} が負なら主動筋を表す。この \mathbf{r} の絶対値はモーメントアームであり、複数の関節を跨ぐ多関節筋や関節から筋の位置までが遠い筋では大きな値

となる。そして、このモーメントアームが大きいほど同一の関節角速度でも筋長速度が大きくなり、アクチュエータの限界である最大筋長速度に達しやすい。実際に最大筋長速度に達するかどうかは、筋アクチュエータごとの最大筋長速度も考慮する必要がある。よって、以下の指標 q が大きな筋ほど最大筋長速度に達しやすい。

$$q(\theta) = r(\theta) \circ \dot{l}^{limit} \quad (5.28)$$

ここで、 \circ はベクトルの要素ごとの除算、 \dot{l}^{limit} は最大筋長速度を表す。

また、これらの最大筋長速度に達しやすい筋には、モーメントアームが大きい・最大筋長速度が小さいということ以外にも特徴がある。速い関節角速度を出すような動きを考えると、例えば手を振り下ろす、サッカーボールを蹴る等、重力の影響を使ったものが多い。つまり、アクチュエータの速度だけでなく、重力や身体の慣性を活かして動作させている。ゆえに、それらの動作は上から下へ動く、つまり、Fig. 5.30 のような動作がほとんどである（これは肩と肘の動きとしても、股と膝の動きとしても取れる）。このとき、筋のモーメントアームを見てみると、どの動作においても、動作を阻害する方向の筋、つまり拮抗筋のモーメントアームが主動筋よりも大きくなっていることがわかる。これは、拮抗筋では筋が骨格間を張っているのに対して、主動筋では筋が関節を張っているためである。ゆえに、最大筋長速度が同じ場合、これらの動作では主に拮抗筋の方が遅いため、主動筋がどんなに頑張っても、その動きが阻害されて、速度が制限されてしまうのである。もしここで拮抗筋が働かなければ、より速い速度で主動筋が関節を動かすだけでなく、重力や慣性の力によって、主動筋の速度よりも速く関節が動く可能性がある。また、主動筋は一部が遅くても関節の動作を阻害することはない、拮抗筋が関節角速度を制限する主な理由であることがわかる。つまり、Fig. 5.30 のような動作でなくとも、基本的には拮抗筋が関節角速度を阻害する主な理由となる。

よって、値 q が大きな拮抗筋を何らかの方法で最大筋長速度まで達しないように管理することが、本手法の主眼である。

5.4.3 速度最大化反射システム

5.4.2 節の問題を解決するために、本節では二つの手法について考案・実験・考察をしていく。一つ目は拮抗筋抑制反射であり、モーメントアームの大きな拮抗筋を抑制、つまり流れる電流を 0 とすることで、バックドライバビリティを使って筋を出すという手法である。二つ目は拮抗筋予見伸長反射であり、モーメントアームの大きな拮抗筋を予め緩めておくという手法である。

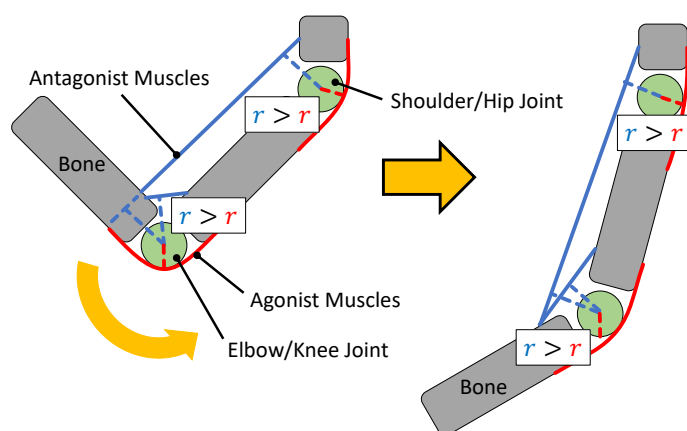


Fig. 5.30: Characteristics of movements with high joint velocity [169]. These movements use the effect of gravity and body inertia.

拮抗筋抑制反射

最大筋長速度まで達しないように管理するのではなく、逆にそれを管理せずに出力を0にしてしまうのが拮抗筋抑制反射である。この手法は非常に単純であり、動作開始すぐに、以下のように q が大きな筋の出力を0にしてしまえば良い。

$$o_i = 0 \quad \text{if } q_i > C \quad (5.29)$$

ここで、 i は着目する筋の番号、 o は電流値、 C は定数である。 C を0とすると、拮抗筋のみの電流値が0となり、 $C > 0$ とすると、拮抗筋の中でもモーメントアームが大きな筋のみの電流値が0となる。もしその筋にバックドライバビリティがあれば、筋は引っ張られることで自然と出ていき、それらの筋には最大筋長速度が存在しなくなる。 q の大きさは動作中に徐々に変化していくため、この制御は常に低レイヤで走り続ける。この手法はバックドライバビリティがあることが前提となるが、これは実機実験において検証する。

拮抗筋予見伸長反射

モーメントアームの大きな拮抗筋を予め緩めることで、確実に最大筋長速度まで達しないように管理するのが拮抗筋予見伸長反射である。どの筋を予め緩めるかを考える際に、以下の2つの点を考慮する必要がある。

まず、手や足を振り上げるときに、その重力に逆らった姿勢が実現可能でなければならない。動作方向、つまり手や足を振り下げる動作の拮抗筋は振り上げる動作の主動筋であり、これらを単純に緩め

てしまつては、その姿勢を実現することができない。つまり、予め緩める筋を0、緩めない筋を1としたマスクベクトル \mathbf{m} を考えたときに、以下の二次計画法において解がある、つまり姿勢 θ^{start} を実現できることを保証しなければならない。

$$\begin{aligned} & \underset{\mathbf{f}}{\text{minimize}} && (\mathbf{m} \otimes \mathbf{f})^T W_1 (\mathbf{m} \otimes \mathbf{f}) \\ & \text{subject to} && \tau^{nec} = -G^T(\theta^{start})(\mathbf{m} \otimes \mathbf{f}) \\ & && \mathbf{m} \otimes \mathbf{f}^{min} \leq \mathbf{m} \otimes \mathbf{f} \leq \mathbf{m} \otimes \mathbf{f}^{max} \end{aligned} \quad (5.30)$$

ここで、 \otimes はベクトルの要素ごとの乗算、 \mathbf{f} は計算される筋張力、 W_1 は重み行列 (本節では単位行列とする)、 τ^{nec} は θ^{start} を実現するために必要な関節トルク、 $\mathbf{f}^{\{min, max\}}$ は筋張力の最小値・最大値を表す。これを満たす筋張力 \mathbf{f} が計算可能であれば、 θ^{start} は実現可能であり、 θ^{start} のまま \mathbf{m} で指定した筋を緩ませることが可能である。これは筋の冗長性ゆえに可能となる原理である。筋を緩めることで他の筋に対する張力が増えてしまうが、少しの時間であれば問題ないと考え、このような動作戦略を取った。

次に、当然その拮抗筋を緩めることで、関節角速度が速くならなければならない。 \mathbf{m} を決め、以下のようなシミュレーションを行うことで、目標関節角度まで移動する時間を概算することができる。

$$\begin{aligned} & \underset{\Delta\theta}{\text{minimize}} && (\theta^{end} - \theta - \Delta\theta)^T W_2 (\theta^{end} - \theta - \Delta\theta) \\ & \text{subject to} && -\mathbf{m} \otimes \dot{\mathbf{l}}^{limit} \Delta t \leq \mathbf{m} \otimes (G(\theta)\Delta\theta) \leq \mathbf{m} \otimes \dot{\mathbf{l}}^{limit} \Delta t \end{aligned} \quad (5.31)$$

ここで、 $\Delta\theta$ は現在関節角度 θ からの予想される変位、 W_2 は重み行列 (本節では単位行列とする)、 t は現在時刻、 Δt はシミュレーションの間隔を表す。これにより、 θ から Δt 秒間で最大どの程度 θ^{end} に近づけるかを表す $\Delta\theta$ を計算することができる。このシミュレーションを $\theta = \theta^{start}$ からはじめ、 $\theta \leftarrow \theta + \Delta\theta$ 、 $t \leftarrow t + \Delta t$ というように更新していく。 $\|\theta - \theta^{end}\|_2 < \epsilon$ となったところでシミュレーションを終了し、そのときの t がかった時間 t^{cost} となる。この t^{cost} が小さくなるような \mathbf{m} を求める必要がある。実際にはロボット・モータの慣性や筋経路のモデル誤差、摩擦等が多く存在するため、この計算はあくまで簡易的な概算であるが、それでも大まかな特性を知ることが可能である。軸駆動型ロボットと違い筋骨格ヒューマノイドはその複雑な構造ゆえに詳細なモデル化が難しいため、このようなシンプルな方法を取った。

上記の θ^{start} を実現できる筋張力が存在し、 t^{cost} が小さいような \mathbf{m} を探索していく。全探索を行っても良いが、 \mathbf{q} が大きな筋を抜くことで t^{cost} が下がることは明白である。そのため、 \mathbf{q} が大きな筋から順に \mathbf{m} の値を0にしていき、 θ^{start} を実現できる \mathbf{f} が存在しなくなったところで探索をやめる、と

いう手法を取る.

最後に, m の値が 0 となった筋をどの程度伸ばす必要があるかを計算する. 最終的に求めた m によって先のシミュレーションを行って $\Delta\theta$ の遷移を得る. この遷移から $G(\theta)\Delta\theta$ によって筋長の遷移が計算でき, 常に最大筋長速度で筋を伸ばした時の筋長遷移との差分の最大値が, 伸ばすべき最小の筋長変化 Δl^{pred} となる. θ^{start} の時点で Δl^{pred} だけ筋長を伸ばすことで, 拮抗筋が主動筋を阻害せず, より速い動作が可能となる.

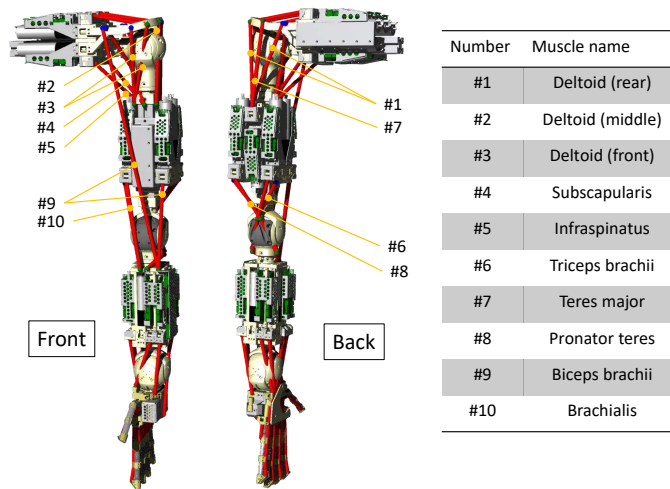


Fig. 5.31: The muscle arrangement of the left arm of Musashi used in this study [169].

5.4.4 実験

実験セットアップ

本節では, 筋骨格ヒューマノイド Musashi [87] の左腕を用いる. その筋配置はこれまでとは多少異なり, Fig. 5.31 のようになっている. 本節では主に肩と肘の 5 自由度を用い, それらに関係する筋は 10 本, うち二関節筋が 1 本含まれている. これらの関節角度を S-p, S-r, S-y, E-p, E-y と表す (S は肩関節, E は肘関節, rpy は roll, pitch, yaw を表す). 全筋アクチュエータ [152] のモータは 90W, ギア比 29:1 の Maxon BLDC Motor であり, 最大筋長速度の特性は一致している. ただし, [152] における目標筋長と現在筋長の差分に対して電流を流すような制御では Fig. 5.32 に示すように最大筋長速度に一瞬で到達するわけではない. これは, モータドライバを小さくするために 1 シャント方式を採用しており, 出力が振動するためゲインを上げられないことが原因であると考えられる. そのため, Eq. 5.31 における $-j_{limit}$ と j_{limit} を j_{min} と j_{max} に置き換え, 現在の筋長が $j > 0$ のとき, 以下のようにそれらを更新

して用いる.

$$\dot{l}^{max} = \min(\dot{l} + \alpha \Delta t, \dot{l}^{limit}) \quad (5.32)$$

$$\dot{l}^{min} = \min(\dot{l} - \alpha \Delta t, 0) \quad (5.33)$$

ここで, α は定数であり, 本節では, Fig. 5.32 から $\alpha = 0.46 \text{ [m/s}^2]$, $\dot{l}^{limit} = 0.30 \text{ [m/s]}$ と同定された. これは, 現在筋長速度を現在から上げる場合は時間に比例して徐々に速度が上がるのに対して, 下がる場合は一気に 0 まで下げることができる, という制約である. $\dot{l} < 0$ の場合も同様で, より速いマイナス方向の速度を出すには制限がかかるが, 速度を 0 まで戻すことに対して制限はない.

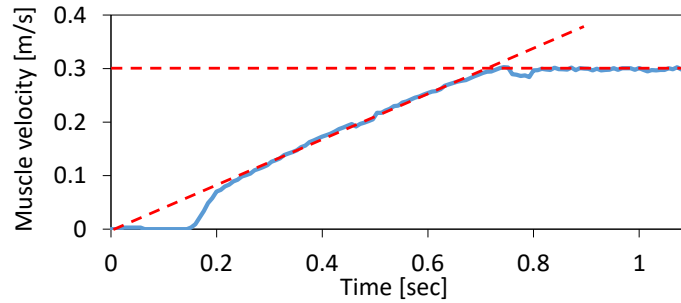


Fig. 5.32: Characteristics of muscle length velocity transition [169].

本節では Musashi が左手を振り下ろすような動作について扱う. シミュレータ, 実機における動きを Fig. 5.33 に示す. このときの関節角度, 筋長の速度等を記録していく. 通常の筋骨格ヒューマノイドは肩甲骨や球関節等の複雑な関節により関節角度を測定することができないが, 本節で扱う Musashi は関節モジュール内に角度センサを有するため, これを測定することが可能である. また, 筋長は筋アクチュエータについたエンコーダの値を用いている. 本節のパラメータとして, $C = 0$, $\Delta t = 0.03$ とする.

基本実験

提案手法を検証する前に, シミュレーション・実機において本動作ではどの程度の関節角速度・筋長速度が出るのかを確認する. まず, 拮抗筋予見伸長反射の後半で説明した方法を用いてシミュレーションを行う. このとき, マスク m は全ての要素が 1 のベクトルとする. そのときの関節角速度遷移を Fig. 5.34 に示す. 最大関節角速度は E-p の 2.4 rad/s であり, 収束までに 0.99 秒かかった.

次に実機において同じ動作を行った際の関節角速度・筋長速度・筋張力の遷移を Fig. 5.35 に示す. 最大関節角速度は E-p の 2.6 [rad/s] であり, シミュレーションと似た結果が得られた. 筋長速度は主に, 拮抗筋である #9 の上腕二頭筋, #10 の上腕筋の速度が上限である \dot{l}^{limit} まで達していることがわか

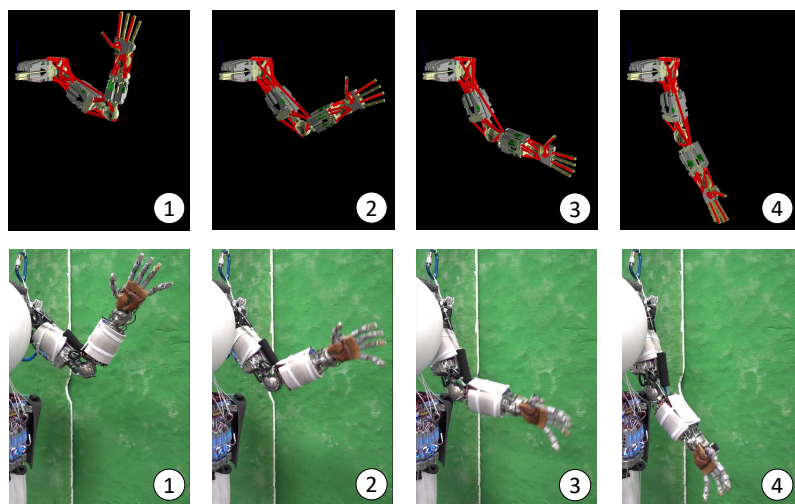


Fig. 5.33: Experimental motion of simulation and actual robot [169].

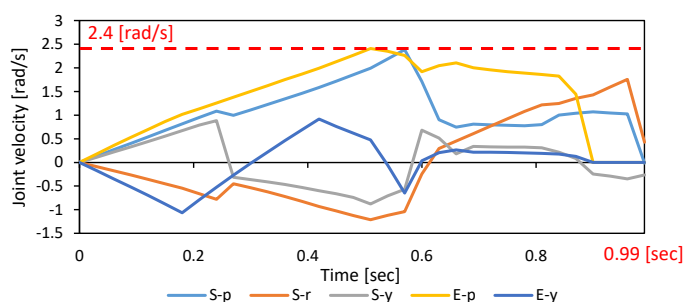


Fig. 5.34: Transition of joint angle velocity in simulation, without any proposed controls [169].

る。また、このときの筋張力は最大で約 290 N であり、主に#1 の肩の主動筋、#6 の肘の主動筋に大きな負荷がかかっている。拮抗筋としては、#9 の多関節筋である上腕二頭筋に約 50 N の力がかかり続けている。なお、3 回動作を行った際にセンサ値の遷移に違いはほとんど見られなかった。

拮抗筋抑制反射実験

拮抗筋抑制反射をシミュレーション・実機において検証する。まずシミュレーションでは、ロボットにバックドライバビリティがあると仮定し、Eq. 5.29 のように $q_i > C$ となる筋 (#2, #3, #9, #10) のマスク m を 0 として動作を行った。そのときの関節角速度遷移を Fig. 5.36 に示す。最大関節角速度は E-p の 4.4 rad/s であり、収束までに 0.6 秒かかった。

次に実機において同じ動作を行った際の関節角速度・筋長速度・筋張力の遷移を Fig. 5.37 に示す。最大関節角速度は E-p の 3.7 [rad/s] であり、シミュレーションよりは小さめの関節角速度が得られた。

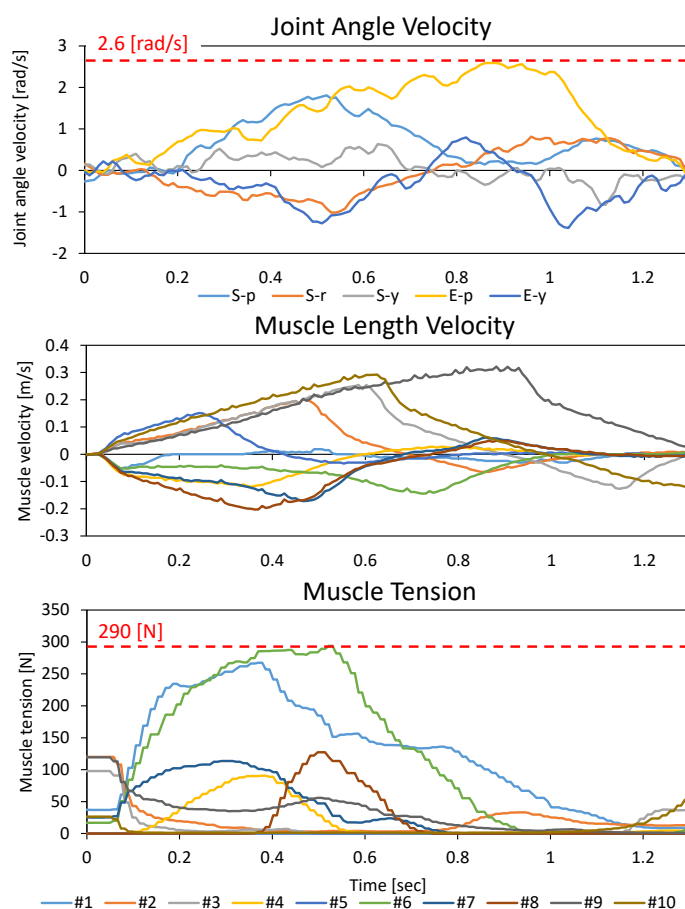


Fig. 5.35: Transition of joint angle velocity, muscle length velocity, and muscle tension in the actual robot, without any proposed controls [169].

筋長速度は主に、拮抗筋である#9の上腕二頭筋、#10の上腕筋の速度が上限である \dot{l}_{limit} 以上の速度を出していることがわかる。また、このときの筋張力は最大で約 180 N であり、主に#1の肩の主動筋、#6の肘の主動筋に大きな負荷がかかっている。拮抗筋にはほとんど力がかかっていない。なお、3回動作を行った際にセンサ値の遷移に違いはほとんど見られなかった。

拮抗筋予見伸長反射実験

拮抗筋予見伸長反射をシミュレーション・実機において検証する。まずシミュレーションにおいて、条件を満たすマスク m を計算する。その結果、モーメントアームが大きな筋は順に#9、#10、#3となるが、#9と#10の両者を伸ばしてしまうと、肘のトルクを確保できないため、#9のみがマスクされた。そのときの関節角速度遷移を Fig. 5.38 に示す。最大関節角速度は E-p の 3.3 rad/s であり、収束までに

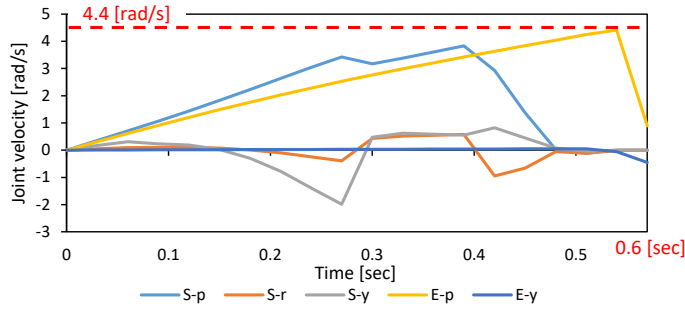


Fig. 5.36: Transition of joint angle velocity when using a method inhibiting antagonist muscles in simulation [169].

0.78 秒かかった。

次に実機において同じ動作を行った際の関節角速度・筋長速度・筋張力の遷移を Fig. 5.39 に示す。最大関節角速度は E-p の 3.4 [rad/s] であり、シミュレーションとほぼ同じ関節角速度が得られた。筋長速度は主に、拮抗筋である#3 の三角筋前部, #10 の上腕筋の速度が上限である \dot{l}_{limit} 付近に達していることがわかる。また、このときの筋張力は最大で約 140 N であり、主に#6, #8 の肘の主動筋に大きな負荷がかかっている。拮抗筋としては、#10 の上腕筋に約 60 N の力がかかり続けている。なお、3 回動作を行った際にセンサ値の遷移に違いはほとんど見られなかった。

Table 5.1: Comparison among the basic motion (**Basic**), the motion when using the method inhibiting antagonist muscles (**Method-1**), and the motion when using the method elongating antagonist muscles (**Method-2**) [169].

	Basic	Method-1	Method-2
Maximum $\dot{\theta}$ (simulation) [rad/s]	2.4	4.4	3.3
Maximum $\dot{\theta}$ (actual robot) [rad/s]	2.6	3.7	3.4
Maximum \dot{l} [m/s]	$\leq \dot{l}_{limit}$	$> \dot{l}_{limit}$	$\leq \dot{l}_{limit}$
Maximum T [N]	290	180	140

5.4.5 議論

通常の動作 (**Basic**) と拮抗筋抑制反射 (**Method-1**), 拮抗筋予見伸長反射 (**Method-2**) を用いたときの動作の比較を Table 5.1 に示す。まず、シミュレーションから理論的な最大関節角速度の大きさは **Basic** < **Method-2** < **Method-1** であることがわかった。また実機実験より、その結果は実機においても同様であることが示された。よって、本手法は関節角速度最大化において有効である。しかし、**Method-1** においては、シミュレーションと実機で大きな誤差がある。これは、**Method-1** がバックドライバビリ

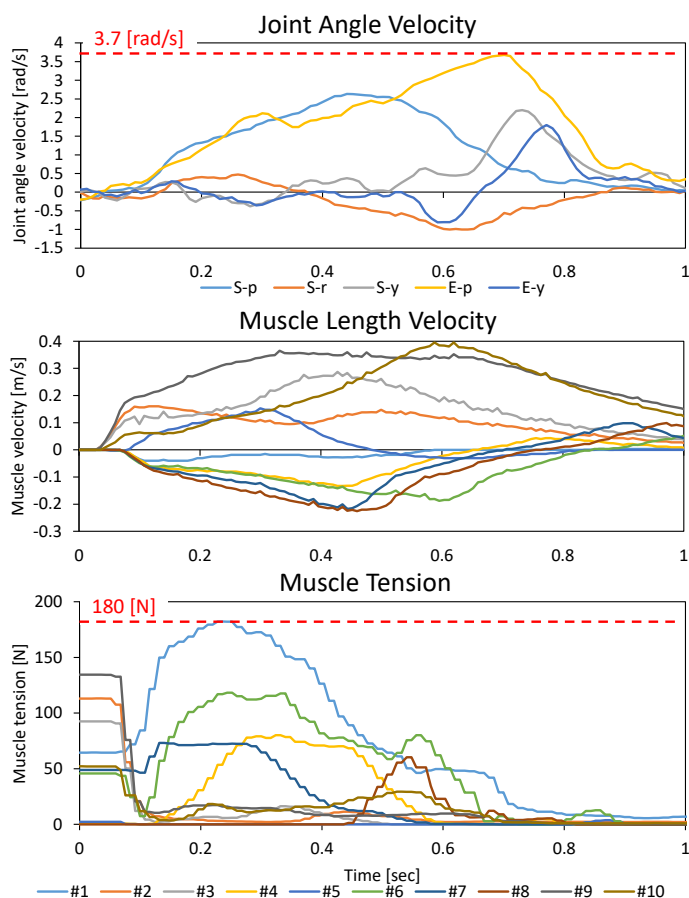


Fig. 5.37: Transition of joint angle velocity, muscle length velocity, and muscle tension when using a method inhibiting antagonist muscles in the actual robot [169].

ティを前提とした手法であることが理由として挙げられる。そのため、現在のアクチュエータのギア比は 29:1 であるが、これをより高くしてしまうと、関節角速度はさらに落ちる可能性がある。次に、筋長速度の違いについて考察する。**Method-1**では拮抗筋はバックドライバビリティによって勝手に伸びるため、 i_{limit} よりも大きな速度が出ていることがわかる。それに対して、**Method-2**は最初から拮抗筋を伸ばしてしまうため、大きな筋長速度を出さずに済んでいることがわかる。最後に、筋張力について考察する。**Basic**では強い拮抗によって大きな力が発揮されているのに対して、**Method-1**や**Method-2**では半分程度の筋張力しか発揮されていない。よって、拮抗筋を抑制、または予め伸ばすことで、関節角速度最大化だけでなく、筋張力の削減にも効果があると考えられる。

まとめると、**Method-1**はバックドライバビリティが高いならば有効であるが、バックドライバビリティが低い場合は**Method-2**よりも性能が悪くなる可能性がある。これに対して、**Method-2**は**Method-**

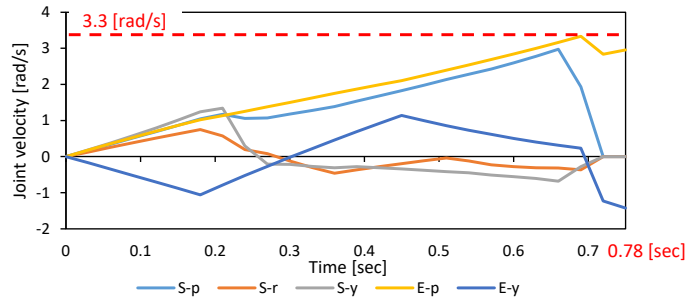


Fig. 5.38: Transition of joint angle velocity when using a method elongating antagonist muscles in simulation [169].

1 に比べて性能が劣ることがある反面、バックドライバビリティに依存せず有効である。しかし、予め筋を伸ばすため、関節角度が θ^{start} のときに高い筋張力が必要になってしまうことがある。

本節では非常にシンプルな手法によってより大きな関節角速度を出す手法を開発した。しかし、ロボットの摩擦やヒステリシス等の動的な要素や、より良いモデル化によって、より詳しい解析ができる可能性がある。また今後、速い動きをしたときの、正確な関節角度遷移の実現も必要になっていくと考える。

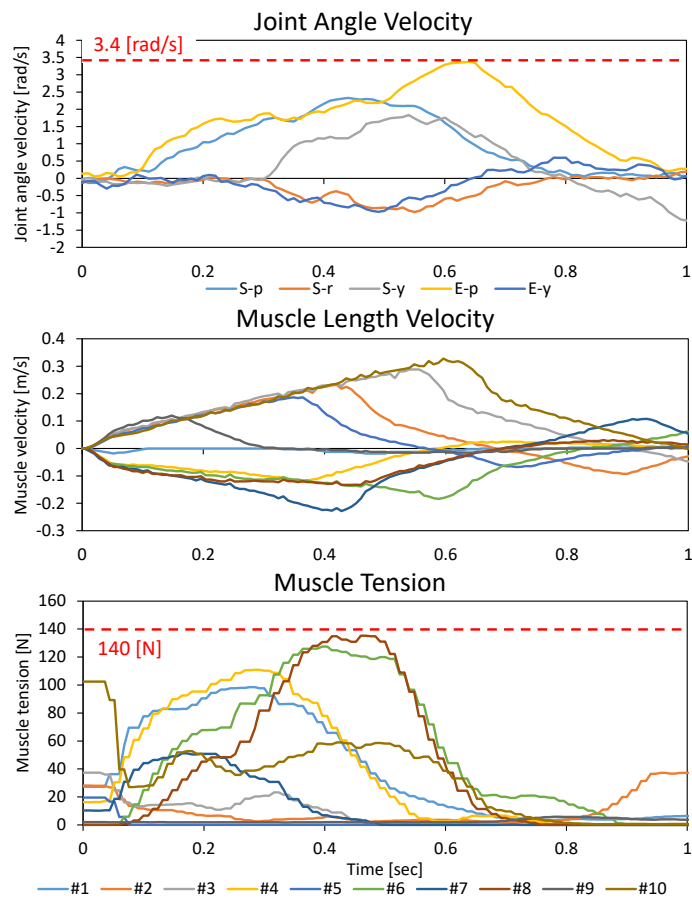


Fig. 5.39: Transition of joint angle velocity, muscle length velocity, and muscle tension when using a method elongating antagonist muscles in the actual robot [169].

5.5 拮抗筋抑制制御

5.5.1 概要と先行研究

モデル化の難しい筋骨格腱駆動ヒューマノイドにおいては、幾何モデルと実機の間には大きな差異が生じる。この解決のために様々な学習手法が開発されているが [122, 125], 誤差を完全に無くすることはできない。また, Fig. 5.40 のように肩甲骨と肩を用いて腕を真上まで挙げるような、誤差の生じやすい広可動域動作を実現した研究は皆無である。

筋骨格腱駆動ロボットの制御には、単純にモデルの筋長を与えるものから、筋剛性制御 [121], puller-follower 制御 [195], 筋張力制御 [123] 等が存在する。しかし、そのどれも幾何モデルの正しさを前提にしているがゆえ、広可動域を動作させようとするとき実機との誤差によって筋張力が高まってしまうか、到達したい位置まで到達することができない。そこで、人体構造を模した筋骨格腱駆動ヒューマノイドを内力を溜めずにスムーズに動作させるために、人体の反射に着目した。人体の反射には伸長反射や腱反射、相反性神経支配などが存在するが、その中でも、相反性神経支配という主動筋に対して拮抗筋を抑制する機構によって人は体を柔軟に動かしている。この拮抗筋抑制の考え方をうまくヒューマノイドに適用することで、広可動域動作を実現できると考えた。また, [122] の関節角度推定を用いることでエンコーダやモーションキャプチャを用いないシステムを目指した。この推定手法は複雑な肩関節や肩甲骨に関して良い精度は出ないものの、この関節角度推定を拮抗関係の判定のみに使うことによってその誤差を許容している。

本節では、実機とモデルの誤差によって内力が溜まり大きな動作の出来なかった筋骨格腱駆動ヒューマノイドの上肢において、人体の相反性神経支配を模した拮抗筋抑制制御を適用するによって、非常に単純なシステムで内力を溜めずに広可動域動作を実現できることを示す [167]。そして、この制御法を他の制御方式と比較し、いくつかの議論を行う。なお、本節の拮抗筋抑制制御は 5.4 節における拮抗筋抑制反射とは目的が異なる。

5.5.2 拮抗筋抑制制御システム

拮抗筋抑制制御システム

拮抗筋抑制制御 (Antagonist Inhibition Control, AIC) は、筋剛性制御の剛性係数を主動筋と拮抗筋で変化させるというシンプルな制御である。システム構成を Fig. 5.41 に示す。

(1) は筋長変化と Extended Kalman Filter (EKF) [122] によって関節角度を推定する。

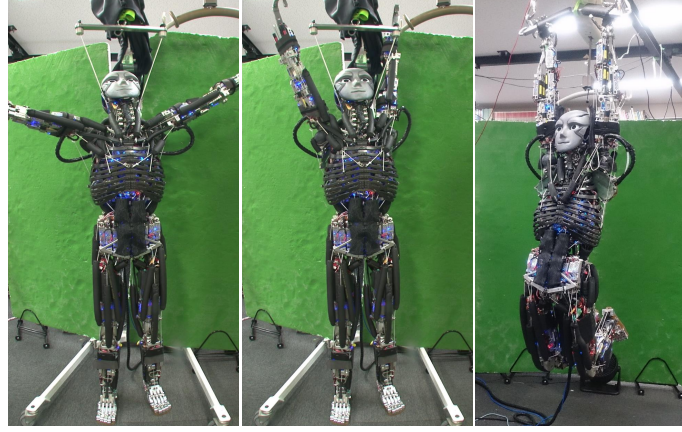


Fig. 5.40: The wide range limb motions of Kengoro [167].

(2) は本手法で最も重要なコントローラである。人間の相反性神経支配のように、拮抗筋の力を抑制することを考える。筋長ヤコビアン $G(\theta)$ は、関節角度が θ の際に、ある方向に関節を動かした際にどれだけ筋が縮むかを表す ($M \times N$) の行列である。つまり、 $G(\theta)(\theta^{ref} - \theta)$ はそれぞれの筋がその方向に動くときに主動筋となるか、拮抗筋となるかを表現する。本手法では筋剛性制御 [121] の筋剛性 k_{stiff} を主動筋か拮抗筋かによって変更する。 k_{stiff} は 0 であればその筋の張力は f_{bias} で一定となり、 k_{stiff} が正であれば指令筋長 l^{ref} に追従する方向に力を発生させる。よってこれはまさに拮抗筋と主動筋であり、Antagonist Inhibition Controller は i 番目の筋に関して以下のような決定をする。

$$s = G(\theta) \frac{\theta^{ref} - \theta}{\|\theta^{ref} - \theta\|_2} [i] \quad (5.34)$$

$$k_{stiff}[i] = k \quad \text{if} \quad s < C \quad (5.35)$$

$$k_{stiff}[i] = 0 \quad \text{if} \quad s \geq C \quad (5.36)$$

となる (ここで s は $\|\theta^{ref} - \theta\|_2$ で割ることでモーメントアームを表すようになり、 k は主動筋の k_{stiff} に与える定数、 C は拮抗関係を決める閾値である)。動作を安定させるために、 k_{stiff} の値は 0 から k に、そして k から 0 に、 t_k ミリ秒で線形に遷移させている。

(3) は幾何モデルにより指令関節角度を指令筋長へと変換する。[124] のような学習された関節-筋空間マッピングを利用しても良い。

(4) は Eq. 2.4 の筋剛性制御 [121] を行い、(5) は f^{ref} から、モータに流す実際の電流値を算出する部分である。

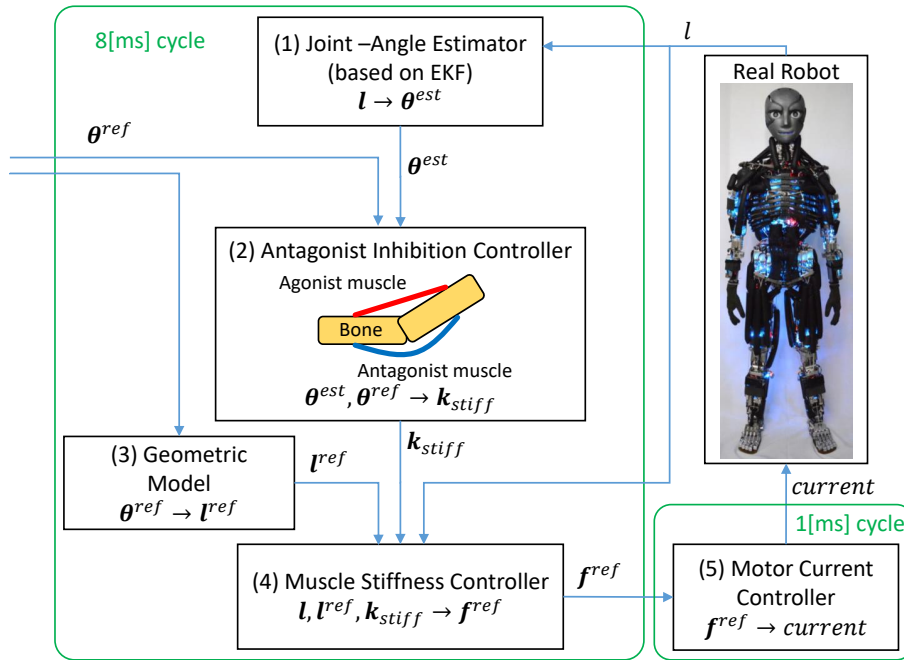


Fig. 5.41: System of antagonist inhibition control [167].

5.5.3 拮抗筋抑制制御の特徴

拮抗状態の分類

拮抗関係の状態には Fig. 5.42 の 1-9 のように 9 つの誤差の種類が存在する。

筋の状態には、ちょうどよい張力を保っているとき、緩んでいるとき、高張力となっているときの 3 種類が存在し、その組み合わせとしての 9 種類である。これらは矢印の方向に動作する際、この 9 つの状態を遷移しながら安定した状態である 10-13 の状態に収束する。最終的な状態として、10 は最も良い、実機とモデル誤差がないため緩むまず、引っ張り合うこともしない、程よい張力を保った状態である。それに対して、11,12 では両方ともモデルと完全に筋長が一致してはいるものの実機との誤差によって片方が緩んでしまうような状況であり、13 は拮抗関係にある筋が引っ張り合い、高張力を発揮しあってしまうような状況である。この状態において、最も危険な状態は 13 の状態である。これは、実機とモデルの誤差が大きければ大きいほどより高張力で拮抗し合い、筋や骨の損傷となりかねない。そして拮抗筋抑制制御の最大の利点は、この 13 の状況を必ず回避できる点にある。拮抗筋抑制制御において拮抗筋は指令した筋長に追従せずに一定張力を発揮し続けるため、13 は必ず 10 の状況に落ち着く。また、拮抗筋は一定張力を発揮し続けるため、11,12 の状況も 10 に落ち着くことになり、筋の弛みを防止することもできる。対して、筋骨格駆動ヒューマノイドの完全な幾何モデルを作るのは難

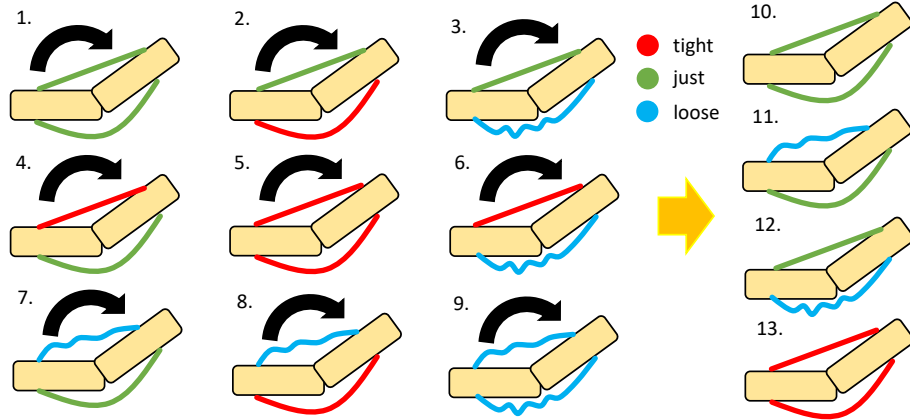


Fig. 5.42: Various types of antagonistic model error [167].

しく、モデルの正しさを前提にしてしまった場合には 11,12,13 の状況を回避することができない。

AIC とその他制御の比較

一つ新しい制御を考案し、それと拮抗筋抑制制御の違いを論じる。拮抗筋抑制制御では、主動筋か拮抗筋かの判断に以下の正負を用いていた（簡単のため、分母の $\|\theta^{ref} - \theta\|_2$ は除いている）。

$$s = G(\theta)(\theta^{ref} - \theta)[i] \quad (5.37)$$

この式において、 $\theta^{ref} - \theta$ は $d\theta$ 、 $G(\theta)d\theta$ は dI であり、以下の式と同値であるように見えるため、この式を拮抗筋抑制制御における主動筋か拮抗筋かの判断基準に使う制御を考える。

$$s = (I^{ref} - I)[i] \quad (5.38)$$

元々の拮抗筋抑制制御を関節ベース拮抗筋抑制制御 (JAIC)、ここで考えた拮抗筋抑制制御を筋長ベース拮抗筋抑制制御 (MAIC) と呼ぶこととする。JAIC では、モデルの中で拮抗していれば必ず主動筋に対して拮抗筋が抑制されることになる。つまり、必ず 13 の状況を回避することができる。それに対して MAIC では、筋長を縮ませたい場合は単純に縮ませ、筋長を伸ばしたい場合は一定の張力をかけ続ける。しかし、この場合 13 の状況に成り得る。なぜなら動作の際、拮抗筋がモデル誤差によって I^{ref} よりも縮んでしまった場合、主動筋と拮抗筋で引き合ってしまうからである。一見同じように見える 2 つの制御であるが、筋長ベースの拮抗筋抑制制御は 13 の状況を回避できない。この違いは、筋長ベースではまさに現在の筋長をそのまま使っているのに対し、関節ベースでは、現在の筋長から推定された関節角度を用いることにある。実機の筋長をそのまま使うとモデルと実機の誤差の影響を大きく受け

てしまうのに対して、実機の筋長を一度関節角に変換してモデルの中に落とし込んでしまえば、それは理想的なモデルの中であるため誤差の影響を受けないのである。そして、拮抗筋抑制制御では主動筋と拮抗筋の判定のみを行うが、この拮抗関係はモデル誤差が生じにくいという点も重要である。例えば Fig. 5.42 のような一軸関節では、筋長に関してのモデル誤差は当然生じ得るが、拮抗関係はモデルであっても実機であっても変化することはない。

その他の特徴

まず、拮抗関係の判定に用いられる定数 C に関して考察する。 C は動かしたい方向に対するモーメントアームの大きさの閾値を表している。 C が 0 の場合はモーメントアームがどれだけ小さくても、0 より小さいならば主動筋、0 より大きいならば拮抗筋とすることを意味する。これは本来推奨されるべき状態なため、本手法では基本的には $C = 0$ としている。しかし、筋長やコピアンはモデルから求めており、 $G(\theta)d\theta$ が 0 より大きいかわかりで拮抗筋か主動筋かが完璧に求まるとは限らない。つまり、これにもある程度誤差があるということである。 $C = 0$ とすることで筋長やコピアンの誤差による拮抗が起これるという問題はあるが、間違った拮抗関係を作ってしまう筋はモーメントアームの小さいものだけなため、大きく内力が溜まるということはない。また、 C を負方向に大きくすれば、動かしたい方向に少ししかモーメントアームのない主動筋がみな抑制され、拮抗筋抑制の効果が大きくなる。つまり、誤差があっても拮抗関係が起これないようにすることが可能なのである。これによって、意図しない方向への動きが多少許容されるため、うまく主動筋間の筋張力が釣り合うような方向に動作する現象が見られる。しかし、それは同時に意図しない動きを生み、位置の精度に関して問題が生じうる。逆に C を正方向に大きくすれば、動作を妨げる方向に少ししかモーメントアームのない拮抗筋が主動筋として参加し、拮抗筋抑制の効果は薄れるものの位置精度の問題は少なくなる。次に、関節角度推定値に関して考える。複雑な関節の関節角度推定値はあまり正確とは言えない。しかし、拮抗筋抑制制御ではモデル通りに実機の主動筋を動作させ、主動筋拮抗筋の判定のみに正確さに劣る関節角度推定値を用いる。拮抗関係はモデル誤差が生じにくいため、関節角度推定値の誤差はあまり問題にならない。

5.5.4 実験

本節では筋骨格ヒューマノイド Kengoro [86] を使い実験を行う。筋配置は Fig. 4.42 である。また、肩関節の構造が特殊であるため、これを Fig. 5.43 に示す。肩の肩甲骨上腕関節は球関節で 3 自由度であり、肩甲骨は胸鎖関節と肩鎖関節の 6 自由度を胸郭と肩甲骨の接触によって拘束し、3 自由度 (上方回旋下方回旋、内転外転、挙上下制) を形成している。人間は手を真上まで挙げるような動作の際、肩の 3

自由度だけではそれを行うことはできず、肩の動作と同時に、肩甲骨を上方回旋させることによってそれを行う。その比は 2:1 と言われ、肩が roll 方向に 120 度動く間に肩甲骨は 60 度上方回旋する。よって、人間が自分の背中を触ったり、棒にぶら下がったりするような広く可動域を使った動作をするためには、この肩甲骨と肩を連動させて動作させることが重要となる。

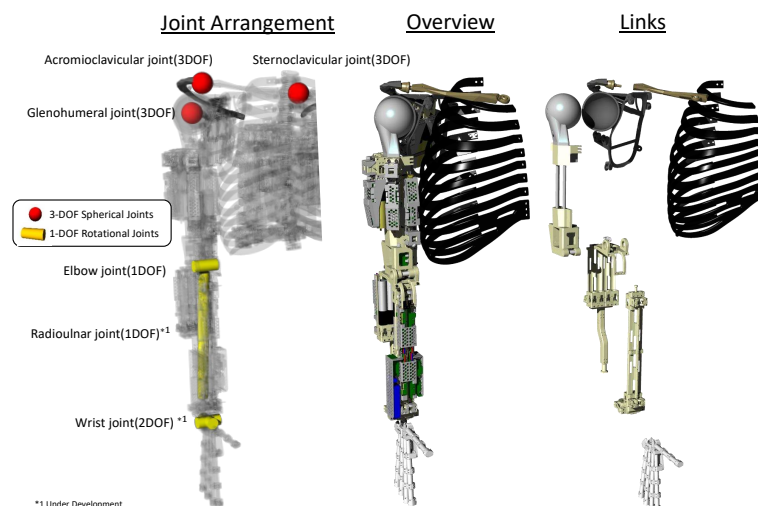


Fig. 5.43: The joint structure of the scapula and shoulder of Kengoro [167].

まず、肘関節を使った基本動作実験を行う。次に、肩単体、肩甲骨単体による拮抗筋抑制制御の動作テストを行い、この制御の有用性を示す。次に、肩甲骨と肩を使った、手を真上まで挙げる広く可動域動作を行い、その動作実現性を確認する。比較対象として、本論文で開発した拮抗筋抑制制御の k_{stiff} が一定である単純な筋剛性制御を行うこととする。最後に、モデル誤差による内力の高まりを回避した結果可能となった長時間のぶら下がり・懸垂動作を行う。この際の拮抗筋抑制制御・筋剛性制御においてはすべて、 $f_{bias} = 20$ [N], $k = 10$, $C = 0$ として実験を行った。

肘関節を使った基本実験

広く可動域動作を行う前に、肘の一軸を用いて簡単な実験を行った。腱悟郎の肘の一軸は上腕三頭筋・上腕筋・上腕二頭筋の 3 つの筋で構成されている。肘一軸の幾何モデルとしては、実験結果がわかりやすいように筋肉の起始点と終始点のみで表された簡単なモデルを使用している。肘を 0 度から -90 度まで、-30 度ずつ 3 秒進み 3 秒休みを順々に行い、関節ベース拮抗筋抑制制御 (Joint-based AIC, JAIC), 筋長ベース拮抗筋抑制制御 (Muscle-based AIC, MAIC), 筋剛性制御 (Muscle Stiffness Control, MSC), 筋張力制御 (Joint Space Control, JSC) でその際の比較を行った。その際の結果を Fig. 5.44 に示す。MAIC

と MSC の内力の高まりは同じような挙動を示している。これは, MAIC が上に述べられたようにモデルと実機の誤差によって最終的に状態 13 に落ち着き, 最終的に拮抗筋と主動筋が大きく引き合ってしまうからである。JAIC と比べると MAIC・MSC は内力の高まりによって -90 度での追従は悪いが, MSC は常に筋が引き合っているため, 動作角が小さく内力が小さい状況では非常に安定していると言える。また, JAIC と JSC は似た挙動を示しているが, JSC は筋長ヤコビアンのみを使ったトルク制御のため振動しやすく関節角度の追従も遅い。JSC は筋長ヤコビアン of 誤差に大きな影響を受けるため, 例えば複雑な肩関節の Roll 方向動作では 120 度指令に対して 60 度程度しか腕が上がっていない。JAIC での s の値を見ると, 上腕筋と上腕二頭筋は常に主動筋, 上腕三頭筋は常に拮抗筋となっていることが読み取れ, 内力の高まりを回避できていることがわかる。また, もし θ が θ^{ref} を超えてしまったとしても, この値 s の正負が全て逆になるだけであるため, 必ず内力の高まりを回避できる。

これらアプローチは複雑でモデル化の難しい筋骨格駆動ヒューマノイドをある程度の正確さの幾何モデルでも実機破損を回避しつつ動作させることができる。また, 拮抗筋抑制制御は人間にも取り入れられているため腱悟郎の人体シミュレータとしての意義も全うできる制御であると考える。

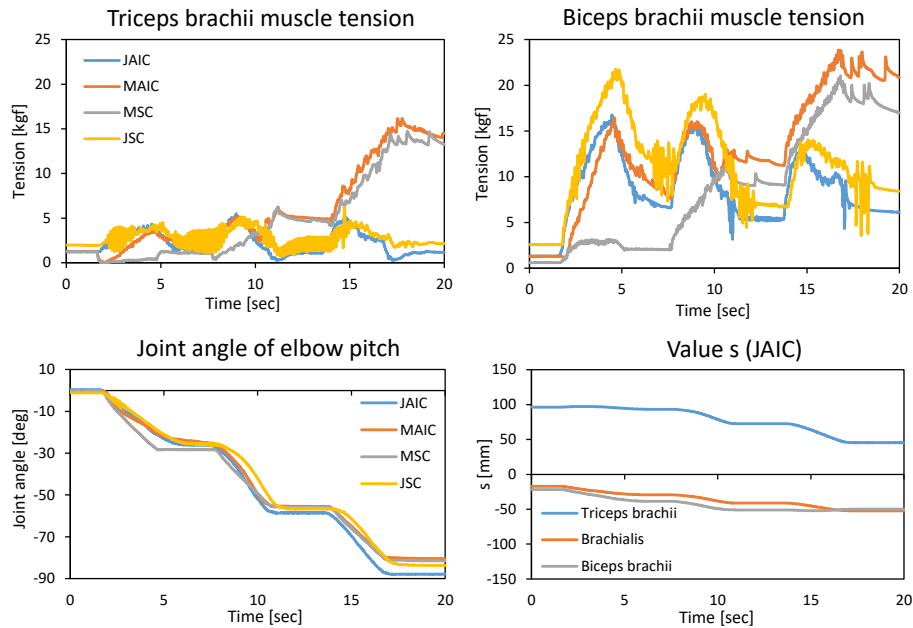


Fig. 5.44: Basic experiment of the elbow joint [167]. Upper graph shows comparison of muscle tensions of biceps and triceps brachii among joint-based AIC (JAIC), muscle-based AIC (MAIC), muscle stiffness control (MSC), and joint space control (JSC). Lower left graph shows joint angle of the elbow pitch, and lower right shows value s of JAIC.

AICによる肩甲上腕関節動作実験

肩は Fig. 4.42 のうちの #6–#15 の 10 本の筋で駆動されている。実験動作としては Fig. 5.45 のように外転を行い、その後腕を降ろす動作である。実験結果を Fig. 5.46 に示す。筋剛性制御よりも拮抗筋抑制制御のほうが最大筋張力が 43 kgf から 28 kgf にまで下がっている。特筆すべきは、筋剛性制御では肩甲下筋 (Subscapularis), 三角筋前部 (Deltoid (front)) が非常に大きな張力を発揮しているのに対し、拮抗筋抑制制御では一切張力を発揮していないことである。これは、肩甲下筋が外転方向に対してモーメントが全くなく、外旋に対してモーメントのある筋であるため、拮抗筋抑制制御では完全に抑制されていることが理由である。肩甲下筋はモデル誤差が非常に大きく、それゆえ意図しない外旋方向に対して力を発生させてしまう。それを逆方向に補うために三角筋前部 (Deltoid (front)) は大きな力を出さなければならず、筋剛性制御では無駄に張力を発揮しあっている。対して拮抗筋抑制制御では、動作させたい方向にモーメントがない筋は抑制されるため無駄な張力を発揮せず、全体として張力のピークが非常に小さくなっている。また、40 [sec] からは三角筋 (Deltoid) の 3 本の筋張力が均等に割り振られているが、これは動かしたい方向に対するモーメントの小さな筋が指令に追従せずに外転内転以外の自由度を多少許すことで、主動筋間の負荷が分散されている現象だと思われる。主動筋間の負荷分散の研究としては [190] が例として挙げられるが、主動筋を人が自ら選ぶ必要がある。本手法の拮抗筋抑制制御の主動筋拮抗筋判断部を利用することにより、拮抗筋の抑制と主動筋間負荷分散が同時にできると考えられる。

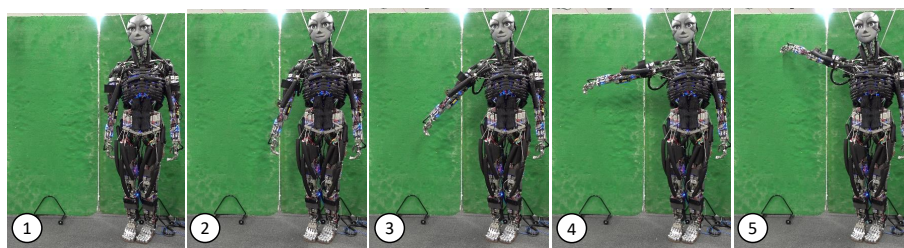


Fig. 5.45: Experimental motion of the shoulder [167].

AICによる肩甲骨関節動作実験

肩甲骨は Fig. 4.42 のうちの #1–#7 の 7 本の筋で駆動されている。実験動作としては Fig. 5.47 のように挙上下制・上方下方回旋を連続して行う。実験結果を Fig. 5.48 に示す。この場合、筋張力の時間変化に関してある程度の効果が得られている。拮抗筋抑制制御では、上方下方回旋において拮抗筋で

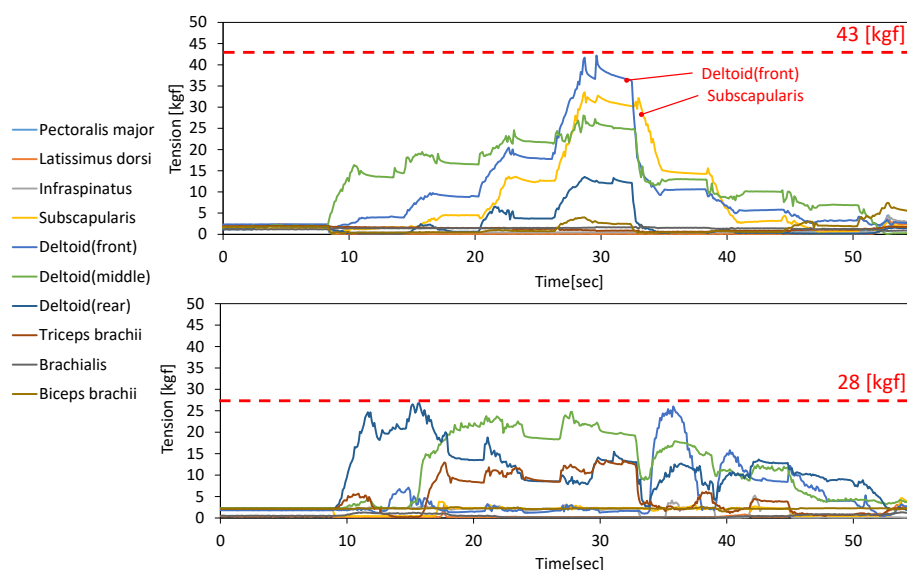


Fig. 5.46: Result of the shoulder motion experiment [167]. Comparison of muscle tension between muscle stiffness control (upper graph) and antagonist inhibition control (lower graph).

ある広背筋 (Latissimus dorsi) が抑制されており, 筋張力のピークが減っている。しかし, 肩甲骨関節は自由度に対して筋が非常に少なく拮抗関係が生まれにくい。また, 可動域が狭いためモデル誤差が乗りにくい。そのため, 筋張力最大値は 44 kgf から 37 kgf まで落ちているが, 肩ほどの効果は得られていない。

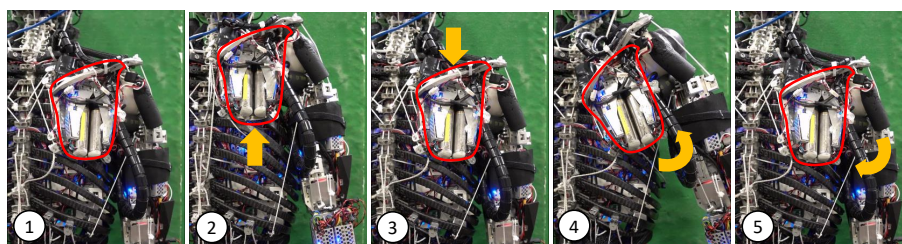


Fig. 5.47: Experimental motion of the scapula [167].

AIC による肩甲骨・肩甲上腕関節動作実験

肩甲骨と肩を使った手を真上まで挙げる広可動域動作を行う。これは肩を 120 度外転する間に肩甲骨と一緒に 60 度上方回旋する動作であり, 最終的に腕が 180 度まで回転する動作である。実験動作としては, Fig. 5.49 のように腕を真上まで上げ, その後下ろしていく。実験結果を Fig. 5.50 に示す。ま

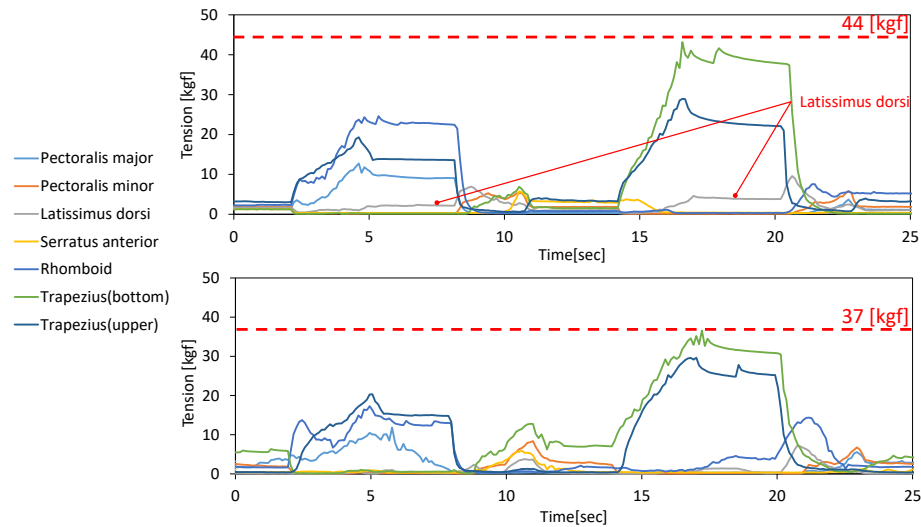


Fig. 5.48: Result of the scapula motion experiment [167]. Comparison of muscle tension between muscle stiffness control (upper graph) and antagonist inhibition control (lower graph).

ず、最大筋張力としては筋剛性制御で 55 kgf、拮抗筋抑制制御で 45 kgf と 10 kgf 程度拮抗筋抑制制御の方が筋張力を緩和できている。筋剛性制御で最大筋張力を発揮している僧帽筋上部に関しては張力測定ユニットの定格を超えており、値が飽和しているため、実際にはさらに大きな値となっていると考えられる。次に、大きな筋張力を発揮している筋の分布であるが、肩関節単体動作のときと同様に、拮抗筋抑制制御では直上拳上方向にほとんどモーメントアームのない棘下筋と肩甲下筋が抑制されているのに対して、筋剛性制御はその二つが大きな筋張力を発揮しあっている。また、拮抗筋抑制制御では肩甲骨を主に僧帽筋下部と僧帽筋上部を用いて動作させているのに対して、筋剛性制御ではモデル誤差の関係から僧帽筋上部のみに動作を頼っており、大きな張力が出てしまっている。最後に、拮抗筋抑制制御は人間に近い筋張力の高まりをしていると考えられる。筋剛性制御ではモデル誤差によって腕を真上まで上げた際に最大筋張力を発揮してしまっているが、人間の場合、腕を直角に保つ状態が最も大きく重力補償トルクが必要であり、筋張力が高まると考える。拮抗筋抑制制御では実機とモデルの誤差を吸収することができ、その動作に必要な筋のみに力が働いているため、人間に近い筋張力の高まりとなっている。

長時間ぶら下がりと懸垂運動実験

これまでに見たように、単純に筋長を追従させるような筋剛性制御では広可動域になるほど筋張力の高まりが生じ、非常に危険である。また、温度上昇により短時間しか広可動域動作を続けることがで

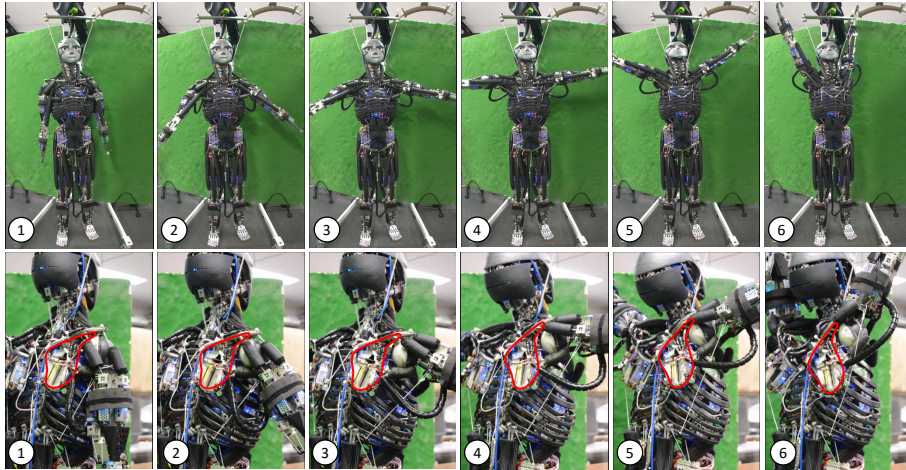


Fig. 5.49: Experimental motion of raising arm by using the scapula and shoulder [167]. Front view (upper) and side view (lower).

きない。そこで最終実験として、長時間のぶら下がり動作を行い、拮抗筋抑制制御の効果を検証する。また、その際に懸垂動作を行い、筋張力の高まり方を見ることで今後必要な制御を考える。参考として、筋骨格腱駆動ヒューマノイドの肩は脱臼する球関節でありリンクとして繋がっていないため、通常のロボットのように力を加えなくてもリンク自体で体を支えることができるわけではないことに留意されたい。実験動作結果を Fig. 5.51 に示す。まず棒に掴まり、腕を上げるような指令を送りつつ棒を持ち上げ、ぶら下がる。懸垂動作を3回行い、その後様々なポーズをさせながら14分間ぶら下がった。その際、懸垂動作以外では筋に発熱はほとんどなく、拮抗筋抑制制御により安全に長時間の広可動域動作が実現できることが示された。また、懸垂動作の際の筋張力を Fig. 5.52 に示す。拮抗筋抑制制御を用いることで筋張力の高まりを回避することができ、誤差の大きなねじり筋などの筋張力の高まりがほとんど発生していない。しかし、懸垂動作ではあまり肘が曲がっておらず、これは上腕筋の筋張力が限界に達しているからだと考える。さらに肘のトルクを増やすためには、上腕筋と上腕二頭筋の協調が必要であり、今後、主動筋間の負荷分散機構 [190] と拮抗筋抑制制御を統合する必要があると考える。

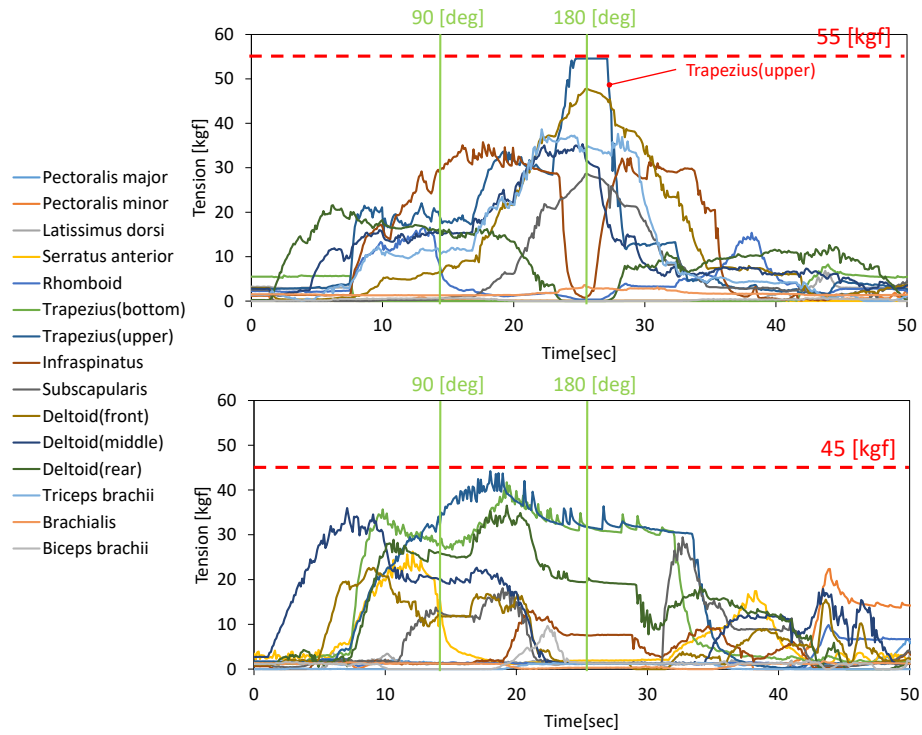


Fig. 5.50: Result of the raising arm motion experiment [167]. Comparison of muscle tension between muscle stiffness control (upper graph) and antagonist inhibition control (lower graph).

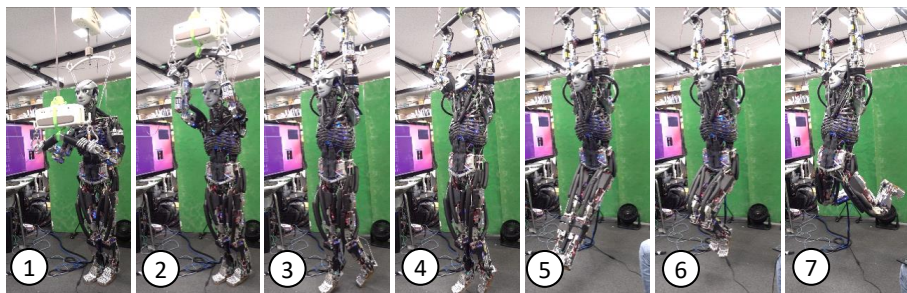


Fig. 5.51: Motion of pull-up (3-4) and dangling (5-7) [167].

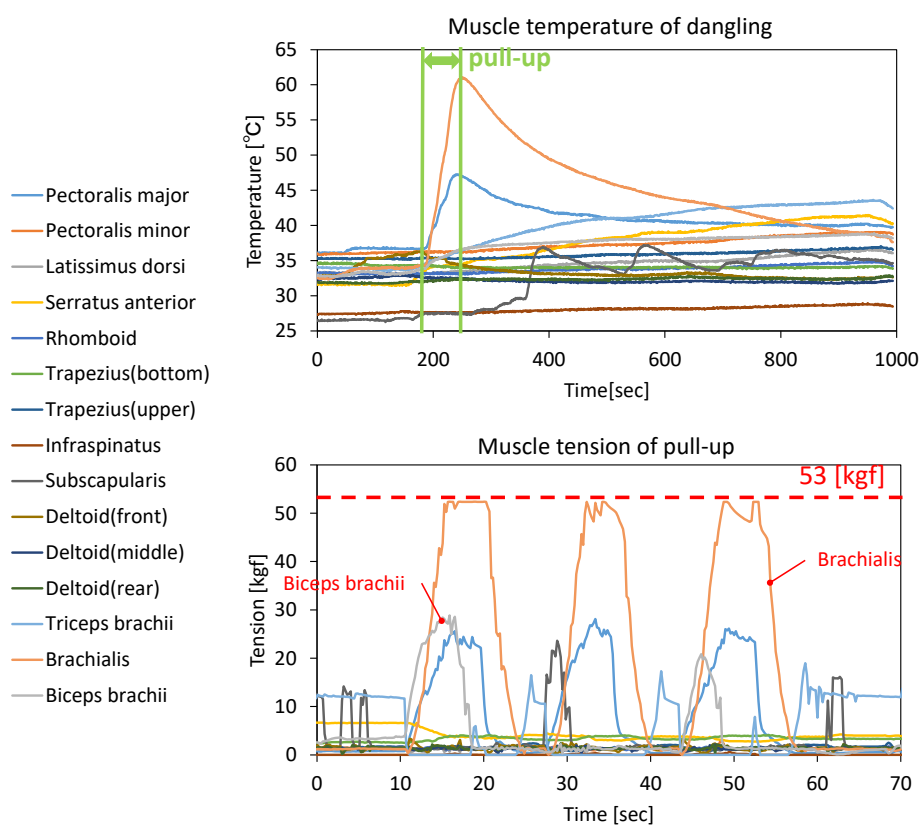


Fig. 5.52: Muscle temperature of dangling (upper graph) and muscle tension of pull-up motion (lower graph) [167].

5.6 伸長反射制御

5.6.1 概要と先行研究

人間の反射には様々な種類が存在するが、その中でも最も基本的なのは、伸長反射、ゴルジ腱反射、相反性神経支配である。これらをシミュレーションまたは実機の筋骨格構造に適用した例を紹介する。伸長反射はジャンプ時 [196, 197], または歩行時 [198] の安定性を増すことができることが示されている。相反性神経支配はモデル誤差を許した広可動域動作 [167] に有効であることが示されている。手の指に伸長反射やゴルジ腱反射、相反性神経支配を組み込むことで、特にゴルジ腱反射によって身体を安全に動作させることができることが示されている [199]。また、Hebb 則による学習を用いることで、シミュレーション上ではあるが、これらの反射が詳細な筋骨格モデルにおいて自己組織化することが示されている [200]。

これらから、伸長反射は下半身の安定的な姿勢保持、ゴルジ腱反射は身体を安全に保つこと、相反性神経支配は拮抗を減らし安全かつ効率的に動作することに有用であることがわかる。しかし、伸長反射は主に下半身の制御のみに有用性が見出されており、特に実機において上肢における有用性・実際のアプリケーションが見出されていない。そこで本節では、上肢に関する伸長反射について扱う [201]。筋骨格ヒューマノイド Musashi に伸長反射を実装し、そのアプリケーションについていくつかの仮説を構築する。実機においてそれらの実証実験を試みることで、その有用性を明らかにする。

5.6.2 伸長反射システム

人体における伸長反射

筋紡錘は筋繊維に平行に付着する紡錘形の器官であり、筋の長さ l や短縮速度 \dot{l} を検知する受容器である。筋紡錘からは Ia 群繊維が脊髄の α 運動ニューロンに直接興奮性接続されており、何らかの原因で急に筋が伸ばされると筋紡錘からのインパルス頻度が増大し、 α 運動ニューロンを興奮させ最終的に伸ばされた筋が収縮する。これは伸長反射と呼ばれる反射ループを成し、筋長を出力とする負のフィードバック系を成す。

腱器官は筋の端に配置されており、筋紡錘が筋繊維と平行に配置されているのに対し、筋繊維と直列に接続する。腱器官からは Ib 群繊維が抑制性介在ニューロンを経て脊髄の α 運動ニューロンに接続しているため、筋が収縮し筋張力が増すと腱器官からのインパルス頻度が増大し、 α 運動ニューロンの興奮作用が抑制される。これは腱反射系とよばれる反射ループを成し、筋張力を出力とする負のフィードバック系を成す。

筋紡錘からの Ia 求心性繊維は主動筋の運動ニューロンを興奮させると同時に、抑制性のニューロンを介して拮抗筋の運動ニューロンを抑制するような神経回路が用意されている。このような拮抗筋と主動筋の運動ニューロン間の相互作用を相反性神経支配という。

以下、本節ではこの中でも伸長反射を筋骨格ヒューマノイドの実機に実装し、その有用性を確認していく。

伸長反射の実装

前述したように、伸長反射は筋長 l が素早く伸ばされたときに、筋を収縮させるような制御である。ここで、伸長反射が起こる条件について考える。先ほどの定義からすると、一般的には筋長の速度 $\Delta l = l_{t+1} - l_t$ (l_t は時刻 t の筋長を表す) が、閾値 $C_{stretch}$ よりも大きくなったときに伸長反射が発生する。しかし、2.5.1 節で述べたように、筋骨格ヒューマノイドには非線形弾性要素が含まれていることがほとんどである。この場合、何かの衝撃で急に腕が押されても、モータよりも柔軟で衝撃を受けることが可能な非線形弾性要素の変形が支配的となる。非線形弾性要素にかかる筋張力 f とその伸び Δn の関係を指数関数として $f = e^{k\Delta n}$ (k は定数とする) とすると、以下のように式を変形することで、筋が伸ばされたかどうかを判定できる。

$$\begin{aligned} \Delta n_{t+1} - \Delta n_t &> C_{stretch} \\ \frac{1}{k} \log(f_{t+1}) - \frac{1}{k} \log(f_t) &> C_{stretch} \\ f_{t+1} - f_t &> C'_{stretch} \end{aligned} \quad (5.39)$$

ここで、 $C'_{stretch} = e^{kC_{stretch}}$ とする。つまり、筋張力の前時刻との差分 $\Delta f = f_{t+1} - f_t$ が $C'_{stretch}$ より大きくなったときに、伸長反射を発生させれば良い。

次に、伸長反射の動作について考える。これは定義通り、条件を満たした瞬間に指令筋長 l^{ref} をある一定長さ $\Delta l_{stretch}$ だけ収縮させる。その後、 Δt_{loose} 時間かけて収縮された $\Delta l_{stretch}$ 分の筋を元に戻していく。ここで、ある筋の伸長反射により他の筋の伸長反射が誘発されてしまうのを防ぐため、一つの筋に伸長反射が起きたら、周辺の筋に関しては伸長反射を起こさないように抑制する。ゆえに、伸長反射のフローチャートは Fig. 5.53 のようになる。なお、Fig. 5.53 のプロセスは全ての筋で同時並列的に起こるため、同じタイムステップ時に条件を満たすことができれば、同時に複数の筋が伸長反射を起こすことがあり得る。また、右腕、左腕など、明らかに伸長反射を連鎖的に誘発しない独立した部位に分けてそれぞれこのプロセスを行うことで、タイミングがずれていても、右腕、左腕にそれぞれ伸長反射が起こりうる。

ここで、本手法における伸長反射の挙動を決めるパラメータは $C'_{stretch}$, $\Delta l_{stretch}$, Δt_{loose} の3つである。これらのパラメータによる伸長反射の違いは、後の実験で議論する。

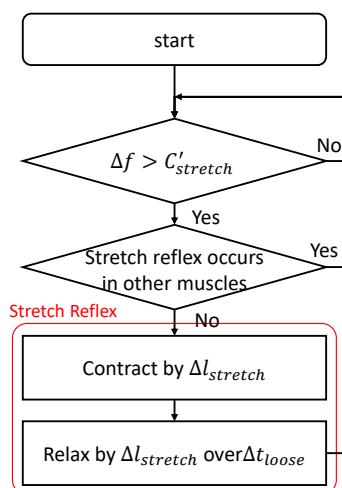


Fig. 5.53: The flow chart of the implemented stretch reflex [201].

伸長反射のアプリケーション

この伸長反射を上肢に適用したときに、どのようなアプリケーションが考えられるかについて考える。その簡単な分類を Fig. 5.54 に示す。本節では、それらをまず受動的・能動的という二つに分けた。

受動的とは、上肢への突然の衝撃によって伸長反射が発生するような場合である。そしてこれは、危険回避と姿勢保持という2つに分類される。危険回避とは、関節角度リミット近くで衝撃が加わった場合、伸長反射を入れることで、リミットまで到達せず、身体を安全な可動範囲に保つことが可能であるという仮説である。姿勢保持とは、突然の衝撃が加わったあと、伸長反射があることで、上肢の関節角度を衝撃が加わる前と同じに保つことが可能であるという仮説である。また、姿勢保持にも、身体に制御が入っていない、つまり一定筋長を保っているような場合と、身体に制御が入り、自身の関節角度を一定に保とうとフィードバックが行われている場合という2つに分けて考えることができる。

能動的とは、自分から環境に力を加え、伸長反射を起こすことで、通常よりも素早く・強い力を出すことができるという仮説である。重量挙げやジャンプがその好例だと考える。

本節ではこれらの仮説のもと、筋骨格ヒューマノイドの実機において実験を行い、伸長反射の有効性について検証していく。

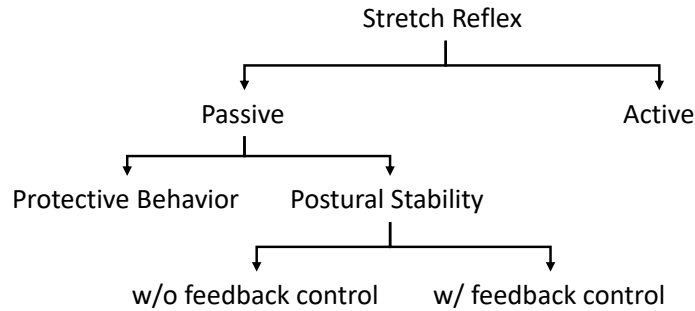


Fig. 5.54: The classification of applications of stretch reflex [201].

5.6.3 実験

実験セットアップ

本実験では筋骨格ヒューマノイド Musashi [87] を用いる。その左腕の筋配置を Fig. 4.24 に示す。人体を模倣した筋配置であり、片腕で 18 の筋 (#1–#18) を有し、二関節筋も含まれる。本実験では主に肩と肘の 5 自由度を用い、それらに関係する筋は 10 本 (#1–#10)、うち二関節筋が 1 本 (#9) 含まれている。これらの関節角度を S-p, S-r, S-y, E-p, E-y と表す (S は肩関節, E は肘関節, rpy は roll, pitch, yaw を表す)。

本節では以下において、Fig. 5.54 に示された 4 つの動作を対象に、実験を行っていく。なお、 $C'_{stretch}$ については、伸長反射が容易に起こらないように、基本的な動作を行った際に記録された Δf の最大値より大きな値である 15 N を全ての実験で採用した。また、特に記載がない場合は $\Delta l_{stretch} = 10.0$, $\Delta t_{loose} = 0.5$ とする。

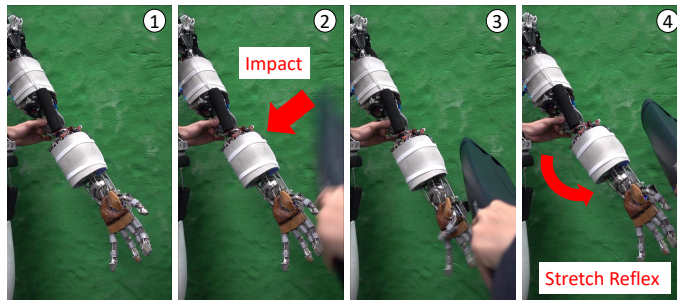


Fig. 5.55: The experiment of the protective behavior of stretch reflex when adding sudden impact to the elbow joint [201].

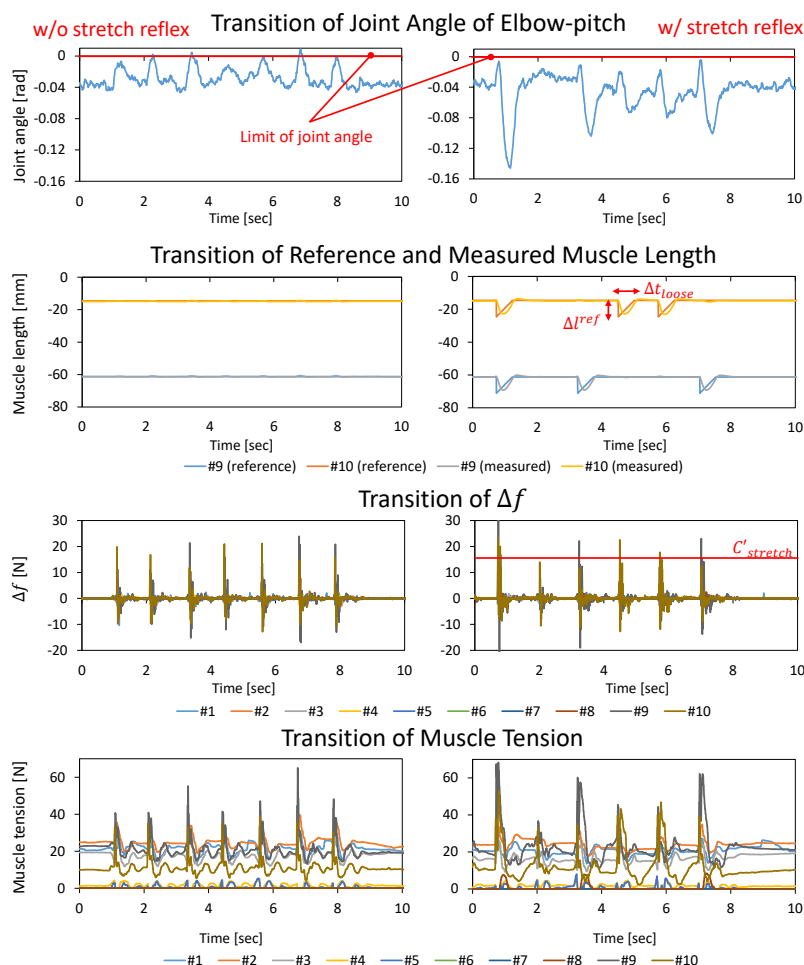


Fig. 5.56: The experimental results of the protective behaviors with or without stretch reflex [201]. The graphs show the joint angle of E-p, the comparison between the reference and measured muscle lengths regarding #9 and #10, the change in muscle tension Δf , and muscle tension f .

伸長反射の危険回避実験

伸長反射を入れたときの危険回避に関する実験を行う。Fig. 5.55 のように、E-p を少しだけ曲げ (約-0.04 rad), 上腕を掴んで固定した状態で、前腕に衝撃を連続的に加える。このとき、伸長反射を入れた時と入れない時における振る舞いの違いを検証した。

E-p の関節角度 θ , 衝撃によって主に伸ばされる筋#9 と#10 の指令筋長 l^{ref} と測定された筋長 l の比較, 筋張力の変化 Δf , 筋張力 f の遷移を Fig. 5.56 に示す。伸長反射がない場合は、衝撃を加えた際、肘の関節角度限界である 0.0 rad に到達し、めり込んでいることがわかる。肘の関節角度制限部品が 3D プリントパーツであるため、それを変形させるような強い力が加わっている。これに対して、伸長反射

を入れた場合は、関節角度限界にめり込むことはない。 Δf が $C'_{stretch}$ を超えた瞬間に伸長反射が起こり、指令筋長が $-\Delta l_{stretch}$ だけ一気に変化し、それが Δt_{loose} かけて元に戻っていくことがわかる。上腕二頭筋である#9 と上腕筋#10 は、片方だけ伸長反射が起こったり、同時に起こったりしている。伸長反射がある場合とない場合では最大筋張力に大きな差はないが、伸長反射を入れた場合は入れない場合に対して、制御が入る分だけ筋張力が高い期間が少し長くなる。

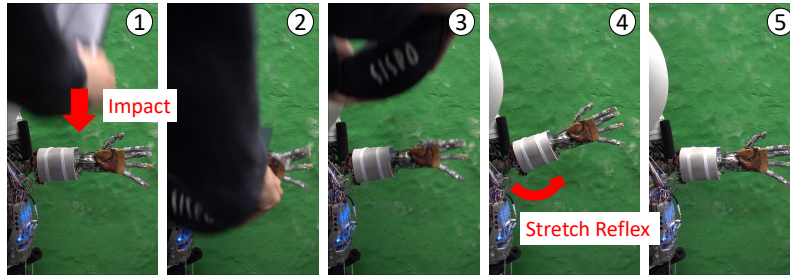


Fig. 5.57: The experiment of stretch reflex for postural stability [201].

伸長反射の姿勢安定化実験

伸長反射を入れた時の、姿勢安定化に関する効果について検証を行う。Fig. 5.57 のように、E-p を曲げ、その状態で前腕に連続的に衝撃を加える。前述の実験では関節角度限界付近で危険回避動作について検証したのに対して、本実験では衝撃が加わる前と後での姿勢変化について考察する。また、伸長反射のパラメータを変化させ、それによる振る舞いの違いについても検証する。

E-p の関節角度 θ , #9 と #10 の指令筋長 l^{ref} と測定された筋長 l の比較、筋張力 f の遷移を Fig. 5.56 に示す。伸長反射を入れない場合 (0) と伸長反射を入れた場合: (1) $\Delta l_{stretch} = 10, \Delta t_{loose} = 0.5$, (2) $\Delta l_{stretch} = 10, \Delta t_{loose} = 1.0$, (3) $\Delta l_{stretch} = 20, \Delta t_{loose} = 1.0$ について比較を行う。筋長のグラフから、それぞれのパラメータに従って、伸長反射の際に妥当な筋長指令が生成されていることがわかる。このとき、それぞれ 7 回の衝撃を加える前と後における肘の関節角度の差分について検証する。伸長反射を入れないとき 0.023 rad なのに対して、(1) では 0.0079 rad, (2) では 0.0025 rad, (3) では 0.0071 rad であった。伸長反射を入れた場合は、伸長反射を入れない場合に比べて、衝撃による関節角度変化を抑制できていることがわかる。また、最大筋張力については、 $\Delta l_{stretch}$ が大きな (3) について 91.7 N と最も大きかったが、全体として大きな違いは見られなかった。

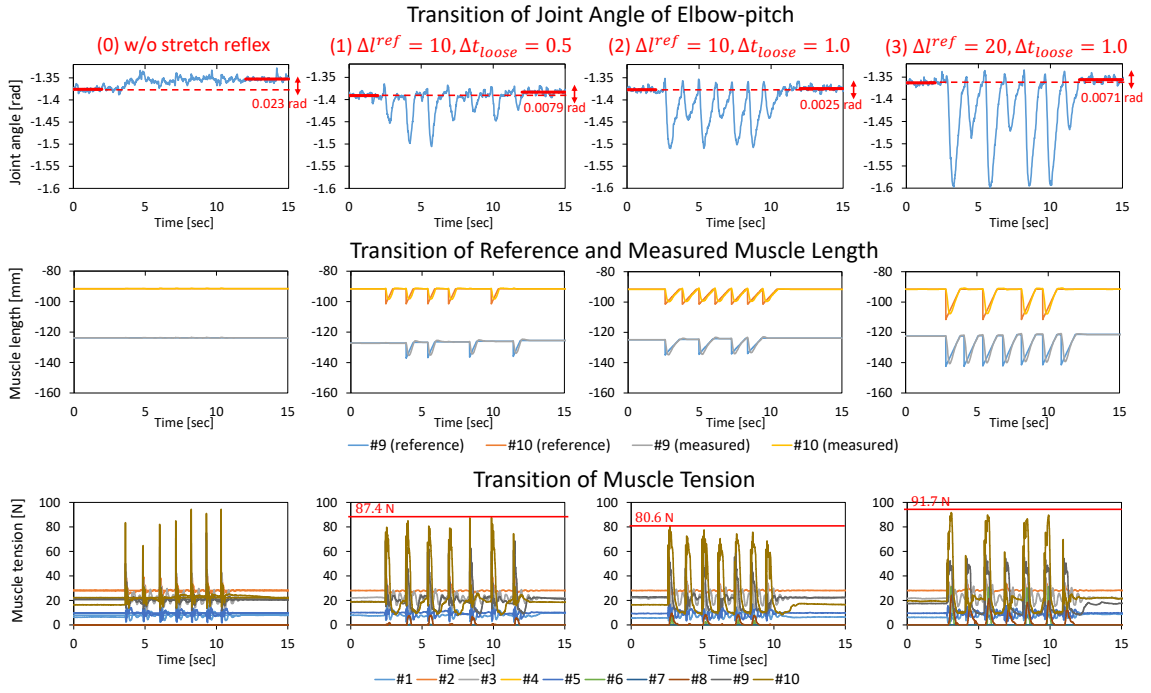


Fig. 5.58: The experimental results of stretch reflex for postural stability [201]. The graphs show the joint angle of E-p, the comparison between the reference and measured muscle lengths regarding #9 and #10, and muscle tension f , without stretch reflex (0) and with stretch reflex: (1) $\Delta l_{stretch} = 10, \Delta t_{loose} = 0.5$, (2) $\Delta l_{stretch} = 10, \Delta t_{loose} = 1.0$, and (3) $\Delta l_{stretch} = 20, \Delta t_{loose} = 1.0$.

伸長反射の姿勢安定化実験：関節角度フィードバックとの併用

伸長反射に加えて、関節角度フィードバックを入れた時の姿勢安定化に関する効果について検証を行う。Fig. 5.59 のように、E-p の関節角度を-90 度に保つようにフィードバック制御を加えた状態で、3.6 kg のダンベルの入った袋を、約 20 cm 上から前腕の先に対して落として荷重を増加させる実験を行う。本実験も前節と同様に、伸長反射のパラメータを変更しながら比較する。ここで関節角度フィードバックとは、現在の関節角度 θ を測定し、 θ^{ref} を指令関節角度、 $\theta^{virtual}$ を仮想的な指令関節角度として、 $\theta^{virtual} \leftarrow \theta^{virtual} + \alpha(\theta^{ref} - \theta)$ のように仮想的な指令関節角度を更新していく方法である。その後、指令筋長 l^{ref} は関節角度と筋長の写像 h を用いて、 $l = h(\theta^{virtual})$ [100] のように計算されて実機に送られる。なお、本節では $\alpha = 0.3$ で、この制御は 5 Hz で行われる。

伸長反射を入れない場合 (0) と伸長反射を入れた場合: (1) $\Delta t_{loose} = 1.0$, (2) $\Delta t_{loose} = 3.0$, (3) $\Delta t_{loose} = 5.0$ における E-p の関節角度の遷移を Fig. 5.59 に示す。このとき、衝撃が加わった際の関節角度の最大値は (0) -1.49 rad, (1) -1.52 rad, (2) -1.50 rad, (3) -1.51 rad と、指令値 θ^{ref} である -1.57 rad

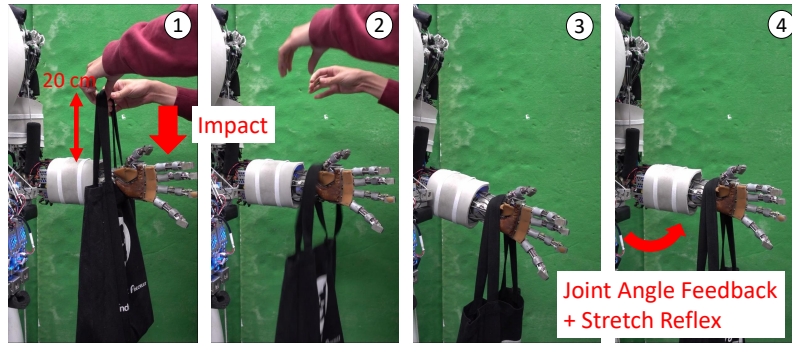


Fig. 5.59: The experiment of using stretch reflex with joint angle feedback control for postural stability [201].

からの変化が伸長反射を入れない場合に最も大きかった。つまり、伸長反射を入れた場合は衝撃が加わった際の最大関節角度変位が抑制されている。また、 $f_{thre} = -1.55$ [rad] を閾値として、衝撃が加わった瞬間からそれ以降 f が f_{thre} を上回らない最初の時間までの期間を Δt_{conv} とする。このとき、(0) $\Delta t_{conv} = 1.81$ [sec], (1) $\Delta t_{conv} = 2.51$ [sec], (2) $\Delta t_{conv} = 4.03$ [sec], (3) $\Delta t_{conv} = 0.42$ [sec] となった。 Δt_{loose} が小さいと、伸長反射がない場合に比べて収束までの時間が長くなってしまいうのに対して、ある閾値を超えると、収束までの時間が伸長反射がない場合に比べて大幅に短くなることがわかった。

伸長反射の能動的発動実験: 重量挙げ

伸長反射を能動的に用いる動作について、重量物を持ち上げる動作を例に検証を行う。Fig. 5.61 のように、重さ 10 kg のダンベルを両手で持ち上げる動作を 1 秒で指令する。このときの伸長反射の効果を検証する。

伸長反射を入れない場合と入れた場合について、右腕の E-p の関節角度と筋張力の遷移を Fig. 5.62 に示す。伸長反射を入れない場合と入れた場合において、ダンベルを持ち上げる前と後の関節角度変化はほとんど変わらない。しかし、伸長反射を入れた場合は、一瞬だけ関節角度が大きく変化し、その後戻っていくような推移が見られる。筋張力は、伸長反射を入れない場合は最大で 234 N, 入れた場合は最大で 257 N となった。また、持ち上げた後の筋張力は、伸長反射を入れない場合は 132 N, 入れた場合は 86 N となった。伸長反射を入れることで、一瞬の最大筋張力は増大するものの、持ち上げたあとの筋張力は減少することが分かった。

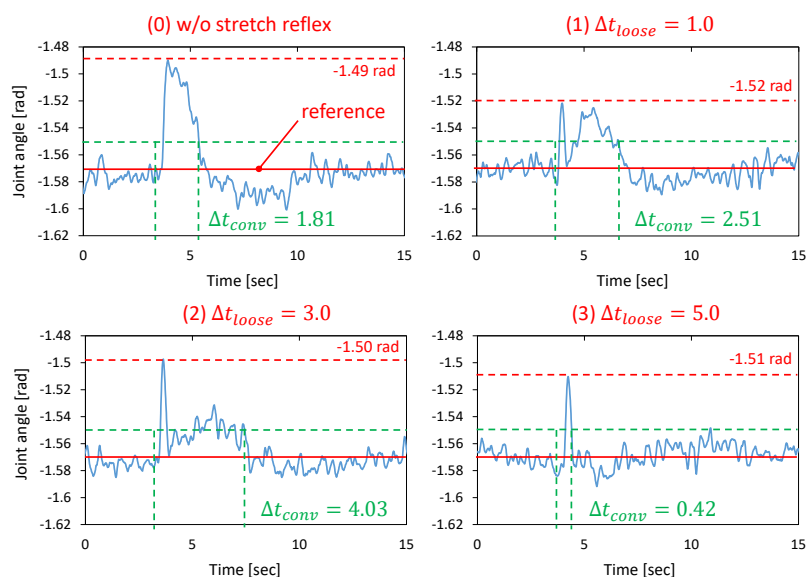


Fig. 5.60: The experimental results of using stretch reflex with joint angle feedback control for postural stability [201]. The graphs show the joint angle of E-p without stretch reflex (0) and with stretch reflex of modified parameters: (1) $\Delta t_{loose} = 1.0$, (2) $\Delta t_{loose} = 3.0$, and (3) $\Delta t_{loose} = 5.0$.

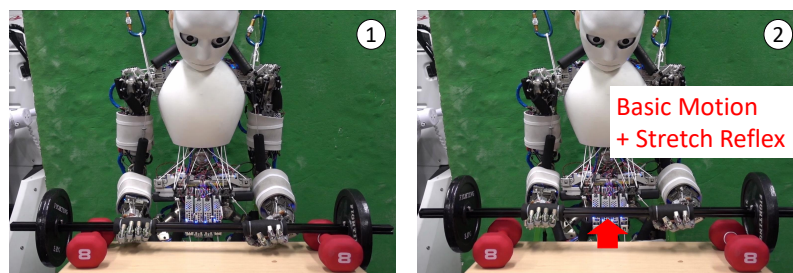


Fig. 5.61: The experiment of actively using stretch reflex when lifting a heavy dumbbell [201].

5.6.4 議論

これまでの4つの実験の結果をまとめ、考察する。

危険回避実験では、伸長反射がない場合とある場合について、関節角度限界付近において衝撃を加えたときの挙動を確認した。伸長反射がない場合は関節角度限界に達し、関節のリミット部品に対して大きな力がかかってしまう一方、伸長反射を入れた場合は関節角度限界に達しないような危険回避を行うことができる。

姿勢安定化実験では、肘を大きく曲げた状態で衝撃を加え、伸長反射のパラメータを変化させながら、そのときの姿勢安定化の効果について確認した。伸長反射を入れることで、入れない場合に比べて、

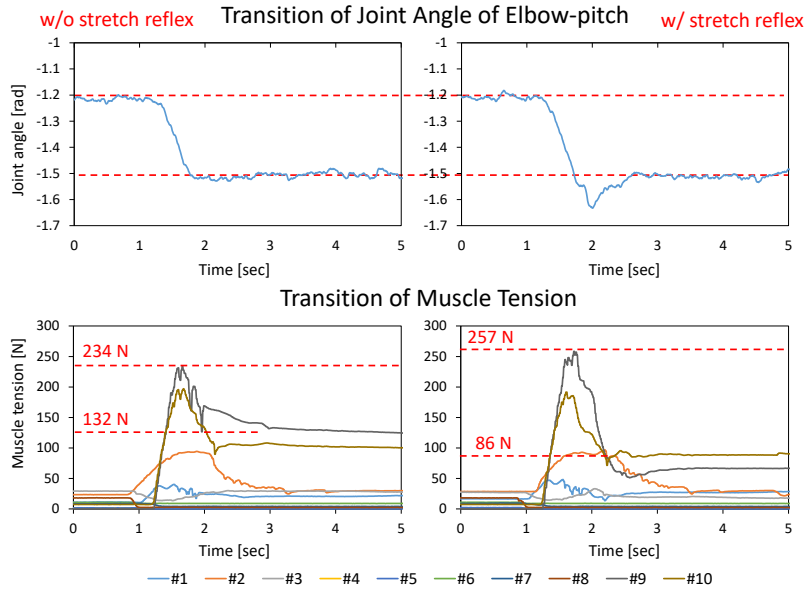


Fig. 5.62: The experimental results of actively using stretch reflex when lifting a heavy dumbbell [201]. The graphs show the joint angle of E-p and muscle tension, without stretch reflex (0) and with stretch reflex of modified parameters: (1) $\Delta t_{loose} = 1.0$, (2) $\Delta t_{loose} = 3.0$, and (3) $\Delta t_{loose} = 5.0$.

衝撃を加える前と後での関節角度の変化が小さくなった。筋骨格ヒューマノイドには摩擦によるヒステリシスが大きく、伸長反射がない場合は、衝撃によって徐々に関節角度が変化してしまう。これに対して、伸長反射を入れることで一度衝撃で変化した関節角度が元に戻り、常に衝撃前と同じ関節角度を保持し続けることができる。 $\Delta t_{stretch}$ を小さくすればかかる筋張力を小さくできる一方伸長反射の効果は薄まり、 Δt_{loose} を小さくすれば伸長反射の期間を短くしてすぐに次の衝撃に対応できる一方、小さくし過ぎると伸長反射の効果が薄れ、後述するフィードバック等の観点からもある程度の長さがあった方が良い。このように、伸長反射のパラメータにはトレードオフが存在する。

関節フィードバックと併用した場合の姿勢安定化実験では、重い物体を突然持たせたときの、関節角度フィードバックと伸長反射を同時に入れた場合の姿勢安定化について確認した。伸長反射を入れることで、入れる前に比べて衝撃時の最大関節角度変化が抑制された。また、 Δt_{loose} が小さいと、伸長反射後にすぐに筋が戻ってしまい、伸長反射がない場合と同じようにフィードバックがかかり、より収束までの時間が遅くなってしまふ。それに対して、十分な長さの Δt_{loose} が確保されると、衝撃後に伸長反射で θ^{ref} まで一気に戻った後、そのまま関節角度フィードバック制御と相まって θ^{ref} が保ち続けられるような現象が見られた。つまり、 Δt_{loose} を適切に設定することで、衝撃や荷重が加わった後の素早い姿勢安定化が可能となるのである。

伸長反射の能動的発動実験では、能動的に伸長反射を起こす例として、重量物の持ち上げの際の伸長反射の効果について確認した。伸長反射を入れることで、一瞬だけ最大筋張力が増大してしまうものの、最終的な筋張力は減少した。非常に重い物体に無理な力をかけることで伸長反射が起き、一瞬だけ筋張力が増大する。摩擦による筋のヒステリシスが大きいと、通常の動作時にはその摩擦を乗り越えるために大きな筋張力が必要となるのに対して、伸長反射を入れた場合は一瞬だけ筋を速い速度で動かすため、その摩擦を乗り越えて最終的に収束する筋張力が減少することがわかった。つまり、意図して無理な力を環境にかけることで伸長反射を誘発させ、それを利用することで最終的な必要筋張力を削減してタスクを遂行することができる。

このように、伸長反射は関節角度限界を回避し、関節フィードバックと合わせることで速い収束を促すことができる。また、筋の摩擦によるヒステリシスを克服し、衝撃による姿勢変化を抑制し、能動的に使用することで筋張力削減に寄与する。これまで主にシミュレーションや下肢における姿勢安定化のみについて議論されてきたのに対して、これらの結果は実機における摩擦等の影響が加味されて始めて得られるものである。

本節ではパラメータの変化による挙動の違いについて見たが、これらを自動的に決定する手法が必要である。人間の伸長反射のパラメータは、タスクや環境によって変化することが知られており [202]、この機構を筋骨格ヒューマノイドにも組み込むことが求められる。

5.7 本章のまとめ

本章では、身体図式学習を支える、ロボットの柔軟性と冗長性の欠点を補完する5つの反射制御について述べた。

まず、モータ温度管理のための、温度モデルパラメータのオンライン学習、そのモデルを使ったモータコア温度推定、異常検知、モータコア温度管理のための出力制限制御について提案した。モータハウジング温度のマッチングによって温度モデルパラメータをオンラインで学習させることができる。また、ニューラルネットワークの重みではなく明示的に解釈可能なパラメータを用いることで、異常検知が可能となる。モータコア温度を最大値に保つための最大出力の時系列も誤差逆伝播法を用いて得ることが可能であり、これによってモータコア温度を管理することが可能となる。最後に、本手法の有効性をシミュレーション・実機により示し、筋骨格ヒューマノイドへの適用も実現した。

次に、筋骨格ヒューマノイドが幾何モデルとの誤差による無駄な筋内力を除去し、環境を積極的に利用して必要な筋張力を最低限に保つ筋弛緩制御を開発した。現在のタスクに支障が出ない範囲内で、重要度の低い筋から順に弛緩させていき、筋内力を減らしていく。また、環境によって身体を拘束することで、重要度の高い主動筋の筋張力も抑えていくことができる。筋弛緩制御を用いることで、通常の基本動作において内力の高まりを抑えられるだけでなく、ハンドル操作のような環境接触下におけるタスクも、小さな力で遂行することが可能となった。

次に、冗長で複雑な拮抗腱駆動構造を有する筋骨格ヒューマノイドにおいて、アクチュエータによって規定される最大関節角速度よりも大きな速度を出す手法を二種類提案した。一つは動作時に拮抗筋の電流値を0として抑制し、バックドライバビリティを用いる手法である。もう一つは動作前に予め弛緩可能な拮抗筋を弛緩させてしまう手法である。両者ともシミュレーション・実機実験からより速い関節角速度を出すことが可能であることがわかった。また、前者はバックドライバビリティに依存し、後者はバックドライバビリティに依存しない。

次に、人体の反射である相反性神経支配を模した拮抗筋抑制制御による動作範囲の拡張について述べた。完全に正確な幾何モデルでなくとも、筋張やコビアンと関節角度推定値から主動筋か拮抗筋かを判定し拮抗筋を抑制することによって、幾何モデルと実機の誤差による動作の妨げや、筋や骨格の破損、筋の弛みを防ぐことができることを提案した。拮抗筋抑制制御の肩甲骨や肩への適用を通して、通常の筋剛性制御では広可動域になればなるほど内力の溜まる状態を解消できることを示した。そして、今まで大きく内力が溜まるために非常に危険だったぶら下がりや懸垂という広可動域動作を、拮抗筋抑制制御により安全に長時間実現できることを示した。

最後に、筋骨格ヒューマノイドの上肢実機に伸長反射を組み込み、その有用性について検証した。受動的と能動的に分け、受動的動作については衝撃が加わった際の危険回避・姿勢保持・関節フィードバック制御を入れた際の姿勢保持について、能動的動作については重量物体の持ち上げ動作について有用性を検証した。伸長反射を入れることで、関節角度限界近くでの衝撃に対応し、関節に負荷をかけないようにすることができる。また、衝撃を加えられた際に、ヒステリシスによる関節角度変化を抑制し、パラメータを調整することで、フィードバック制御と合わせて素早い収束を実現することができる。重量物体を持ち上げる際は、意図して身体に衝撃を加えることによって誘発される伸長反射を利用することで、最終的により小さな筋張力でタスクを遂行することができる。つまり、筋骨格ヒューマノイドの上肢において、実験から人間の伸長反射の複数の有用なアプリケーションを見出すことに成功した。

第6章

身体図式学習の基礎実験

本章の概要を Fig. 6.1 に示す. 本章では, 第 3 章に述べた一般化多感覚相関モデルにおける基礎的な 4 種類のネットワーク構成 (Fig. 3.4) について, それぞれの具体例を示す基礎実験を行う. まず, 6.1 節において, 説明の都合から先に第 6 章と第 7 章の実験に関して, 身体図式ネットワーク構造の自動決定実験をまとめて述べておく. 次に, 6.2 節において, (s-1) の具体例である Tool-Body Network with Parametric Bias を用いた適応的道具先端操作学習実験について述べる. 次に, 6.3 節において, (s-2) の具体例である Musculoskeletal AutoEncoder を用いた筋骨格ヒューマノイドにおける関節-筋空間マッピング学習実験について述べる. 次に, 6.4 節において, (d-1) の具体例である Recurrent Neural Network with Parametric Bias を用いた動作スタイルを含む模倣学習実験について述べる. 最後に, 6.5 節におい

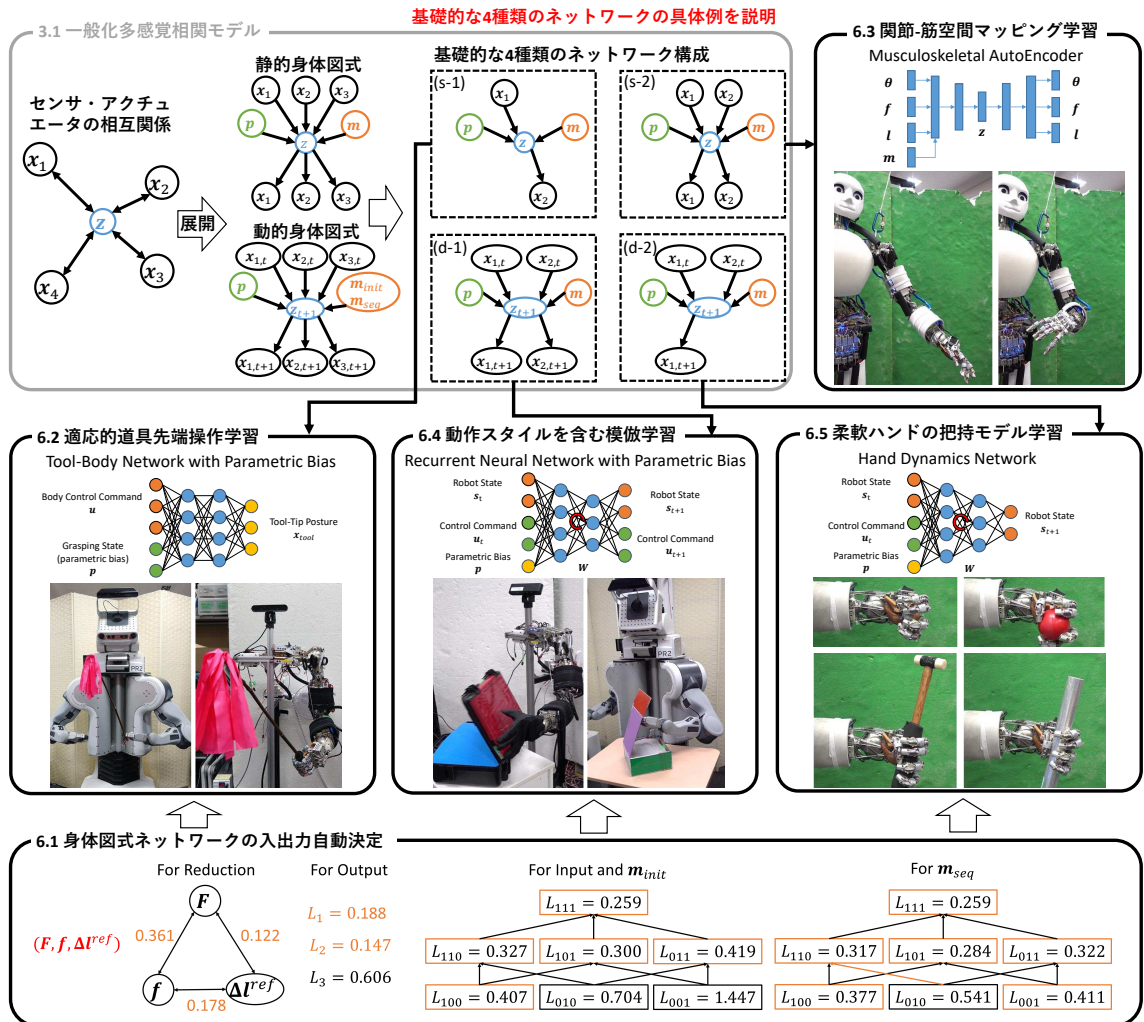


Fig. 6.1: The overview of this chapter.

て, (d-2) の具体例である Hand Dynamics Network を用いた筋骨格柔軟ハンドの把持モデル学習実験について述べる. これら4つの基礎実験により, 身体図式学習を用いたロボットの環境適応能力向上について確認する.

6.1 身体図式ネットワークの入出力自動決定実験

以降で述べる 6.2 節, 6.3 節, 6.4 節, 6.5 節, 7.6 節, 7.7 節, 7.3 節, 7.4 節について, 一般化多感覚相関モデルの入出力自動決定に関する実験を行う. ここで, 静的身体図式は 6.2 節, 6.3 節, 7.3 節の3つである. 動的身体図式は 6.4 節, 6.5 節, 7.6 節, 7.7 節, 7.4 節の5つである. なお, C_{thre}^{reduce} については全ての実験で 0.05, $C_{thre}^{\{out, in, init, seq\}}$ については実験ごとに变化させている.

6.1.1 静的身体図式の入出力自動決定実験

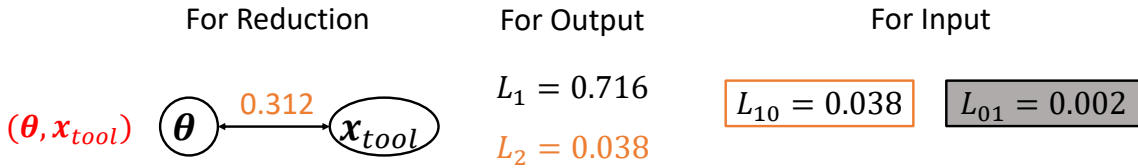
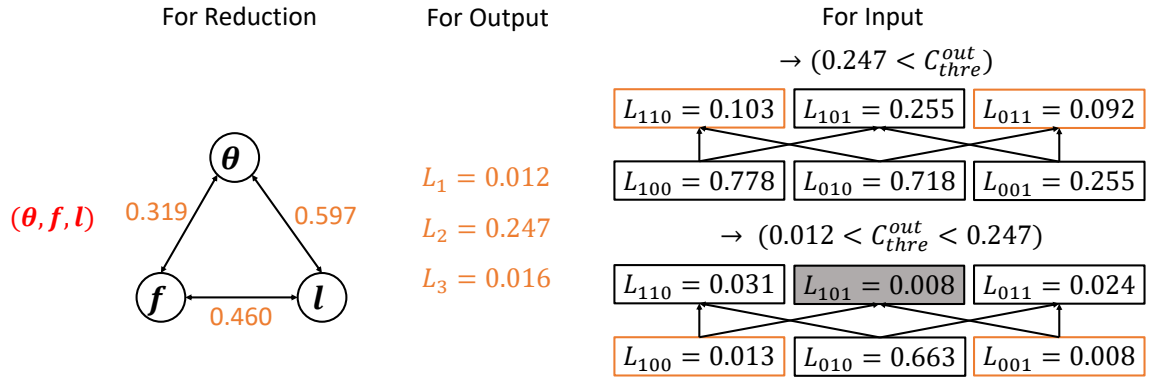


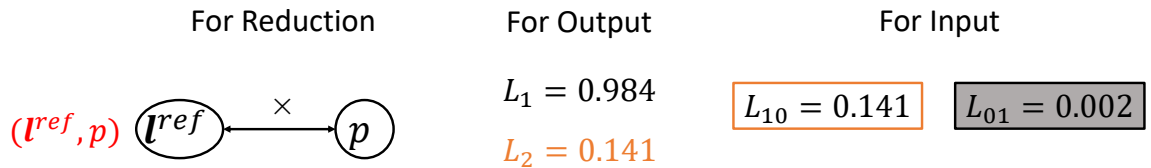
Fig. 6.2: Mutual information and loss values in Sec. 3.2.2 for the relationship among (θ, x_{tool}) in Sec. 6.2.

まずは 6.2 節の身体-道具間関係学習について実験を行う. 用いるセンサは (θ, x_{tool}) の2つであり, ランダムな関節の動きからデータを取得する (なお, x_{tool} は道具先端位置を表す). 得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.2 に示す. まず, 不要なセンサ・アクチュエータの削除について, θ と x_{tool} の間の相互情報量が計算され, これが C_{thre}^{reduce} 以上であり, 2つのセンサが関係し合っていることがわかった. 次に, ネットワーク出力の決定のための $L_{\{1,2\}}$ を計算した. L_2 は小さいのに対して, L_1 は大きく, 推論できるとは言いがたい. よって, ネットワークの出力は x_{tool} のみとなった. 次に, ネットワーク入力決定のための L_m を計算した. 出力が x_{tool} のみであるため, $L_{\{01,11\}}$ は計算されない. よって, L_{10} のみが計算され, これが十分小さいと判断し, 入力 θ となった. よって, $\theta \rightarrow x_{tool}$ という, 6.2 節における TBNPB と同じネットワークが構築された.

次に 6.3 節の筋骨格構造に関する静的身体図式について実験を行う. 用いるセンサは (θ, f, l) の3つであり, ランダムな関節と筋張力の動きからデータを取得する (なお, 本実験では筋骨格ヒューマノイド Musashi ではなく, 1 自由度の腱駆動シミュレータに対して実験を行った). 得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.3 に示す. まず, 不要なセンサ・アクチュエータ

Fig. 6.3: Mutual information and loss values in Sec. 3.2.2 for the relationship among (θ, f, l) in Sec. 6.3.

の削除について, (θ, f, l) のそれぞれのモデル間の相互情報量が計算され, 全ての辺が C_{thre}^{reduce} 以上であり, それぞれのセンサが関係し合っていることがわかった. 次に, ネットワーク出力の決定のための $L_{\{1,2,3\}}$ を計算した. L_1 と L_3 はほぼ同程度に小さいのに対して, L_2 はそれらと比べると多少大きい, 全体として損失自体は小さいと言える. ここで, $0.247 < C_{thre}^{out}$, つまり全センサを出力に用いる場合と, $0.012 < C_{thre}^{out} < 0.247$, つまり f が使われない場合について考える. これらに対して, ネットワーク入力決定のための L_m を計算した. 前者では, 例えば C_{thre}^{in} を 0.15 程度に設定した場合, L_{110} と L_{011} のみがこれを下回るため, これらマスクに用いられたセンサの和集合である (θ, f, l) 全てが入力として使われた (なお, 出力が (θ, f, l) であるため, L_{111} は計算されない). これは 6.3 節の Musculoskeletal AutoEncoder と全く同じ形である. マスクについては 6.3 節における $\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$ が足りないが, 実際の状態推定や制御では用いていないため, 全く同じオペレーションが可能である. また, C_{thre}^{in} を 0.3 程度に設定した場合, さらにマスクの種類が増え, 6.3 節における全マスクに $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ を加えた計 4 つのマスクが利用可能になる. 後者では, 例えば C_{thre}^{in} を 0.015 程度に設定した場合, L_{100} と L_{001} のみがこれを下回るため, これらマスクに用いられたセンサの和集合である (θ, l) が入力として使われた (なお, 出力が (θ, l) であるため, $L_{\{101, 111\}}$ は計算されない). これは, 筋張力を考慮しない版の Musculoskeletal AutoEncoder であり, この特殊系が, これまで一般的に使われてきた $\theta \rightarrow l$ のネットワークである. つまり, これら閾値の違いのみで, 様々な筋骨格構造に関するネットワークが構築可能である.

Fig. 6.4: Mutual information and loss values in Sec. 3.2.2 for the relationship among (l^{ref}, p) in Sec. 7.3.

最後に 7.3 節の筋骨格構造に関する危険回避を行う静的身体図式について実験を行う。用いるセンサは (l^{ref}, p) の 2 つであり、ランダムな筋長の動きからデータを取得する (なお, p は動作における危険度を表す)。得られたデータに基づく 3.2.2 節におけるそれぞれの損失を Fig. 6.4 に示す。なお, この例では直接のセンサデータではなく $\{0, 1\}$ の確率を扱っているため, 相互情報量による計算が難しく, 不要なセンサ・アクチュエータの削除に関する操作は行わない。まず, ネットワーク出力の決定のための $L_{\{1,2\}}$ を計算した。 L_2 は小さいのに対して, L_1 は大きく, 推論できるとは言いがたい。よって, ネットワークの出力は p のみとなった。次に, ネットワーク入力決定のための L_m を計算した。出力が p のみであるため, $L_{\{01,11\}}$ は計算されない。よって, L_{10} のみが計算され, これが十分小さいと判断し, 入力は l^{ref} となった。よって, $l^{ref} \rightarrow p$ という, 7.3 節における DAN と同じネットワークが構築された。

6.1.2 動的身体図式の入出力自動決定実験

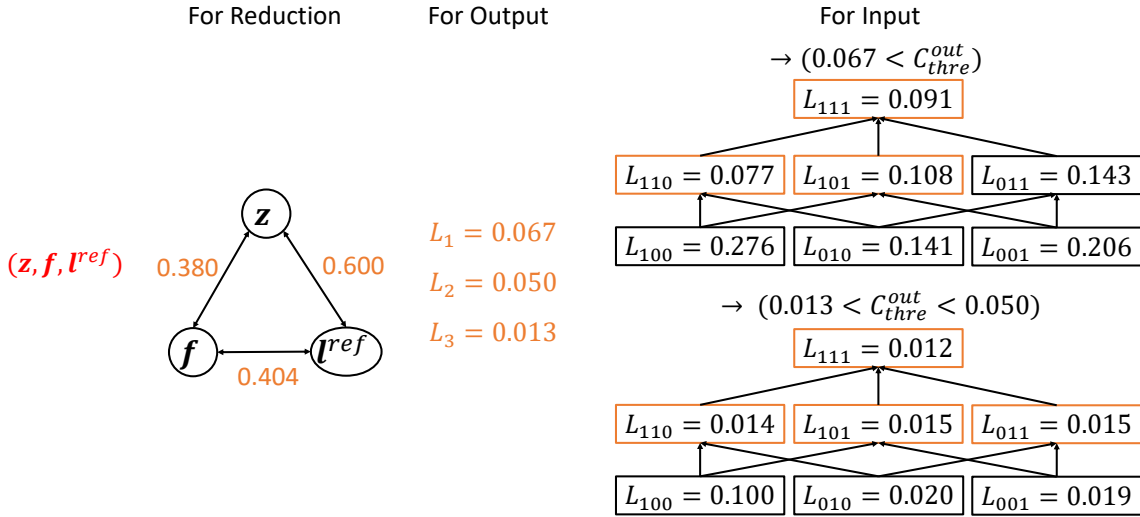


Fig. 6.5: Mutual information and loss values in Sec. 3.2.2 for the relationship among (z, f, l^{ref}) in Sec. 6.4.

まずは 6.4 節の筋骨格ヒューマノイドによる模倣学習について実験を行う。用いるセンサは (z, f, l^{ref}) の 3 つであり, 人間が MusashiLarm を操縦した際のデータを取得する (なお, z は視覚情報を表す)。得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.5 に示す。まず, 不要なセンサ・アクチュエータの削除について, (z, f, l^{ref}) のそれぞれのモデル間の相互情報量が計算され, これが C_{thre}^{reduce} 以上であり, それぞれのセンサが関係し合っていることがわかった。次に, ネットワーク出力の決定のための $L_{\{1,2,3\}}$ を計算した。どれも十分に損失は下がったが, その中でも L_3 が最も低かった。ここで, $0.067 < C_{thre}^{init}$, つまり出力が (z, f, l^{ref}) の全ての場合と, $0.013 < C_{thre}^{init} < 0.050$,

つまり、出力が l^{ref} のみの場合を考える。前者後者それぞれについて、ネットワーク入力決定のための L_m^{init} を計算した（なお、 L_m^{init} と L_m^{seq} が大きく変わらなかったため、 L_m^{seq} は省略している）。前者は入力出力ともに (z, f, l^{ref}) となっており、一般的な模倣学習 $(s_t, u_t) \rightarrow (s_{t+1}, u_{t+1})$ と同じネットワーク構成になっていることがわかる。これは、6.4 節における RNNPB と同じ構成のネットワークである。一方、後者は $(z, f, l^{ref}) \rightarrow (l^{ref})$ となっており、この $(s_t, u_t) \rightarrow u_{t+1}$ も模倣学習の形として良く見られる。また、 L_{110}^{init} は多少ではあるが $L_{\{101,011\}}^{init}$ よりも小さく、同様に模倣学習に見られる $s_t \rightarrow u_{t+1}$ の形も可能であることがわかった。つまり、いくつかの閾値によって、模倣学習における様々なネットワーク構成が再現可能であった。

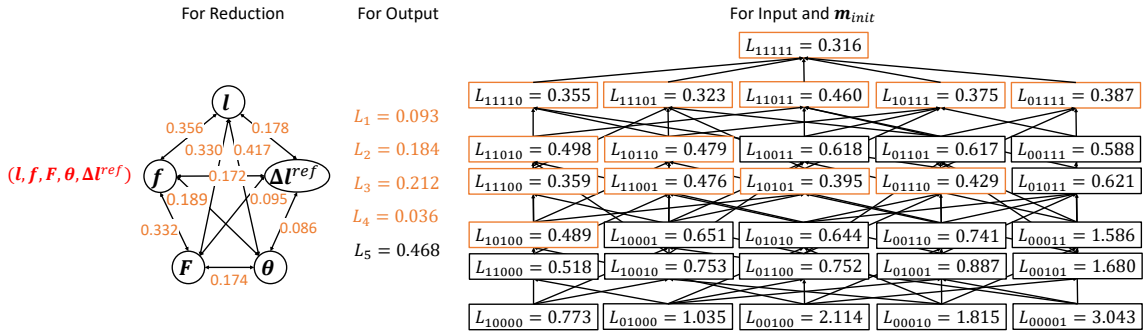


Fig. 6.6: Mutual information and loss values in Sec. 3.2.2 for the relationship among $(l, f, F, \theta, \Delta l^{ref})$ in Sec. 6.5.

次に 6.5 節の筋骨格ハンドにおける動的身体図式について実験を行う。用いるセンサは $(l, f, F, \theta, \Delta l^{ref})$ の 5 つであり、ランダムな筋長の動きからデータを取得する（なお、 F は接触力センサの値 $f_{contact}$ 、 Δl^{ref} は指令筋長速度を表す。また、 Δl と $\Delta \theta$ については簡略化のため除いている）。得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.6 に示す。まず、不要なセンサ・アクチュエータの削除について、 $(l, f, F, \theta, \Delta l^{ref})$ のそれぞれのモデル間の相互情報量が計算され、これが C_{thre}^{reduce} 以上であり、それぞれのセンサが関係し合っていることがわかった。次に、ネットワーク出力の決定のための $L_{\{1,2,3,4,5\}}$ を計算した。 L_5 は $L_{\{1,2,3,4\}}$ に比べて相対的に大きく、推論できるとはいえない。よって、ネットワークの出力は (l, f, F, θ) となった。次に、ネットワーク入力決定のための L_m^{init} を計算した。マスクするセンサは少ない方がより損失が小さく、例えば C_{thre}^{init} を 0.5 と設定した場合では、最小で l と f のみから現在状態を推論することができた。これらに用いるセンサの和集合である $(l, f, F, \theta, \Delta l^{ref})$ 全てのセンサが入力となり、6.5 節と同じネットワークである HADYNET が構築された。これは状態方程式に相当するモデルであり、 Δl^{ref} による (l, f, F, θ) の遷移が表現されている。ここでは m_{seq} については考えなかったが、これを簡易化した 7.7 節の実験で m_{seq} についても

考察する.

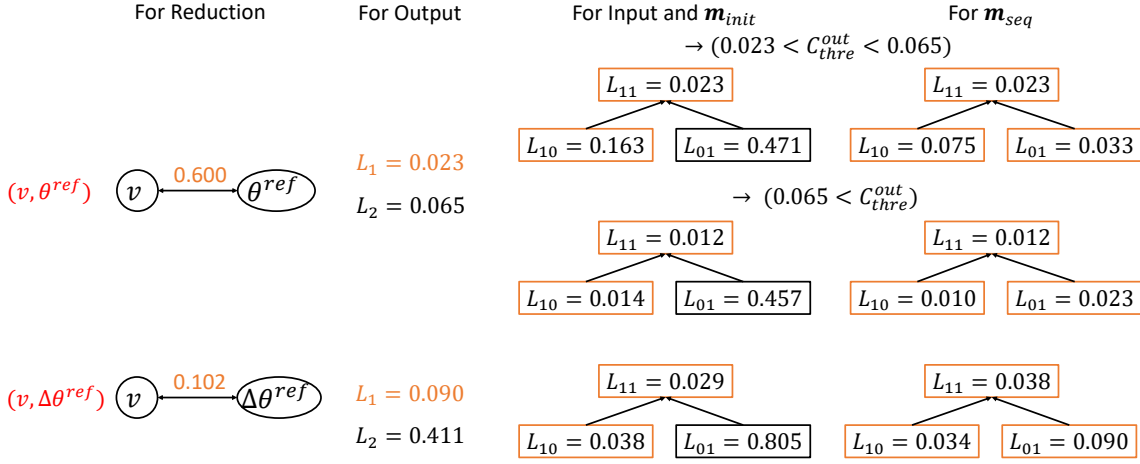


Fig. 6.7: Mutual information and loss values in Sec. 3.2.2 for the relationship among (v, θ^{ref}) or $(v, \Delta\theta^{ref})$ in Sec. 7.6.

次に7.6節の筋骨格ヒューマノイドによるペダル操作について実験を行う。用いるセンサは (v, θ^{ref}) の2つであり、ランダムな足首の動きからデータを取得する(なお、 v は車の速度、 θ^{ref} は右足首の角度を表す。7.6節における θ や $\dot{\theta}$ については簡略化のため除いている)。また、 θ^{ref} について差分を取った $\Delta\theta^{ref}$ を入力として用いた場合についても考察する。得られたデータに基づく3.2.2節におけるそれぞれの相互情報量と損失をFig. 6.7に示す。まず、不要なセンサ・アクチュエータの削除について、 (v, θ^{ref}) または $(v, \Delta\theta^{ref})$ の間の相互情報量が計算され、これが C_{thre}^{reduce} 以上であり、それぞれのセンサが関係し合っていることがわかった。しかし、 $(v, \Delta\theta^{ref})$ については関係性が低く、実際に後の計算では、 (v, θ^{ref}) を使った方が全体的に損失が小さいこともわかる。一方、実際には $\Delta\theta^{ref}$ でもネットワークの順伝播を計算することはできる。つまり、片方のセンサ値を微分することで相互情報量自体は低くなってしまふことが多い。一方ランダムや一定値が出続けるようなセンサ値は相互情報量が0になり、これらはしっかり分類可能であるため、 C_{thre}^{reduce} は低めに設定し、関係のあるセンサを逃さないようにすることが望ましい。次に、ネットワーク出力の決定のための $L_{\{1,2\}}$ を計算した。 (v, θ^{ref}) については、 L_1 も L_2 も同様に小さくなった。ここで、 $0.023 < C_{thre}^{init} < 0.065$ 、つまり出力が v のみの場合と、 $0.065 < C_{thre}^{init}$ 、つまり、出力が (v, θ^{ref}) の場合を考える。これらに対して、ネットワーク入力決定のための $L_m^{\{init, seq\}}$ を計算した。前者・後者に共通で、 $L_{\{11,10\}}^{init}$ は小さくなるものの、 L_{01}^{init} は大きい。一方、 L_{01}^{seq} は $L_{\{11,10\}}^{seq}$ と同じくらい小さい。つまり、ネットワークの入力、または潜在変数 z を計算するセンサとしては v が必ず必要である一方、一度計算された z を更新するには θ^{ref} のみでも良い、という

ことである。これは、 $(v, \theta^{ref}) \rightarrow v$ という状態方程式が成り立つことと一致しており、7.6 節における DDC-Net と同じ構成である。後者のように出力としても θ^{ref} を出すような場合は、模倣学習と同様の形になってしまうが、制御の時は (v_t, θ_t^{ref}) から θ_{t+1}^{ref} を計算する必要はなく、状態方程式の部分だけ取り出してモデル予測制御のように用いても良い。 $(v, \Delta \theta^{ref})$ を用いる場合は、 L_1 に対して L_2 は大きくなり、ネットワーク出力は v のみとなった。 L_m に関する性質は (v, θ^{ref}) を用いたときと大きく変わらない。これは、制御入力を変更する (e.g. 微分する) ことでネットワーク構造が変わる例を表している。データの与え方によって推論のしやすさが変わってくるため、直接の値と微分値を両方ネットワークに入れる、またはデータの与え方による損失の変化や実際の制御挙動を見ながら使用するデータを考察する必要がある。

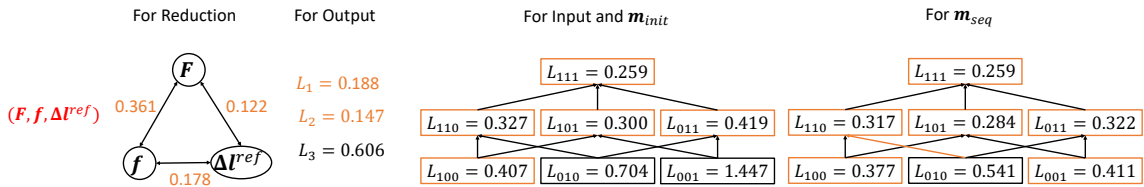


Fig. 6.8: Mutual information and loss values in Sec. 3.2.2 for the relationship among $(F, f, \Delta I^{ref})$ in Sec. 7.7.

次に 7.7 節の筋骨格ハンドによる道具の把持安定化について実験を行う。用いるセンサは $(F, f, \Delta I^{ref})$ の 3 つであり、ランダムな指の筋長の動きからデータを取得する (なお、 F は手のひらに存在する接触力センサの値を表す。また、 I と \dot{I} については簡略化のために除いている)。これは、6.5 節を指の筋だけに適用したより簡素な例である。得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.8 に示す。まず、不要なセンサ・アクチュエータの削除について、 $(F, f, \Delta I^{ref})$ のそれぞれのモダ間の相互情報量が計算され、これが C_{thre}^{reduce} 以上であり、それぞれのセンサが関係し合っていることがわかった。次に、ネットワーク出力の決定のための $L_{\{1,2,3\}}$ を計算した。 L_3 は $L_{\{1,2\}}$ に比べて相対的に大きく、推論できるとは言いがたい。よって、ネットワークの出力は (F, f) となった。次に、ネットワーク入力決定のための $L_m^{\{init, seq\}}$ を計算した。モダを増やすほど精度が良くなっていることがわかる。ここでは先の例と同様に、 L_{001}^{init} は非常に大きい一方、 L_{001}^{seq} は小さい。つまり、ネットワークの入力、または潜在変数 z を計算するセンサとしては $\{F, f\}$ が必要である一方、一度計算された z を更新するには ΔI^{ref} のみでも良い、ということである。これは、 $(F, f, \Delta I^{ref}) \rightarrow (F, f)$ という状態方程式が成り立つことと一致しており、自動的にこの構造がネットワーク内に表現された。

最後に 7.4 節の台車ロボットにおける足回り制御について実験を行う。用いるセンサは (w^{ref}, w) の 2 つであり、Fetch の台車のランダムな走行の動きからデータを取得する (なお、 w は台車の並進・回転

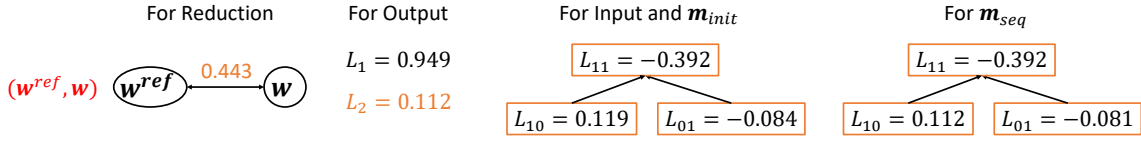


Fig. 6.9: Mutual information and loss values in Sec. 3.2.2 for the relationship among (w^{ref}, w) in Sec. 7.4.

速度, w^{ref} はその指令値を表す). 得られたデータに基づく 3.2.2 節におけるそれぞれの相互情報量と損失を Fig. 6.9 に示す. まず, 不要なセンサ・アクチュエータの削除について, (w^{ref}, w) の間の相互情報量が計算され, これが C_{thre}^{reduce} 以上であり, 2 つのセンサが関係し合っていることがわかった. 次に, ネットワーク出力の決定のための $L_{\{1,2\}}$ を計算した. L_2 は十分に下がったものの, L_1 は下がらず, 推論できるとは言いがたい. よって, ネットワークの出力は w のみとなった. 次に, ネットワーク入力決定のための $L_m^{\{init, seq\}}$ を計算した. なお, 7.4 節では最尤推定に関する損失関数に対して \log を取っているため, マイナスになることがある. \log を取っているため分かりにくい, $L_{\{01, 10, 11\}}^{\{init, seq\}}$ の間に大きな差は無い. よって, 用いるマスクに関するセンサの和集合を取ると, 7.4 節における SPNPB と同じ $(w^{ref}, w) \rightarrow w$ というネットワークが構成された.

6.2 静的道具操作学習 - 身体-道具間関係変化への適応

6.2.1 概要と先行研究

道具操作は人間の本質的な能力の一つである。これまで、ロボットにおける道具使用に必要な要素である、道具認識 [203], 道具理解 [204], 道具選択 [205], 道具操作 [206], 道具生成 [207] に関して様々な研究が行われてきた。その中でも道具操作学習は実際のロボットにおける動作において最も重要である。道具操作にも様々な段階があり、道具の把持、道具と身体的位置関係の理解、道具操作のプランニングがある。道具の把持については、動力学モデルを使う方法 [208, 209] や学習型の方法 [210] が考えられている。道具と身体的位置関係の理解では、手先と道具の間の線形変換またはヤコビアンを求める場合 [211, 212] がほとんどである。道具操作のプランニングには、最適化に基づく動作計画 [213] や、深層学習に基づく動作計画 [214], シンプルな道具軌道の入力と全身逆運動学を使った方法 [215] 等がある。また、これらをひとまとまりにして模倣学習 [216, 33] や強化学習 [217], 自己教師あり学習 [218, 219] により解決する方法も存在する。

しかしそのどれもが道具操作中に逐次的に道具の把持位置・角度等が変化してしまうことを考慮していない。また、変形する道具を扱っている研究も少なく、基本的には剛体の道具が身体に固定された状態のみを扱っている。そこで本研究では、ロボット身体の制御入力と道具の先端位置の間の関係をニューラルネットワークにより学習し、変形する道具等も扱えるような手法を開発する [220] (Fig. 6.10)。Parametric Bias [72, 143] を用いることでセンサ情報からは直接得られない現在の把持状態を逐次的に推定し、それを元に制御入力を変更していく。これらにより、外力により逐次的に変化する把持状態、重くてしなる棒や、ホースから出た水等も扱えるようになると考える。また、筋骨格ヒューマノイド [221, 87] 等の柔軟でより把持状態のモデル化が難しいロボットにも適用を試みる。

Parametric Bias は、模倣学習の文脈においては、暗黙的な道具の違いを埋め込むために利用された例もある [222]。本研究では、直接道具の長さや角度を入力とせず、Parametric Bias を利用することで、データセット作成時の把持状態のアノテーションが必要なくなり、かつ変形する道具や複雑な把持状態を持つロボットを扱うことが可能となる。以降、本研究では把持状態を、Parametric Bias を使った把持位置や把持姿勢等を含む様々な把持状態の暗黙的な表現として定義し用いる。

本研究の代替となり得る手法には主に、(1) 視覚/触覚フィードバックを使った手法、(2) 幾何モデルを使った把持状態推定手法が考えられる。(1) は、道具使用におけるセンサ変化を記憶 [223] または学習 [224] しておくことにより予期せぬ把持状態変化に対してロバストに対応可能な触覚フィードバック、道具先端位置に対する視覚フィードバックが考えられる。(2) は、道具の幾何モデルを使い、手先の

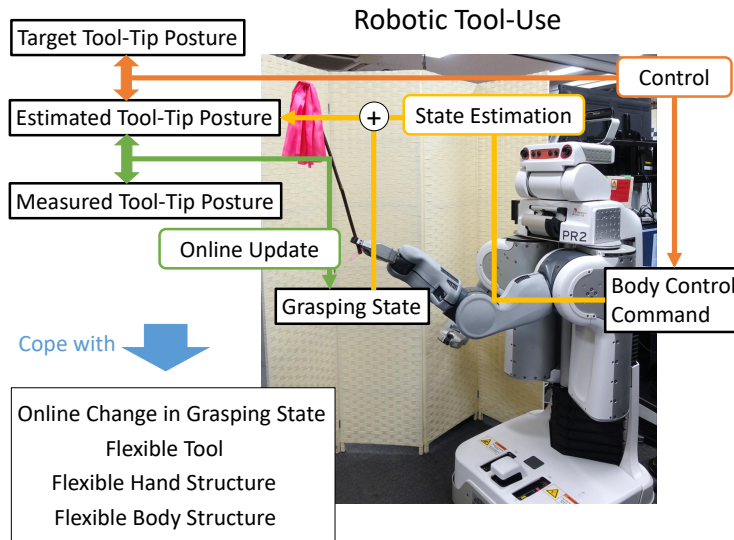


Fig. 6.10: The concept of this study [220]. In robotic tool-use, the tool-tip posture is estimated from the body control command and grasping state, the body control command is calculated from the loss between the target and estimated tool-tip postures, and grasping state is updated online from the loss between the estimated and measured tool-tip postures. This study can also cope with the online change in grasping state and flexible tool, hand, and body structures.

位置と道具先端位置の関係から道具の把持位置や角度を割り出すシンプルな方法である。(1)は把持状態を推定せずにセンサフィードバックによりそれらを補償する方法,(2)は幾何モデルから把持状態を理解して道具を用いる方法とすることができる。しかし,(1)は操作したい状態と制御入力のコビアンが自明ではない柔軟道具や複雑なハンド・ロボット構造等には対応できない。また、教示した動作への追従を主としており、道具・把持状態のモデル化が行われないため、本研究とは適用範囲が大きく異なる。この他にも、模倣学習[225]によるセンサフィードバックを使った道具先端制御もあるが、道具の把持状態変化に適応した例はない。(2)は幾何モデルを前提としているため柔軟道具や複雑な把持状態等を扱うことができない。これらに対して、本研究はニューラルネットワークによる身体と道具の関係のモデル化・暗黙的な把持状態の考慮により、複雑で柔軟な身体・ハンド・道具等に適用可能な汎用的なモデルを提供する。本研究のネットワークは、静的な一般化多感覚相関モデルにおいて、入力の一意性から m を消した形となる。

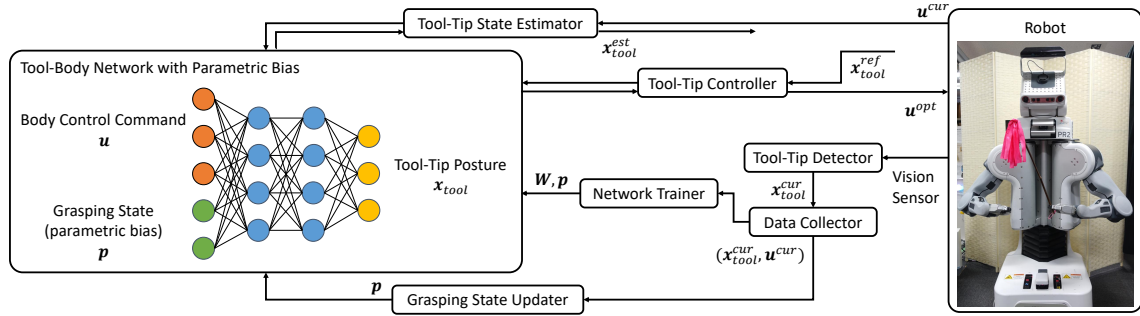


Fig. 6.11: The overall software system [220]: the network structure of TBNPB, network trainer of TBNPB, online grasping state updater through parametric bias, tool-tip state estimator, and tool-tip controller.

6.2.2 逐次的な把持状態変化を考慮した適応的道具先端操作学習

本研究では、道具の先端と身体の制御入力間の静的な関係を表す Parametric Bias を含んだネットワークを、Tool-Body Network with Parametric Bias (TBNPB) と呼ぶ。TBNPB を囲む本研究の全体システムを Fig. 6.11 に示す。

TBNPB のネットワーク構造

TBNPB のネットワーク構造は非常に単純で、以下のように表される。

$$\mathbf{x}_{tool} = \mathbf{h}(\mathbf{u}, \mathbf{p}) \quad (6.1)$$

ここで、 \mathbf{x}_{tool} は道具の先端位置、 \mathbf{h} は TBNPB のネットワーク、 \mathbf{u} はロボット身体の制御入力、 \mathbf{p} は Parametric Bias (PB) とする。 \mathbf{x}_{tool} は位置姿勢を表すことも可能だが、本研究では 3 次元の位置とする。 \mathbf{u} は本研究では関節角度の指令値 θ^{ref} を表すこととする。Parametric Bias は元々再帰的ニューラルネットワークと一緒に用いる場合がほとんどであるが、本研究では静的な対応付けのネットワークにこの Parametric Bias を用いている。

本研究では、ネットワーク構造は 7 層とし、ユニット数については、入力は \mathbf{u} と \mathbf{p} を合わせた次元数 (ロボットによって異なる)、中間の 5 層は全て 300、出力は \mathbf{x}_{tool} の 3 次元とした。活性化関数は Tanh、更新則は Adam [46] とする。ネットワークの入力と出力は訓練時に得られたデータを使って正規化されている。

TBNPB の学習

本項は Fig. 6.11 の Data Collector, Network Trainer の説明である。まず、道具の持つ角度や持つ位置が異なるような、様々な把持状態 k ($1 \leq k \leq K$, K は訓練に使う全把持状態数) について、様々な姿勢におけるデータ $D_k = \{(u, x_{tool})_1, \dots, (u, x_{tool})_{N_k}\}$ を収集する (N_k は把持状態 k に関するデータ数)。また、それぞれの把持状態 k について Parametric Bias p_k を用意する (全ての p_k は 0 に初期化されている)。よって、データ $D_{train} = \{(D_1, p_1), \dots, (D_{N_k}, p_{N_k})\}$ が収集され、この D_{train} を用いて h を学習させる。このとき、 p_k はデータ D_k については共通であり、異なる把持状態については異なる変数とする。学習の際はネットワークの重み W と同時に p_k も誤差逆伝播法によって更新する。これにより、 p_k に把持状態に関する情報が埋め込まれる。

本研究では学習は2段階で行う。まず、シミュレーション上で把持位置や把持姿勢を変化させてデータを収集し、 W と p_k を計算する。その後、シミュレーションで計算された W のみ残し、 p_k を 0 に初期化する。最後に、実機においてデータを収集し、学習を行う。実機で得られるデータは少数なため、Fine-Tuning により対応する。

把持状態のオンライン更新

本項は Fig. 6.11 の Grasping State Updater の説明である。把持状態が常に変化しうることを想定し、オンラインで Parametric Bias p を更新する。道具の先端位置 x_{tool} が認識できているかつ、制御入力 u が直前に収集された制御入力 u^{prev} と離れている、つまり $\|u - u^{prev}\|_2 > C_{collect}$ の場合に、データを収集する ($\|\cdot\|_2$ は L2 ノルム, $C_{collect}$ は閾値とする)。得られたデータ数 N_{online} が、 N_{thre}^{online} を超えてから学習を開始し、その後新しいデータが収集される度に学習を行う。重み W を固定し、 p のみをバッチ数 N_{batch}^{online} 、エポック数を N_{epoch}^{online} として更新する。この際の更新則は、学習率を 0.1 とした Momentum SGD とする。データは N_{max}^{online} ($N_{thre}^{online} \leq N_{max}^{online}$) を最大値とし、それを超えたデータは古いものから順に削除していく ($N_{online} \leq N_{max}^{online}$)。ネットワークの重み W を固定し、小さな次元である Parametric Bias のみ更新することで、過学習を防ぎつつ把持状態のみを更新することができる。

本研究では、 $C_{collect} = 10.0$ [deg], $N_{thre}^{online} = 10$, $N_{batch}^{online} = N_{epoch}^{online} = 3$, $N_{max}^{online} = 20$ とした。また、データ収集のサンプリングレートは 5 Hz である。 $C_{collect}$ は動作全体のスケールに応じて適切に設定すべきである。 N_{thre}^{online} は大きい方が学習初期は安定するが、その分学習し始めるのが遅くなり、基本的には大きめに設定すべきである。 N_{max}^{online} は大きいほど多くのデータを使って正確に把持状態を更新できるものの、把持状態変化への適応には遅くなるため、そのトレードオフを加味して適切に設定すべきである。

TBNPB を使った道具先端位置の推定と制御

本項は Fig. 6.11 の Tool-Tip State Estimator / Controller の説明である. 道具先端位置の推定は非常に単純であり, 現在の \mathbf{p} と制御入力 \mathbf{u} を \mathbf{h} に入力するのみで計算することができる. 道具先端位置の制御は誤差逆伝播法と勾配降下を使った最適化により行う. まず, 現在の制御入力 \mathbf{u}^{cur} を取得し, これを最適化する制御入力 \mathbf{u}^{opt} の初期値とする. 次に, 以下のように最適化を行う.

$$\mathbf{x}_{tool}^{est} = \mathbf{h}(\mathbf{u}^{opt}, \mathbf{p}) \quad (6.2)$$

$$L = \|\mathbf{x}_{tool}^{est} - \mathbf{x}_{tool}^{ref}\|_2 + \alpha L_{const}(\mathbf{u}^{opt}) \quad (6.3)$$

$$\mathbf{u}^{opt} \leftarrow \mathbf{u}^{opt} + \gamma \partial L / \partial \mathbf{u}^{opt} \quad (6.4)$$

ここで, \mathbf{x}_{tool}^{ref} は道具先端位置の指令値, L_{const} は制御入力 \mathbf{u} に関する制約を表し, α は損失関数の重み, γ は学習率を表す. L_{const} については, 例えば \mathbf{u} を現在の制御入力になるべく近い形にしたければ $\|\mathbf{u}^{opt} - \mathbf{u}^{cur}\|_2$ にしたり, ある一つの関節を動かしたくなければ, その関節のみに対して損失を与えて制約をかけたりすることが可能である. γ は, 本研究では Line Search によって, 0 から γ^{max} までの $N_{batch}^{control}$ 種類の γ を試し, 最も L が小さかったものを採用することを $N_{epoch}^{control}$ 回繰り返す (γ^{max} は γ の最大値とする). 最終的に得られた \mathbf{u}^{opt} を使い, 道具の先端位置を制御することができる.

本研究では, $\gamma^{max} = 0.5$, $N_{batch}^{control} = 30$, $N_{epoch}^{control} = 10$ とする.

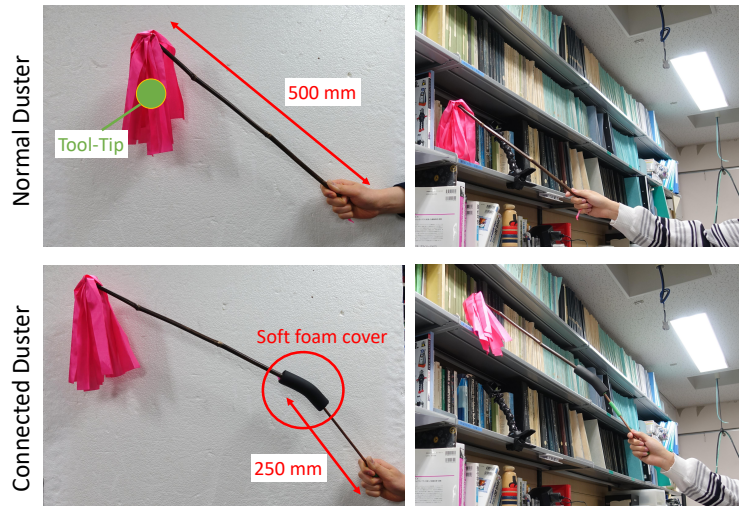


Fig. 6.12: The tools used in this study [220]: normal and connected dusters.

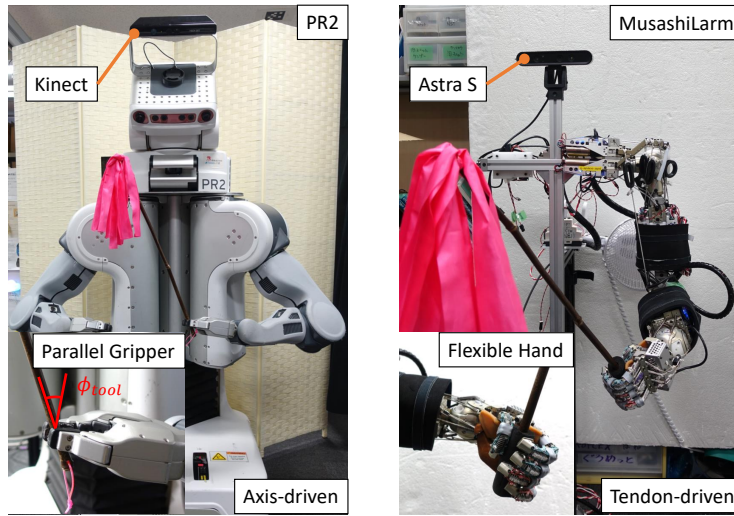


Fig. 6.13: The robots used in this study [220]: PR2 with the parallel gripper and the musculoskeletal arm MusashiLarm with the flexible hand.

6.2.3 実験

実験セットアップ

本研究では、掃除用具の一つであるはたきを使った実験を行う (Fig. 6.12). はたきはその道具の先端位置を制御することで、棚や隙間のほこりを落とす道具である。はたきの先端には色のついた布が付いており、道具の先端位置はこの色を抽出することで認識する。このはたきの布は持つ角度によって棒の先端から重力方向に下がり、手先位置姿勢から道具先端位置を線形変換することはできない。また、より難しい問題として、追加の棒を括りつけることで長さを増したはたきも PR2 実験において利用する。はたきと追加の棒は柔軟な発泡性の素材により接続し、はたきを持つ角度によって道具先端位置が大きく変化するようにになっている。なお、本研究で使うはたきは棒の長さが 500 mm, 布は 200 mm, 追加の棒は 250 mm であり、通常のはたきを Normal Duster, 追加の棒により接続された長いはたきを Connected Duster と呼ぶ。

本研究の実験では、台車型ロボット PR2 のシミュレーションと実機、筋骨格ヒューマノイド MusashiLarm [87] の実機を用いて実験を行う (Fig. 6.13). PR2 / MusashiLarm においては、頭部に Kinect (Microsoft, Corp.) / Astra S (Orbbec 3D Technology International, Inc.) の深度カメラがついており、色抽出した道具の先端の点群を Euclidean Clustering し、一番大きな Cluster の中心位置を道具の先端位置とした。PR2 のハンドは平行グリップ型であり、はたきの把持の際は、主に平行グリップと垂直方向に関する角度 ϕ_{tool} , 道具を持つ位置が変化する。これに対して、MusashiLarm のハンドは切削ばねを使った柔軟ハン

ドであり, 把持状態をパラメータ化することは難しい.

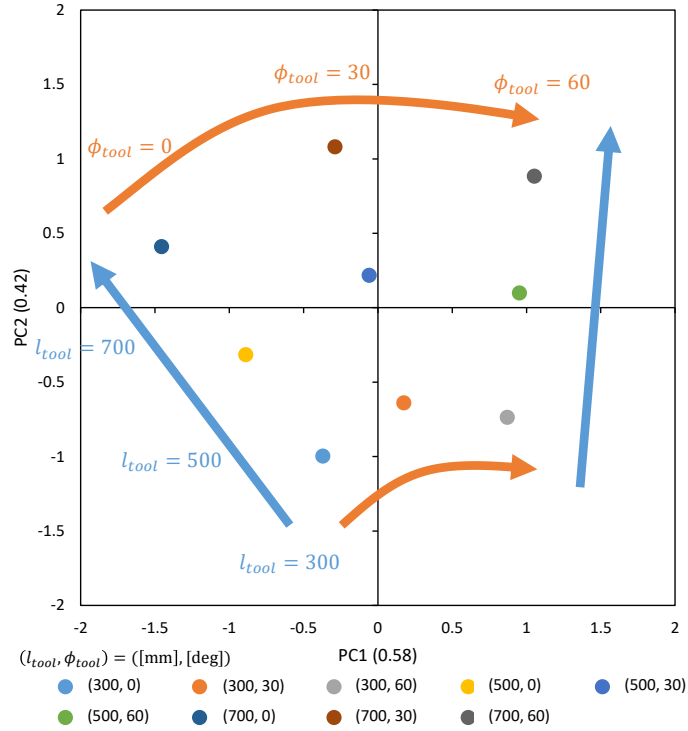


Fig. 6.14: Parametric bias trained in PR2 simulation experiment [220].

PR2 によるシミュレーション実験

PR2 の幾何シミュレータを使って実験を行う. まず, 擬似的に真っ直ぐな棒をはたきとして用意し, これを把持位置 (手先からの道具の長さで表現) l_{tool} , 把持角度 (平行グリッパと垂直方向の角度を一自由度で表現) ϕ_{tool} を 3 種類ずつ変化させながらデータを取得する. はたきの布は棒の先端から重力方向に垂れるため, 道具先端位置は棒の先端から z 方向に -100 mm の場所としてシミュレーションを行う. $l_{tool} = \{300, 500, 700\}$ [mm], $\phi_{tool} = \{0, 30, 60\}$ [deg] とした. 次に, 関節角度範囲を決め, その中で, それぞれの把持状態について関節角度をランダムにサンプリングし, データ D_{train} を取得する. 得られた D_{train} は合計で 9000 個であり, これをバッチ数 300, エポック数 300 として学習させる. なお, \mathbf{u} は 7 次元であり, \mathbf{p} は 2 次元とした. このときに得られた Parametric Bias p_k を PCA を通して 2 次元空間に表現した図を Fig. 6.14 に示す. l_{tool} と ϕ_{tool} の大小に伴って, それぞれの PB が規則的に整列していることがわかる. l_{tool} が大きいほど ϕ_{tool} の変化による PB の違いが大きくなっており, より長い道具ほど角度によって大きく道具先端位置が変化する点と一致している.

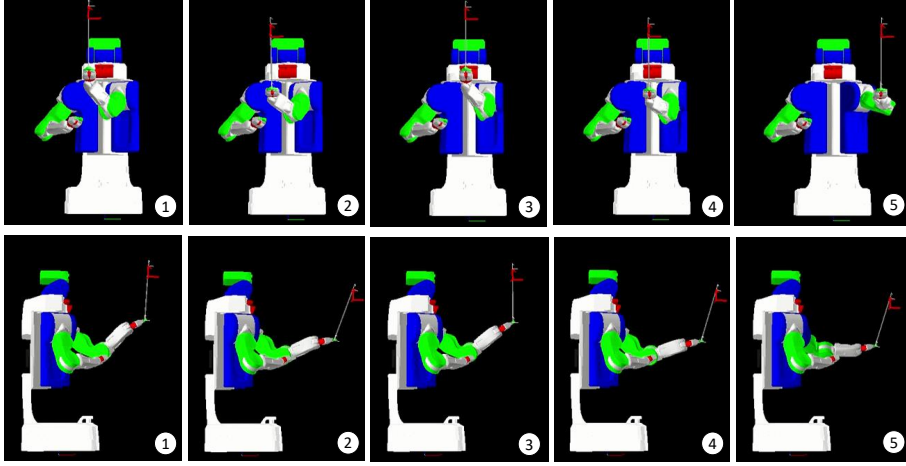


Fig. 6.15: Duster-use motion for PR2 experiments [220].

次に, Parametric Bias のオンライン学習と道具先端位置推定の挙動について実験する. 道具を (1) $(l_{tool}, \phi_{tool}) = (500, 60)$ から $(l_{tool}, \phi_{tool}) = (500, 0)$ に変化させた場合, (2) $(l_{tool}, \phi_{tool}) = (700, 30)$ から $(l_{tool}, \phi_{tool}) = (300, 30)$ に変化させた場合の2つの場合について実験を行う. はたきを振る動作は, Fig. 6.15 に示すようなものである ($(l_{tool}, \phi_{tool}) = (500, 30)$ の道具を持った場合). 道具先端位置の基準点を決め (PR2 のベース中心を原点とする), $(l_{tool}, \phi_{tool}) = (500, 30)$ であれば例えば $(800, -100, 1600)$ [mm], y 方向に一回 100 mm 進み, (x, z) 方向に一回 $(200, -200)$ [mm] 逆運動学を解いて動かして戻す, を交互に行う動作である. y 方向には, 500 mm 進んだら 100mm ずつ戻る. この動きの際の Parametric Bias の動きを Fig. 6.16 に, 道具先端位置の推定誤差の遷移を Fig. 6.17 に示す. (1), (2) の両者について, Parametric Bias は訓練時に得られた現在の把持状態周辺に徐々に近づいていることがわかる. また, それに伴い, 道具先端位置の推定誤差も徐々に下がっていくことがわかる. データが 20 以上集まった時における推定誤差の平均は (1) が 52.2 mm, (2) は 25.9 mm であった.

最後に, 道具先端位置の制御について実験する. Parametric Bias が訓練時に得られた $(l_{tool}, \phi_{tool}) = (500, 30)$ である状態から始め, $(l_{tool}, \phi_{tool}) = (500, 60)$ としたときに, Grasping State Updater を使った場合 (update \mathbf{p}) と, \mathbf{p} は固定して \mathbf{W} を更新した場合 (update \mathbf{W}) での制御誤差を比較する. 前者は \mathbf{p} のみを更新するのに対して, 後者は通常のオンライン学習のように, \mathbf{p} は存在せずに重み \mathbf{W} を更新していくことに相当する (なお, 後者は学習率を 0.01 とした). 動作は Fig. 6.15 と同じ動作である. このとき, $(l_{tool}, \phi_{tool}) = (500, 30)$ として生成された Fig. 6.15 の関節角度 \mathbf{u}^{orig} を基準とするため, $\alpha = 0.3$, $L_{const} = \|\mathbf{u}^{opt} - \mathbf{u}^{orig}\|_2$ とする. 道具先端位置の制御誤差の遷移を Fig. 6.18 の左図に示す. online updater が働かない初期は制御誤差が約 240 mm なのに対して, online updater によって制御誤差が大

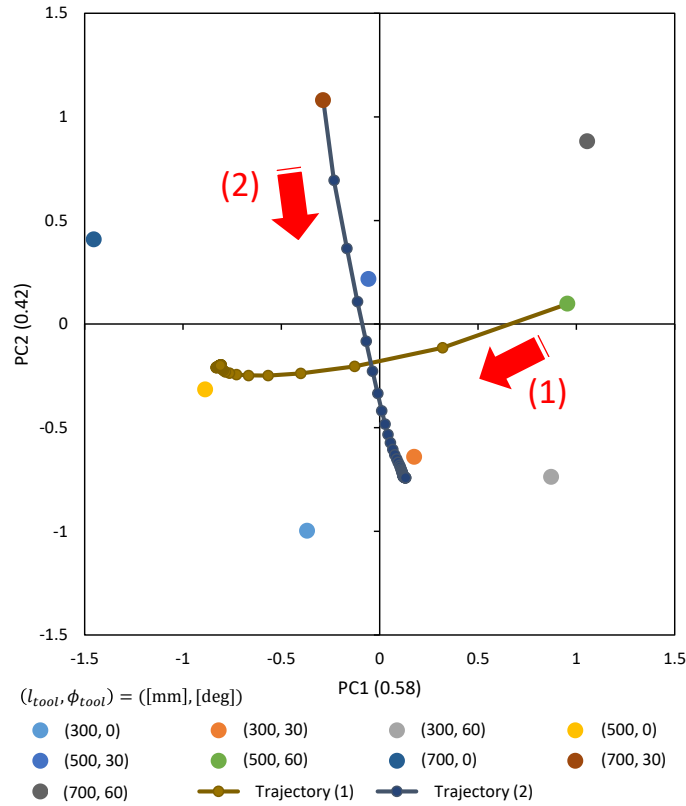


Fig. 6.16: Transition of parametric bias when changing grasping state [220]: (1) $(l_{tool}, \phi_{tool}) = (500, 60) \rightarrow (500, 0)$, (2) $(l_{tool}, \phi_{tool}) = (700, 30) \rightarrow (300, 30)$ in PR2 simulation experiment with grasping state updater.

きく下がっていることがわかる。得られたデータ数が 20 以上となったとき、制御誤差の平均は、前者で 31.5 mm、後方で 19.2 mm であり、ネットワーク全体を更新する後者の方が精度が良かった。その後 online updater を止め、道具先端の回転制約を変えて全く異なる u^{orig} で同じ道具先端位置軌道の動作を行った際の制御誤差の遷移を Fig. 6.18 の右図に示す。 p のみ更新した後では、制御誤差は平均で 22.6 mm なのに対して、 W を更新した後では 207 mm であった。 p のみを更新した場合は他の関節角度においても把持状態更新が効果を発揮するのに対して、 W を更新した場合は訓練に使ったデータに過学習してしまい、他の関節角度においては制御誤差が大きくなってしまったことがわかった。

PR2 による実機実験

PR2 の実機を使った実験を行う。前述のシミュレーションで行った Fig. 6.15 の動作を、基準点を変えながら 3 回行いデータを取得する。把持状態を変化させながら上記を繰り返し、得られた約 1500 のデータを使い、バッチ数を 30、エポック数を 100 として TBNPB を学習させる。このとき、前述のシ

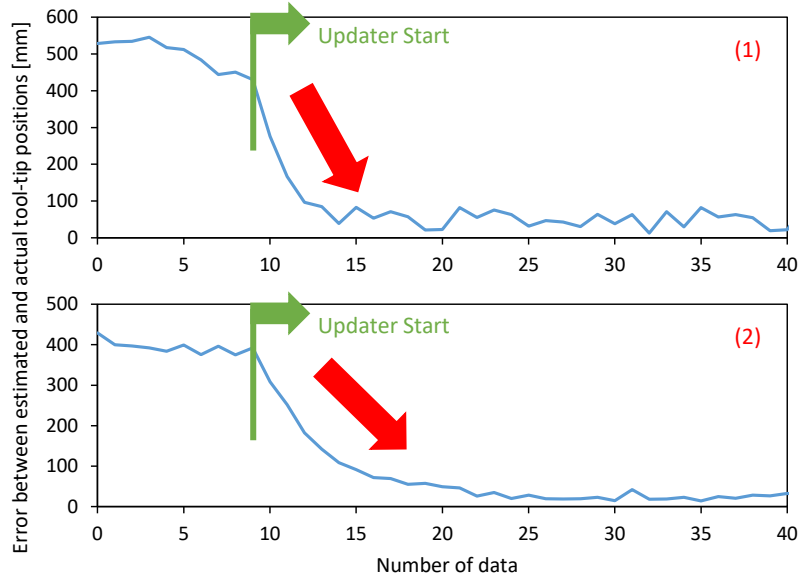


Fig. 6.17: Transition of state estimation error of tool-tip position when changing grasping state [220]: (1) $(l_{tool}, \phi_{tool}) = (500, 60) \rightarrow (500, 0)$, (2) $(l_{tool}, \phi_{tool}) = (700, 30) \rightarrow (300, 30)$ in PR2 simulation experiment with grasping state updater.

ミュレーションで得られたモデルを Fine-Tuning する。また、把持状態は暗黙的に学習されるため、人間が目分量で、道具を長く持った状態 (long) と短く持った状態 (short), $\phi_{tool} = \{0, 30, 60\}$ の状態を作り、データを収集している。学習によって得られた Parametric Bias の分布を Fig. 6.19 に示す。Fig. 6.14 と似てはいるが異なる形で, long/short, $\phi_{tool} = \{0, 30, 60\}$ の軸に沿って PB が分布していることがわかる。また、Fig. 6.18 と同様の形で道具先端制御に関する実験を行った結果を Fig. 6.20 に示す。初期は把持状態がわからないため、制御誤差は約 350 mm と大きいですが、その後データ数が N_{thre}^{online} を超えて Grasping State Updater が実行されてから一気に制御誤差が減り、制御誤差は約 100 mm まで下がっていることがわかる。その後人間が手で道具に力を加え把持状態を変化させているが、制御誤差に大きな変化は無かった。このときの PB の遷移は Fig. 6.19 の “trajectory” の通りであり、(1) が Grasping State Updater 開始後、(2) が把持状態の変化後の遷移である。把持状態の変化を検知して自動で PB が更新されていることがわかる。

次に、Connected Duster を使った実験を行う。先の実験と同様にデータを取得して学習し、その際に得られた Parametric Bias を Fig. 6.21 に示す。Connected Duster は持つ角度によって道具自体が大きく曲がるため、Fig. 6.19 と比較すると、Parametric Bias が short/long に比べて ϕ_{tool} によって大きく変化する形となっている。先と同様の道具先端制御実験を行った結果を Fig. 6.22 に示す。初期の制御誤差

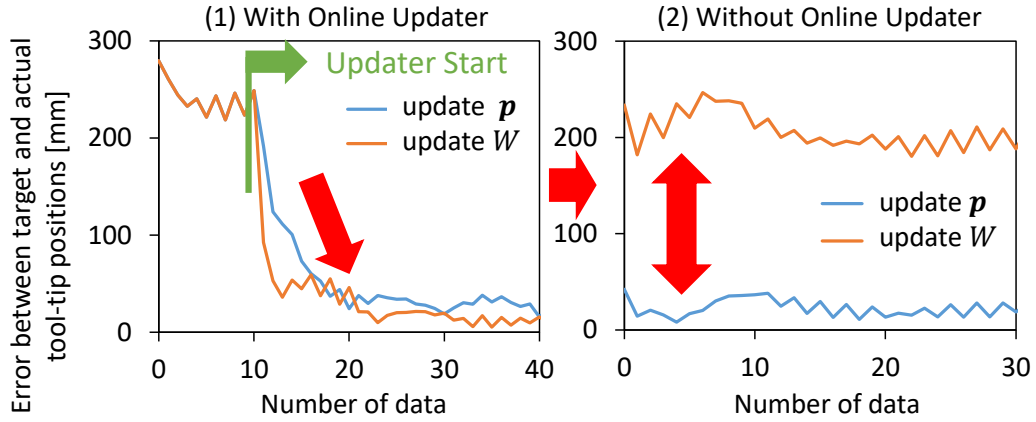


Fig. 6.18: Transition of control error of tool-tip position when changing grasping state [220]: $(l_{tool}, \phi_{tool}) = (500, 30) \rightarrow (500, 60)$ in PR2 simulation experiment with online updaters updating \mathbf{p} or \mathbf{W} . The right experiment is conducted at a different posture from the trained one without online updaters.

は約 1000 mm と非常に大きいですが、徐々に把持状態が分かっていき、150 mm 程度まで誤差が減っている。その後人間が道具に外力を加えて把持状態を変更すると 500 mm 程度まで制御誤差は増えるが、Grasping State Updater によりまた 200 mm 程度まで下がった。このときの PB の遷移は Fig. 6.21 の “trajectory” の通りであり、(1) が Grasping State Updater 開始後、(2) が把持状態の変化後の遷移である。把持状態の変化を検知して自動で PB が更新されていることがわかる。

最後に、Normal Duster を使って行った実際のはたき動作を Fig. 6.23 に示す。壁の物品に接触するような道具先端位置指令を使ってはたき動作を行う。最初は把持状態の認識が正しくないためはたきが物品に当たらないが、更新を行うことで正しくあたり、埃を落とすことができた。

筋骨格ヒューマノイド MusashiLarm による実験

MusashiLarm の実機を使った実験を行う。ここでは PR2 と同様にまず MusashiLarm の幾何シミュレータによりデータを収集し学習させる。この際、 $l_{tool} = 500$ [mm] で固定し、 ϕ_{tool} と垂直な方向に関する道具の角度 ψ_{tool} を定義し、 $\phi_{tool} = \{0, 30, 60\}$ [deg]、 $\psi_{tool} = \{-30, 0, 30\}$ [deg] としてデータを収集した。なお、 \mathbf{u} は 5 次元 (手首 2 次元は入れていない) であり、 \mathbf{p} は 2 次元とした。その後、PR2 と同様に実機によりデータを取得し (この際、関節角度指令は [125] により筋長に変換して行われる)、TBNPB を Fine-Tuning した際の Parametric Bias を Fig. 6.24 に示す。PR2 と違い把持状態が非常に複雑であるため、学習に用いた人間が適当に作成した把持状態を `grasp-{1, 2, 3, 4}` と表記している。この TBNPB を使って PR2 と同様に道具先端位置制御実験を行った結果を Fig. 6.25 に示す。TBNPB を使

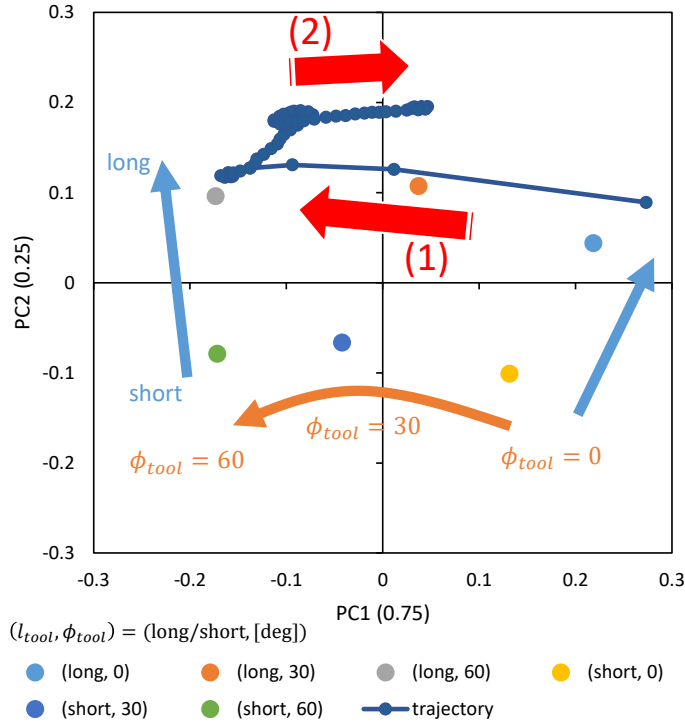


Fig. 6.19: Parametric bias trained in PR2 experiment with the normal duster and its trajectory in the tool-tip control experiment with grasping state updater [220].

わず $\phi_{tool} = 30, \psi = 0$ とした幾何モデルを使った際は約 410 mm 程度の制御誤差があるのに対して, TBNPB を使うことで制御誤差が 260 mm 程度まで下がっている. また, さらに Grasping State Updater が加わることで把持状態が更新され, 制御誤差は 120 mm 程度まで下がった. その後道具に外力を加えることで把持状態が変化し, 制御誤差は大きく上昇するが, Grasping State Updater により, また 180 mm 程度まで下がった.

6.2.4 議論

実験から得られた結果について議論する. PR2 のシミュレータ実験からは, 規則的に特性ごとに PB が自己組織化されることがわかった. それらは道具が長いほど角度の違いによる道具先端位置の変化が大きいというような事実と一致した形で自己組織化される. また, Grasping State Updater により, PB が現在の把持状態の Parametric Bias 周辺に遷移していくことがわかった. これにより, 状態推定・道具先端位置制御が Grasping State Updater がいない場合に比べて正確になっていくことがわかった. また, 本研究のように p のみを更新するのではなく, 通常のオンライン学習のようにネットワーク全体

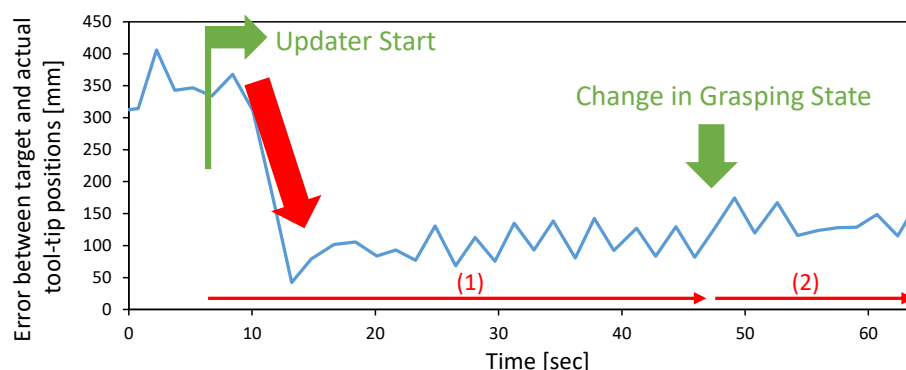


Fig. 6.20: Transition of control error of tool-tip position in tool-tip control experiment of PR2 with the normal duster [220].

を更新した場合は、学習したデータ付近についてはより正確に道具先端位置制御が可能となるものの、過学習が起き、学習していないデータについては、制御誤差が大きくなってしまふことがわかった。本研究の Grasping State Updater はネットワーク全体ではなく把持状態のみを更新するため、学習していないデータについても制御誤差を減少させることが可能である。PR2 の実機においても同様の傾向が見られ、Grasping State Updater による把持状態の更新、それによる制御誤差の減少が可能であった。柔軟な道具である Connected Duster を用いた実験でも同様の結果が得られ、剛体の道具だけでなく、変形する道具にも適用可能であることがわかった。最後に MusashiLarm を使った実験では、把持状態がより曖昧かつ、身体が柔軟な系を扱った。身体が柔軟であるため指令した関節角度を正確に実現できるわけではないため、TBNPB を使わない場合は非常に制御誤差が大きいが、TBNPB を使うことで制御誤差が減り、さらにその曖昧な把持状態を更新することでより制御誤差が減ることがわかった。つまり、把持状態が定義できないような系や、関節角度が正確に実現できないような柔軟系にも適用することができる。また、データの取り方次第では柔軟身体に特有な初期化における再現性の無さや経年劣化についても、Parametric Bias に含めることが可能であると考えられる [146]。よって、剛な軸駆動系、柔軟な腱駆動系、剛な道具、変形する道具、様々なハンドに対して、道具先端位置の推定・制御・把持状態の更新が可能であることがわかった。

本研究の主な限界は (1) データ取得、(2) 道具の種類、(3) 制御誤差である。(1) について、現状道具ごとにそれぞれデータを取得しないといけないため、道具の種類についてスケールしない形になっている。一方、道具の種類についても把持状態と同様に Parametric Bias に埋め込むことは可能はずである。この場合、道具の種類 × 様々な把持状態という大量のデータを取得する必要があるが、それらの内分点にあたるような道具や把持状態は推論可能になると考える。また、シミュレーションの活

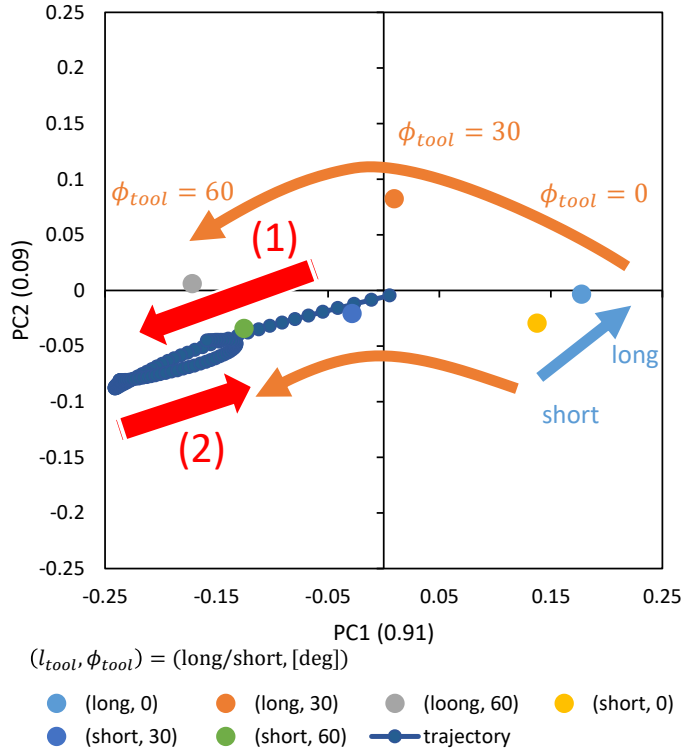


Fig. 6.21: Parametric bias trained in PR2 experiment with the connected duster and its trajectory in the tool-tip control experiment with grasping state updater [220].

用による大量の事前データの取得も一つの方向性である。(2)について、現状剛体や弾性体の道具は使うことができるが、溶解や破損を伴う道具の利用は難しい。剛体や弾性体のように、重力や道具の構造によって \mathbf{u} から \mathbf{x}_{tool} が推論可能な道具は本研究と同様に扱うことができる。一方、溶解や破損のように、変形によって、 \mathbf{u} から \mathbf{x}_{tool} が一意に定まらなくなってしまう道具については、それらの変化が \mathbf{p} に埋め込まれることになる。この場合、道具の変化と把持状態が同じ \mathbf{p} に埋め込まれてしまうため、使い勝手の良くないネットワークになってしまう可能性が高く、今後検証していく必要がある。(3)について、はたきは道具先端位置が不定形であり観測誤差が大きいため実験結果の誤差もそれなりに大きい。また、そこまで正確な操作を必要とする道具ではないため問題は無かったが、ドリルやノコギリ等を扱う際は制御誤差が大きな問題となる。本研究のネットワークの推論精度は主に、道具先端の認識精度と、動作全体における道具先端の動きの大きさに依存する。動作全体が小さければ小さいほど、相対的に道具先端位置に対する推論精度が上がるため、同様にドリル等も扱うことができると考えている。本手法は身体と道具の関係自体を学習するため、一度把持状態が更新されれば、後はよそ見をして別の方向を見ることも可能であり、これが視覚フィードバックとの違いである。しかし、より精

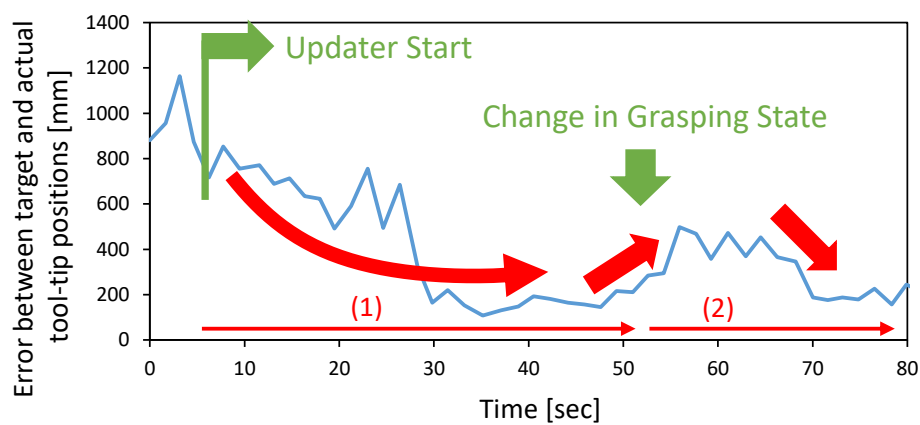


Fig. 6.22: Transition of control error of tool-tip position in tool-tip control experiment of PR2 with the connected duster [220].



Fig. 6.23: Duster-use experiment of PR2 [220].

密な動作を行う場合, TBNPB である程度の道具制御を行い, その後視覚フィードバックと併用して用いることも考える必要がある.

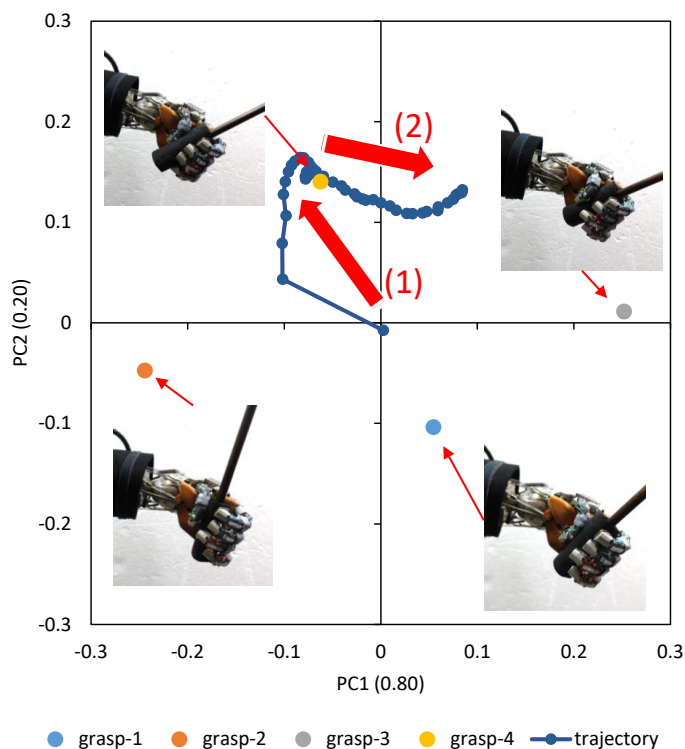


Fig. 6.24: Parametric bias trained in MusashiLarm experiment and its trajectory in the tool-tip control experiment with grasping state updater [220].

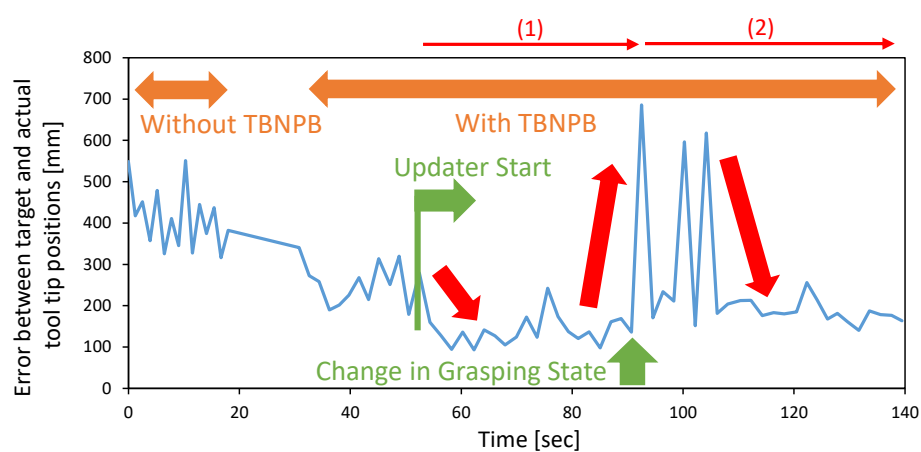


Fig. 6.25: Transition of control error of tool-tip position in the tool-tip control experiment of MusashiLarm [220].

6.3 静的筋骨格身体動作学習 - 関節角度-筋張力-筋長間関係への適用

6.3.1 概要と先行研究

複雑な身体を持ちモデル化困難な筋骨格ヒューマノイドに対して、これまで多くの学習型の制御手法や状態推定手法が開発されてきた。それらの研究の中でも特に実機を扱ったものを Fig. 6.26 にまとめる。まず状態推定について、筋骨格ヒューマノイドは球関節等の存在から一般的には関節角度センサを有さないため、関節角度を推定する手法が開発されている。大久保らは、慣性センサ等から得たデータを使って関節と筋の関係を多項式近似により表し、拡張カルマンフィルタを用いて実機関節角度を推定している [122]。中西らは、関節と筋の関係を表すデータテーブルを用いてマッチングを行うことで、筋長センサから実機の関節角度を推定している [126]。大久保らの方法は、関節と筋の関係をニューラルネットワークを用いて表現した場合にも適用することが可能である。次に制御手法について述べる。茂木らは、手先位置と筋長を対応付けるテーブルを構築し、正確な手先位置を実現する手法を提案している [127]。水内らは、モーションキャプチャデータから関節と筋の関係をニューラルネットワークで表現し、身体制御を行っている [128]。河原塚らは、関節と筋の関係を表すニューラルネットワークをオンラインで学習し身体制御を行う手法 [100]、さらにそれを拡張し、筋張力の影響による身体組織の柔軟性を考慮した手法を開発している [124]。その他にも、強化学習を用いる手法 [129] があるが、実機での使用は難しい。また、筋張力による関節トルク制御手法 [226, 188, 123] も存在するが、これらは筋張ヤコビアンのみ必要なため本手法を用いても行うことができる。しかし、摩擦等の関係から実機における適用においては、良い結果は出ていない。また、学習した結果を状態推定や制御に用いることはあっても、シミュレーションに対して適用した手法はない。補足として、作成した幾何モデルを用いたシミュレーションはいくつか存在する [134, 135, 136]。

	State Estimation	Control	Simulation
Polynomials	Ookubo, et al.		
Data Table	Nakanishi, et al.	Motegi, et al.	
Neural Network	Ookubo, et al.	Mizuuchi, et al.	Offline Learning
		Kawaharazuka, et al.	Online Learning
	+ anomaly detection	This Study	

Fig. 6.26: Classification of previous studies for control and state estimation of musculoskeletal structures and positioning of this study [178].

そこで本節では, 状態推定・制御・シミュレーションを統一的に扱うことができる Musculoskeletal AutoEncoder (MAE) を提案する [178]. これは一般化多感覚相関モデルを用いた静的身体図式の Parametric Bias を含まない一例である. 本ネットワークを実機センサデータをもとにオンラインで更新していくことで, 筋骨格ヒューマノイドの状態推定・制御・シミュレーションをより正確かつ継続的に行うことができるようになる. また, 本節では静的な関係のみを扱っているため, 動的な要素は考慮していない.

6.3.2 Musculoskeletal AutoEncoder

筋骨格センサ間の関係性

本節では, 筋骨格構造において基本的に測定可能な関節角度 θ , 筋張力 f , 筋長 l に焦点を当てる. なお, θ については Eq. 2.14 を使い筋長変化とマーカから関節角度を測定することができる. これまで, [128, 100] においては θ と l の非線形な関係 $\theta \rightarrow l$ を学習していた. さらにそれを発展させ, [124] では筋骨格構造に特有な身体組織の柔軟性を考慮し, $(\theta, f) \rightarrow l$ を学習していた.

実際には, Fig. 6.27 に示すように① $(\theta, f) \rightarrow l$ だけでなく, ② $(f, l) \rightarrow \theta$ や③ $(l, \theta) \rightarrow f$ という関係が存在する. つまり, (θ, f, l) の3つのセンサ情報のうち, 2つから残り1つを求めることができるということである. これは言い換えれば, (θ, f, l) のうち2つのセンサ情報から, 3つ全てのセンサ情報を取得できることに相当する.

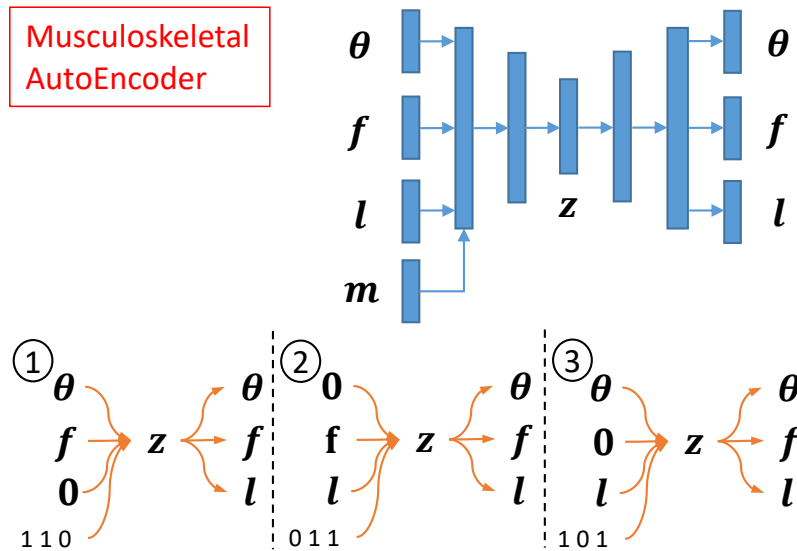


Fig. 6.27: The network structure of Musculoskeletal AutoEncoder [178].

ネットワーク構造とシステムの概要

Fig. 6.27 に Musculoskeletal AutoEncoder のネットワーク構造を示す. (θ, f, l) とマスク変数 m を入力とし, 同様の (θ, f, l) を出力するような, AutoEncoder 型 [227] の一般化多感覚相関モデルである. ここで, 関節数を N , 筋数を M として, (θ, f, l, m) の次元数はそれぞれ $(N, M, M, 3)$ となる. Musculoskeletal AutoEncoder において最もユニット数の少ない中間層の値を z とし, この次元は $N + M$ とする. これは, (θ, f) から (θ, f, l) の全ての値を推論できるため, 最小のユニット数は $N + M$ よりも小さくすることが可能であるためである. 本節では Musculoskeletal AutoEncoder は入力と出力層を含めた 7 層とし, そのユニット数は $(N + 2M + 3, 200, 30, N + M, 30, 200, N + 2M)$ としている. 本節では層の数はそこまで大きく最終的な結果には影響しなかったが, ユニット数に関しては, これ以上減らすと, 精度が落ちていった.

次に, Musculoskeletal AutoEncoder の初期学習・オンライン学習の概要について述べる. 先に述べたように, (θ, f, l) のうち 2 つのセンサ情報から 3 つのセンサ情報全てを取得できるという性質を利用する. つまり, Fig. 6.27 の①のように, 得られるセンサ情報が (θ, f) のみの場合は $(\theta, f, \mathbf{0}, (1, 1, 0))$ を入力として, (θ, f, l) を出力する. ②, ③も同様である. まず関節と筋の関係を含む幾何モデルから Musculoskeletal AutoEncoder を初期学習させる. その後, 実機センサデータを取得し, このネットワークをオンラインで更新していくことになる.

Musculoskeletal AutoEncoder における Encode 部分を $z = h_{enc}(\theta, f, l, m)$, Decode 部分を $(\theta, f, l) = h_{dec}(z)$ とする. また, ①における h_{enc} を $h_{enc,1}(\theta, f) = h_{enc}(\theta, f, \mathbf{0}, (1, 1, 0))$ とし, その他同様に②については $h_{enc,2}(f, l)$, ③については $h_{enc,3}(\theta, l)$ とする. Decode 部分に関しても, $\theta = h_{dec,\theta}(z)$, $f = h_{dec,f}(z)$, $l = h_{dec,l}(z)$ とする.

Fig. 6.28 に Musculoskeletal AutoEncoder を用いた全体システム図を示す. Musculoskeletal AutoEncoder と誤差逆伝播をうまく用いることで, 筋骨格ヒューマノイドの状態推定・制御・シミュレーションを行うことができる.

Musculoskeletal AutoEncoder の初期学習

幾何モデルを用いた Musculoskeletal AutoEncoder の初期学習について述べる. ここで言う幾何モデルとは, 筋の起始点・中継点・終止点を直線で結び筋経路を表現したものである. 関節角度を指定すると, 筋の経路点間の距離から筋長を求めることができる. ここで, 幾何モデルから筋の絶対長さを求める関数を $h_{geo,abs}(\theta)$, 初期姿勢 (全関節角度が 0, つまり $\theta = \mathbf{0}$) からの相対筋長を求める関数を $h_{geo}(\theta)$ とする.

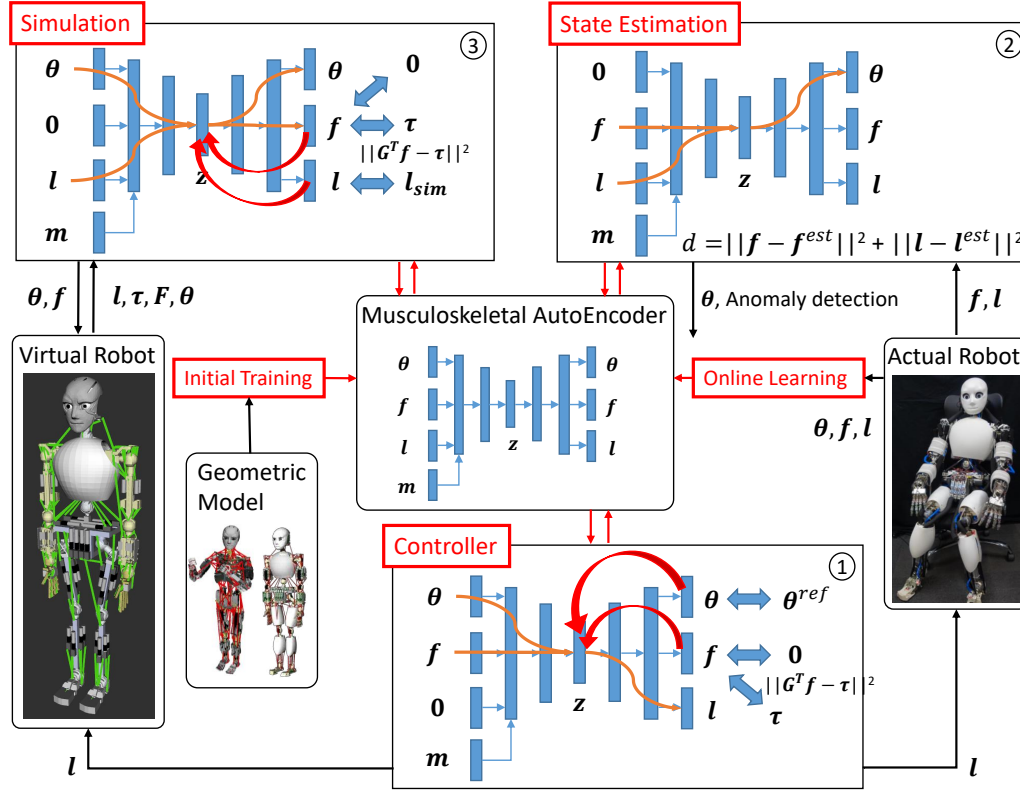


Fig. 6.28: State estimation, control, and simulation methods using Musculoskeletal AutoEncoder [178].

まず、筋単体のテストベッドにより、筋長の全体長さ l_{abs} に比例するダイニーマの伸びを考慮した、筋張力 f と非線形弾性要素を含む筋長変化の関係 $L_e(l_{abs}, f)$ を関数フィッティングにより求める。次に、以下のようなデータを生成する。

$$(\theta, f, l) = (\theta_r, f_r, h_{geo}(\theta_r) - L_e(h_{geo,abs}(\theta_r), f_r)) \quad (6.5)$$

ここで、 θ_r はランダムな関節角度、 f_r はランダムな筋張力である。このデータを用いて、Fig. 6.27 における①②③をランダムに実行する。つまり、 (θ, f, l) のうち一つをランダムに $\mathbf{0}$ にし、対応する m を加えたデータをネットワークの入力とし、 $(\theta^{pred}, f^{pred}, l^{pred})$ を出力する。最後に、損失関数 L_{init} を以下のように計算し、Musculoskeletal AutoEncoder を学習させる。

$$L_{init} = w_\theta \|\theta - \theta^{pred}\|_2 + w_f \|f - f^{pred}\|_2 + w_l \|l - l^{pred}\|_2 \quad (6.6)$$

本節では学習の際のバッチ数を $C_{batch}^{init} = 100$ 、エポック数を $C_{epoch}^{init} = 30$ 、 $w_\theta = 1.0$ 、 $w_f = 10.0$ 、 $w_l = 10.0$ とし、最適化手法は Adam [46] を用いる。

Musculoskeletal AutoEncoder のオンライン学習

実機センサデータを用いたオンライン学習について述べる。

まず、実機からセンサデータ $(\theta^{data}, f^{data}, l^{data})$ を取得する。ただし、筋骨格ヒューマノイドは一般的には関節角度を測定することができないため、Eq. 2.14 のように視覚を用いて実機関節角度を推定する。また、これは関節角度か筋張力が前回の学習の際のデータからある閾値以上離れ、かつ静止している場合に学習データを作成する。

次に、このデータを蓄積・拡張し、Musculoskeletal AutoEncoder をオンラインで更新していく。データ $(\theta^{data}, f^{data}, l^{data})$ を蓄積していき、データ数が C_{thre} を超えたところで、更新を開始する。蓄積したデータの中から C_{data} ($C_{data} \leq C_{thre}$) 個、そして最新のデータ 1 個を取得する。また、 $(0, 0, 0)$ というデータも一個加える。これらデータに対して、それぞれ①②③の 3 種類の m の値を加え、計 $3 \times (C_{data} + 2)$ 個のデータを用いてネットワークを更新する。

本節では、更新の際のバッチ数を $C_{batch}^{online} = 10$ 、エポック数を $C_{epoch}^{online} = 10$ とし、その他定数は $C_{thre} = 20$, $C_{data} = 10$ とした。

Musculoskeletal AutoEncoder による状態推定

まず状態推定の手法について述べる。筋骨格ヒューマノイドは一般的には関節角度を測定することができないため、視覚を用いて実機関節角度を推定する。しかしこの場合、身体にマーカ等をつける必要や、身体を常に見続けなければならない等の制約を受けることになる。そこで、現在の筋張力・筋長情報から、関節角度を推定することで、自身の状態を常に知り続けることが可能となる。

Musculoskeletal AutoEncoder を用いた関節角度推定は [122] 等に比べて非常に単純である。Fig. 6.28 の右上図に示すように、実機から現在筋張力 f 、現在筋長 l を取得し、これを Fig. 6.27 の②の形でネットワークに入力し、関節角度推定値 $\theta^{est} = h_{dec, \theta}(h_{enc, 2}(f, l))$ を得るのみである。

また、これまでの手法では関節角度推定のみしか出来なかったが、Musculoskeletal AutoEncoder を用いることで異常検知をすることができる。先ほどの関節角度推定の際、 θ^{est} だけでなく、 $(\theta^{est}, f^{est}, l^{est}) = h_{dec}(h_{enc, 2}(f, l))$ のように、全ての値を取り出す。ここで、Musculoskeletal AutoEncoder の②では f, l は入力の値と同じ値を復元するように学習されている。しかし、例えば筋が切れた等の異常が発生した場合は、その状態を学習していないため復元が難しくなる。よって、以下の値が上昇する。

$$d = \|f^{est} - f\|_2 + \|l^{est} - l\|_2 \quad (6.7)$$

この値 d を監視することによって、オンライン学習を停止したり、異常を検知して動作を停止したりすることが可能となる。

Musculoskeletal AutoEncoder による制御

ここでは主に2つの筋長指令による位置制御 ([124] を MAE を用いて再現したもの、MAE を使った新しい手法) について述べ、最後に筋張力制御・可変剛性制御について概要を述べる。

まず、[124] と同様の位置制御を Musculoskeletal AutoEncoder を用いて行う方法について説明する。前提として、本節では筋骨格ヒューマノイドを筋剛性制御 [121] により動作させる。ある関節角度 θ^{ref} を実現したい場合、Eq. 2.4 における l^{ref} を決める必要があり、それは以下のように行う。

$$l_{comp}(f) = -(f - f_{bias})/k_{stiff} \quad (6.8)$$

$$l^{ref} = h_{dec,l}(h_{enc,1}(\theta^{ref}, f_{const})) + l_{comp}(f_{const}) \quad (6.9)$$

$$l^{ref} = h_{dec,l}(h_{enc,1}(\theta^{ref}, f)) + l_{comp}(f) \quad (6.10)$$

ここで、 f_{const} はある一定の筋張力 (基本的には f_{bias} を用いる)、 l_{comp} は筋剛性制御によるソフトウェアでの l^{ref} からの筋の伸びを補償するための項である。[124] にあるように、Eq. 6.9, Eq. 6.10 の順に実行することで、 θ^{ref} を実現する。

次に、Musculoskeletal AutoEncoder を用いたより正確な制御手法について説明する。まず、現在の筋張力 f 、指令関節角度 θ^{ref} から、潜在状態 $z = h_{enc,1}(\theta^{ref}, f)$ を求める。次に以下の工程を繰り返す。

- (1) $(\theta^{pred}, f^{pred}, l^{pred}) = h_{dec}(z)$ を求める。
- (2) 損失 $L = h_{control}(\theta^{pred}, f^{pred}, l^{pred})$ を計算する。
- (3) 誤差逆伝播 [140] により z を更新する。

(2) では、以下のように損失関数 $h_{control}$ を用いて損失 L を計算する。

$$L = h_{control}(\theta, f, l) = w_1 \|f\|_2 + w_2 \|\theta - \theta^{ref}\|_2 + w_3 \|\tau^{ref} + G^T(\theta^{ref}, f)f\|_2 \quad (6.11)$$

ここで、 τ^{ref} は幾何モデルから計算された θ^{ref} を保つのに必要な関節トルク値である。また、 $G(\theta, f)$ は筋長ヤコビアンであり、微小な変位 $\Delta\theta$ を加えた $(\theta + \Delta\theta, f)$ と (θ, f) を①の形で Musculoskeletal AutoEncoder に入力したときの出力 l の差から計算することができる。(3) では、以下のように誤差逆

伝播法 [140] を元に \mathbf{z} を最急降下法で更新する.

$$\mathbf{g} = \partial L / \partial \mathbf{z} \quad (6.12)$$

$$\mathbf{z} = \mathbf{z} - \gamma \mathbf{g} / \|\mathbf{g}\| \quad (6.13)$$

ここで, \mathbf{g} は \mathbf{z} に関する L の勾配である. このとき, γ の値を決め打ちしても良いが, 本手法では様々な γ によってバッチを作成し, 最も良い γ を選ぶ. γ の最大値 $\gamma_{\max}^{\text{control}}$ を決め, 0 から $\gamma_{\max}^{\text{control}}$ までの値を $N_{\text{batch}}^{\text{control}}$ 等分し, それらによって更新された \mathbf{z} を $N_{\text{batch}}^{\text{control}}$ 個作成する. もう一度 (1) と (2) を行い, 最も L が小さかった \mathbf{z} を採用する. これら (1)–(3) の工程を, $C_{\text{epoch}}^{\text{control}}$ 回行う. そして最後に, 得られた $\mathbf{l}^{\text{pred}}, \mathbf{f}^{\text{pred}}$ から以下のように \mathbf{l}^{ref} を計算し, 実機に指令する.

$$\mathbf{l}^{\text{ref}} = \mathbf{l}^{\text{pred}} + \mathbf{l}_{\text{comp}}(\mathbf{f}^{\text{pred}}) \quad (6.14)$$

これにより, 関節角度 $\boldsymbol{\theta}^{\text{ref}}$ を実現する中で, 筋張力が最小となるような \mathbf{l}^{ref} を求めることが可能となる. $\|\mathbf{f}\|_2$ を最小化し $\boldsymbol{\tau}^{\text{ref}} = -G^T \mathbf{f}$ を満たすように二次計画法を解いて直接必要な筋張力 \mathbf{f}^{ref} を求めることも可能だが, $\boldsymbol{\theta}^{\text{ref}}$ において \mathbf{f}^{ref} を実現できる保証はなく, 本手法では潜在空間 \mathbf{z} 内において探索している.

最後に, 筋張力制御・可変剛性制御について説明する. 筋張力制御は筋長やコピアンのみ必要なため, [226, 188, 123] と同様に行うことができる. また, 可変剛性制御も同様に, [125] と同じように行うことが可能である.

本節では, $w_1 = 1.0, w_2 = 1.0, w_3 = 0.01, \gamma_{\max}^{\text{control}} = 0.5, C_{\text{batch}}^{\text{control}} = 10, C_{\text{epoch}}^{\text{control}} = 10$ とする.

Musculoskeletal AutoEncoder によるシミュレーション

Musculoskeletal AutoEncoder を用いた筋骨格ヒューマノイドのシミュレーションを行う. ここでは, 筋長を制御指令として関節角度と筋張力の遷移をシミュレーションしている.

まず, 現在のシミュレーションにおける筋張力 \mathbf{f}_{sim} , 筋長 \mathbf{l}_{sim} から, 潜在状態 $\mathbf{z} = \mathbf{h}_{\text{enc},2}(\mathbf{f}_{\text{sim}}, \mathbf{l}_{\text{sim}})$ を求める. 次に, 前述の制御手法と同様に, (1)–(3) の工程を繰り返す. この工程のうち前述の制御手法と異なるのは, (2) の損失の計算のみである. 以降では, 前節の $\mathbf{h}_{\text{control}}$ を $\mathbf{h}_{\text{sim},1}$ または $\mathbf{h}_{\text{sim},2}$ で置き換える. (2) では, 以下のように損失 L を損失関数 $\mathbf{h}_{\text{sim},1}$ により計算する.

$$L = \mathbf{h}_{\text{sim},1}(\boldsymbol{\theta}, \mathbf{f}, \mathbf{l}) = w_4 \|\mathbf{f}\|_2 + w_5 \|\mathbf{l} - \mathbf{l}_{\text{sim}}\|_2 + w_6 \|\boldsymbol{\tau}^{\text{ref}} + G^T(\boldsymbol{\theta}_{\text{sim}}, \mathbf{f}_{\text{sim}}) \mathbf{f}\|_2 \quad (6.15)$$

これにより, ロボットに送った筋長を満たしつつ, 重力補償等に必要なたルク $\boldsymbol{\tau}^{\text{ref}}$ を発揮する潜在空間 \mathbf{z} を得ることができる. 外力を加えた際の動きをシミュレーションしたい場合も, 関節の幾何モデ

ルから必要なトルクを計算することができ、同様にシミュレーションを行うことができる。また、損失 L を変えることによって、外力だけでなく、無理やり身体を外部からある姿勢 θ_{fix} に動かした際やテーブル等に手を置いた時等の動きもシミュレーションすることが可能である。この場合の損失関数 $h_{sim,2}$ は以下ようになる。

$$h_{sim,2}(\theta, f, l) = w_4 \|f\|_2 + w_5 \|l - l_{sim}\|_2 + w_7 \|\theta - \theta_{fix}\|_2 \quad (6.16)$$

注意点として、損失関数をこの $h_{sim,2}$ にする場合は、最初の潜在空間 z を求める際に $z = h_{enc,3}(\theta_{sim}, l_{sim})$ とした方がシミュレータが安定して動作した。最後に、ここで最終的に求まった z から、 $(\theta'_{sim}, f'_{sim}, l'_{sim}) = h_{dec}(z)$ を求め、 $\theta_{sim} = \theta'_{sim}$, $f_{sim} = f'_{sim}$ のようにシミュレータを更新する。これら全ての工程をリアルタイムで回すことで、シミュレーションが行われる。

本節では、 $w_4 = 0.1$, $w_5 = 1.0$, $w_6 = 0.001$, $w_7 = 1.0$, $\gamma_{max}^{sim} = 0.2$, $C_{batch}^{sim} = 10$, $C_{epoch}^{sim} = 3$ とする。ここで、 γ_{max}^{sim} , C_{batch}^{sim} , C_{epoch}^{sim} はそれぞれ、シミュレーション内の計算における $\gamma_{max}^{control}$, $C_{batch}^{control}$, $C_{epoch}^{control}$ と同等の定数を指す。

6.3.3 実験

本節の実験では筋骨格ヒューマノイド Musashi を用いる。左腕の筋配置は Fig. 4.24 である。本節では主に、肩の3自由度、肘の2自由度を用いて実験を行うこととし、関節角度は $\theta = (\theta_{S-r}, \theta_{S-p}, \theta_{S-y}, \theta_{E-p}, \theta_{E-y})$ のように表す (S は shoulder, E は elbow, rpy はそれぞれ roll, pitch, yaw を表す)。ゆえに、 $N = 5$, $M = 10$ である。

オンライン学習実験

Musculoskeletal AutoEncoder のオンライン学習について述べる。本節では [125] と同様に、3段階でロボットを動かすことを繰り返し、学習を行う。まず、Eq. 6.9 において関節角度限界の中でランダムに θ^{ref} を指定し、 f_{const} は最初に一律で f_{bias} を指定する。次に指令関節角度をそのまま Eq. 6.10 を実行し、最後に、筋張力指令を f_{bias} から f_{limit} の間でランダムに指定してもう一度 Eq. 6.9 を実行する。この3つの手順を繰り返すことで、効率よく関節角度と筋張力の空間を学習させる。ここで、 $f_{bias} = 30$ [N], $f_{limit} = 200$ [N] とする。関節角度推定値 θ^{est} と関節モジュールのセンサ値 θ の差分 $\|\theta^{est} - \theta\|_2$ の遷移を Fig. 6.29 に示す。オンライン学習を実行するにつれ、 $\|\theta^{est} - \theta\|_2$ が徐々に下がっていくことがわかる。10分間の実験を行った結果を指数関数近似した曲線から、関節角度推定誤差は10分間で 0.324 [rad] から 0.154 [rad] と、約半分程度まで下がっていることがわかる。

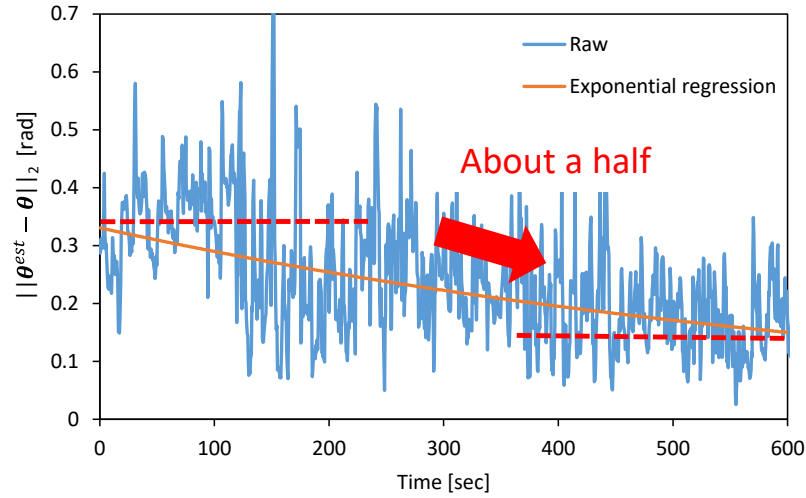


Fig. 6.29: The transition of the error between the current and estimated joint angles while executing online learning of Musculoskeletal AutoEncoder [178].

オンライン学習前と学習後における Musculoskeletal AutoEncoder の変化を Fig. 6.30, Fig. 6.31 に示す. Fig. 6.30 は筋張力が 0 の状態で, θ_{s-p} を $[-120, 30]$ [deg] の範囲で変化させたときの筋長変化を表している. また, Fig. 6.31 は初期姿勢において筋張力を $[0, 500]$ [N] の範囲で変化させたときの筋長変化を表している. 両者とも, 学習前と学習後でそれぞれ大きく変化している. Fig. 6.30 では, 関節変化に対する筋変化は原点を維持しながら最大で 40 [mm] 程度変化している. 特に, θ_{s-p} を動作させる主要な筋 #2, #3, #7 に大きな変化が起きていることもわかる. また, Fig. 6.31 では, より大きく非線形性が出るもの (#2) やほぼ直線になるもの (#6) まで様々である. #2 は非常に強い力がかかりやすい筋であり, 非線形弾性要素が劣化で変化したため非線形性がより大きく出ていると考えられる. また, #6 は肘に関わる筋で取り回しの都合上非常に摩擦が強く, 非線形性が消えてしまっていると考えられる.

最後に, $||\theta^{est} - \theta||_2$ に誤差が残ってしまうのは, 筋や関節の摩擦によるヒステリシスの影響が大きいと考えられる.

状態推定実験

状態推定について実機実験を行い評価する. オンライン学習を行った後にそれを止め, 学習する前とした後における関節角度推定値の追従の差を評価した. 実験の途中で, 重量物体の把持, また, 筋一本の機能を停止することを行い, その際の関節角度推定値の追従, また, 異常度 d の測定を行う. 本実験中は一切のオンライン学習は行っていない. 実験の様子を Fig. 6.32 に, 実験結果を Fig. 6.33 に示す.

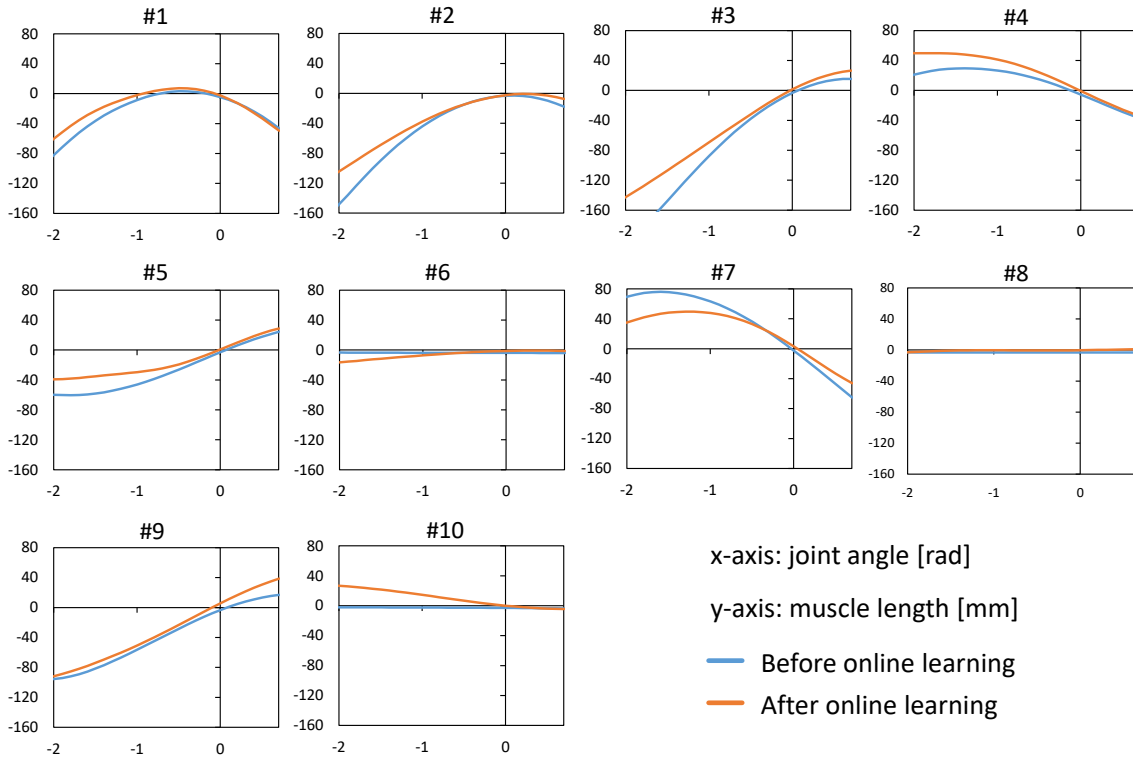


Fig. 6.30: The transition of muscle lengths by changing joint angles in Musculoskeletal AutoEncoder, before and after the online learning [178].

ランダムに関節を動かした際には、学習前と後で、関節角度推定誤差が、平均で 0.414 [rad] から 0.186 [rad] と、大きく下がっていることがわかる。また、オンライン学習の際に筋張力の空間も効率的に学習しているため、重量物体を把持した際も大きく関節角度推定誤差が上がることはなかった。最後に、異常検知度 d の推移を観察する。学習後は、一本の筋 (Fig. 4.24 の #2) の機能を停止することで、 d が急上昇しており、異常を検知可能になったことがわかった。一方で、学習前では d はあまり上昇しない。これは、初期学習の際には筋張力を全空間でランダムに決めているため、実行不可能な筋張力においても同じようにセンサ値を復元可能であり、 d が下がるためである。

制御実験

筋長による関節位置制御について実機実験を行い評価する。まず、評価を行う関節角度 θ_{eval} を5点ランダムに決定する。あるランダムな関節角度 θ_{random} から θ_{eval} に動作することを θ_{random} を変えながら5回行い、そのときの $\|\theta_{eval} - \theta\|_2$, $\|f\|_2$ の平均と分散を評価する。これは、ヒステリシス等の影響を考慮してこのように行った。幾何モデルを直接使った際 (geometric), 学習後に Eq. 6.9, Eq. 6.10

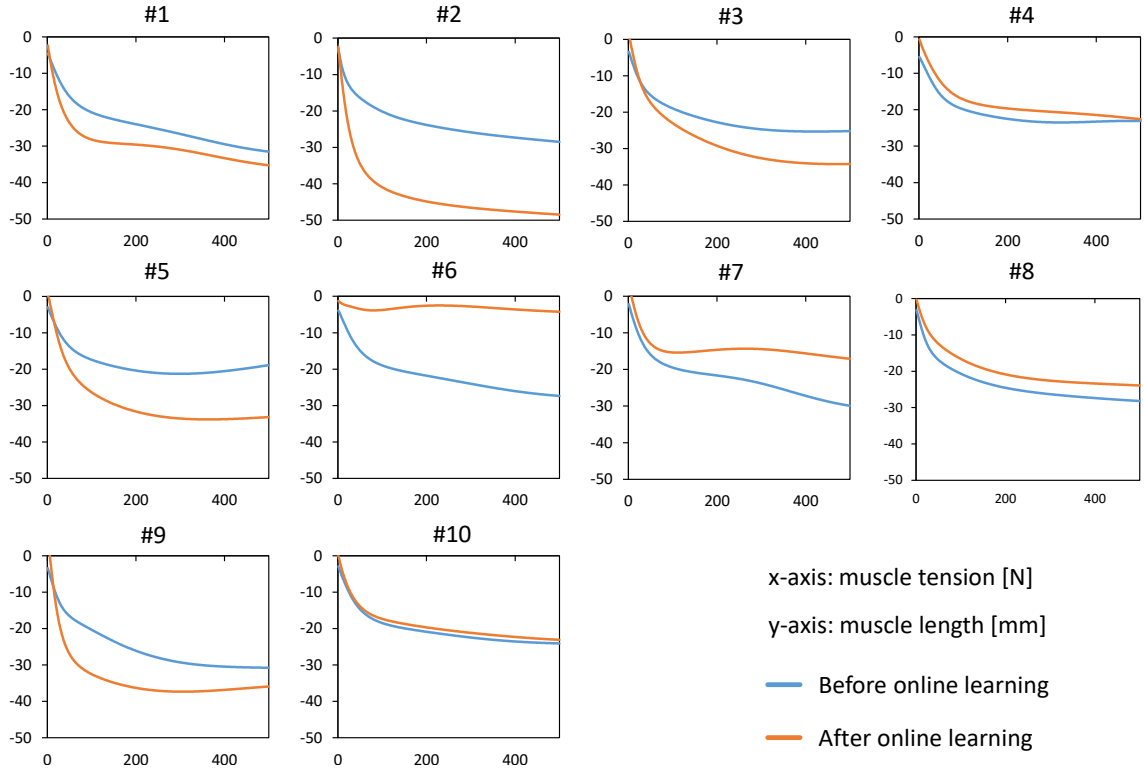


Fig. 6.31: The transition of muscle lengths by changing muscle tensions in Musculoskeletal AutoEncoder, before and after the online learning [178].

の順に動かした際の結果 (**first, second**), また, 学習後に Eq. 6.14 を用いて動かした際 (**keep**) の結果を Fig. 6.34 に示す.

幾何モデルよりも, 学習後の制御の方が圧倒的に関節角度の実現精度が高いことがわかる. また, Eq. 6.10 は Eq. 6.9 の後の現在筋張力を使うため, より精度が高い. Eq. 6.14 は精度に関しては Eq. 6.10 とほぼ同じような挙動をしている. また, 筋張力に関しては幾何モデルを使ったものが他の手法に比べ圧倒的に小さい. これは, 学習されていないため幾何モデルと実機の誤差が大きいため, 指令関節角度を実現できず緩んでしまっているからであると考えられる. また, Eq. 6.10 は筋張力が高まりやすく, Eq. 6.9 に比べて高い筋張力を発揮してしまっていることがわかる. それに対して, Eq. 6.14 は全姿勢において Eq. 6.9 や Eq. 6.10 よりも $\|f\|_2$ が小さい.

Eq. 6.9, Eq. 6.10 は順に実行しなければならないため姿勢の実現に時間がかかり, かつ内力の高まりが大きい. それに対して, Eq. 6.14 は内力の高まりを抑えつつ, Eq. 6.10 と同等の追従性を得ることが出来た.

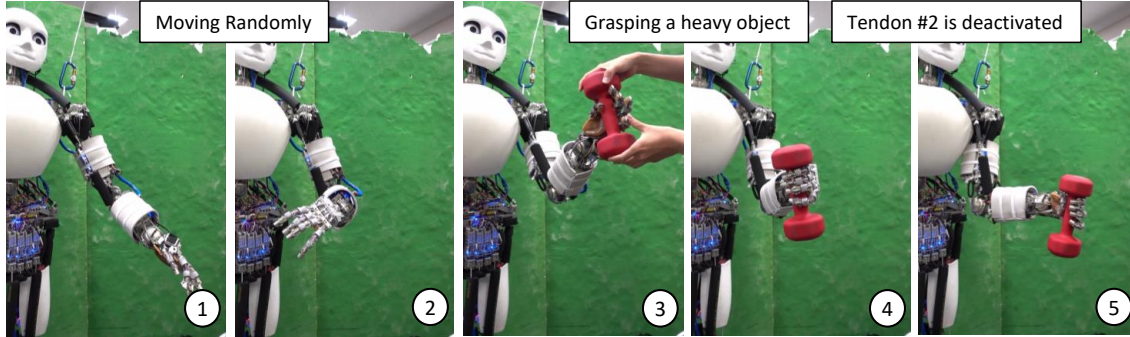


Fig. 6.32: The experiment of state estimation using Musculoskeletal AutoEncoder [178].

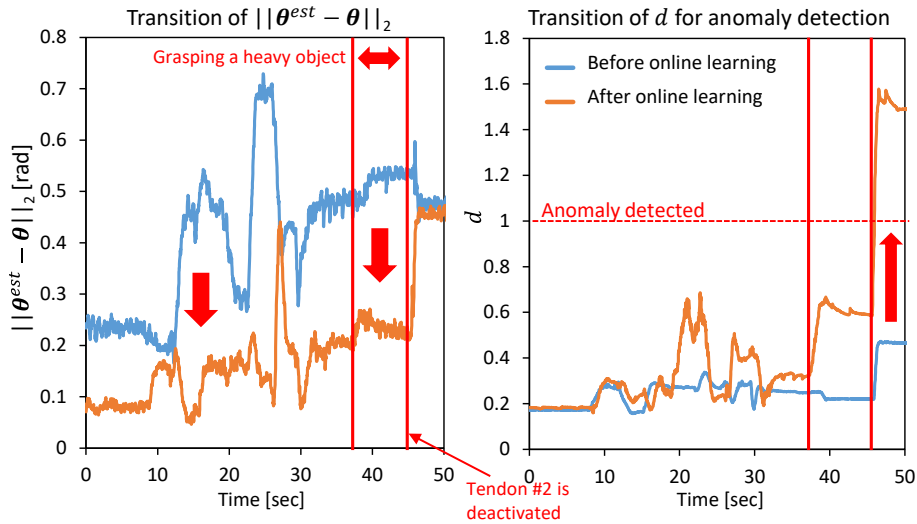


Fig. 6.33: The result of state estimation using Musculoskeletal AutoEncoder [178].

シミュレーション実験

シミュレーションに関する実験を行う。オンライン学習前と学習後において、実機の動きとシミュレータ上のロボットの動きの差異を評価する。学習前のシミュレータ・学習後のシミュレータ・実機動作の際におけるセンサ値の比較を Fig. 6.35 に示す。筋張力の値が高いほど筋の色が赤く、小さいほど色が緑色になるようにしている。学習前に比べ、学習後の方が実機センサ値に筋張力・関節角度が近づいていることがわかる。学習前と学習後における $\|\theta_{sim} - \theta\|_2$ と $\|f_{sim} - f\|_2$ の遷移を Fig. 6.36 に示す。学習前に比べ学習後の方が、関節角度については平均で 0.335 [rad] から 0.162 [rad], 筋張力については平均で 104.8 [N] から 92.2 [N] と、実機センサ値とシミュレータ値の誤差が小さくなっていることがわかる。これにより、シミュレータの挙動を学習により実機に近づけていくことが可能であ

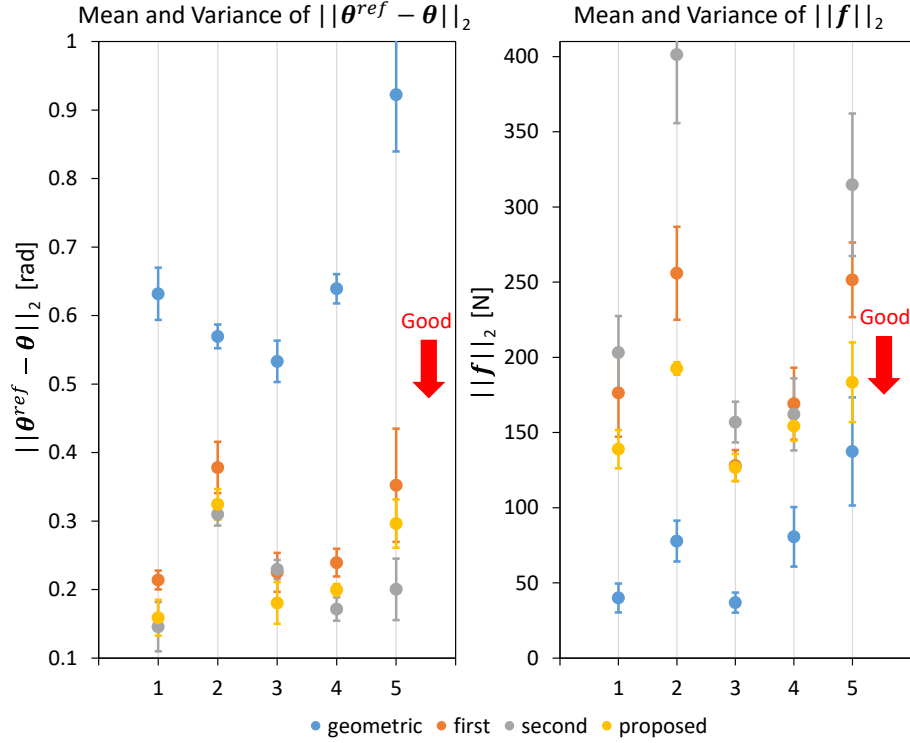


Fig. 6.34: The average and variance of $\|\theta_{eval} - \theta\|_2$ and $\|f\|_2$ among four controls: **geometric**, **first**, **second**, and **proposed** [178].

ることがわかった。

また、損失関数を変えることでシミュレータの挙動を変化させることができる様子を Fig. 6.37 に示す。これは、肘を 90 度に曲げ、 z 方向に -50 N、 y 方向に 50 N を順にかけ、その後 θ_{s-p} を無理やり 30 deg にした際の挙動である。力を指定する際は Eq. 6.15 を用い、姿勢を指定する際は Eq. 6.16 を使用している。かける力を変えたり、 θ_{fix} を与えたりすることで、それに応じた関節角度・筋張力を確認することが可能である。

6.3.4 Musculoskeletal AutoEncoder の特殊系に関する考察

先行研究である身体図式学習 [100, 124, 125] と、この Musculoskeletal AutoEncoder の関係性について述べる。これらは Musculoskeletal AutoEncoder の特殊系であり、それは Fig. 6.38 のような関係を持っている。(a) は関節角度-筋長マッピング (JLM, joint-length mapping) [100] であり、 $\theta \rightarrow l$ の関係性のみを表現している。(b) は関節角度-筋張力-筋長マッピング (JTLM, joint-tension-length mapping) [124, 125] であり、 $(\theta, f) \rightarrow l$ の関係性を表現している。

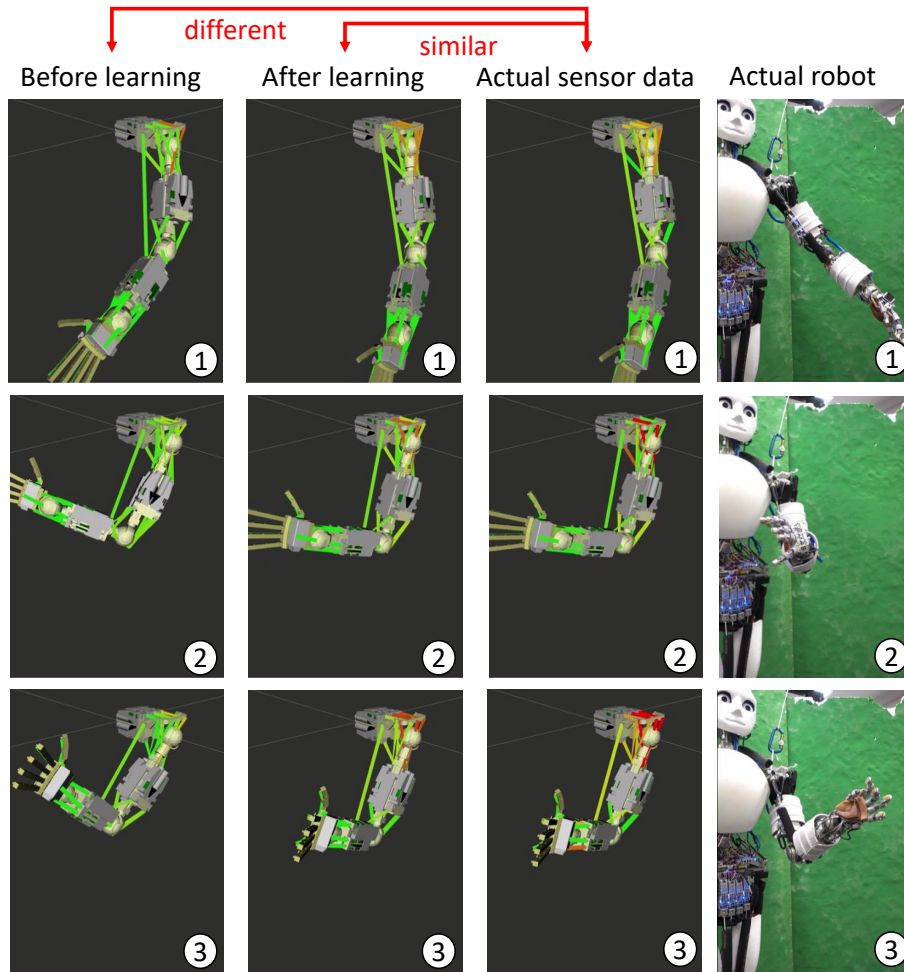


Fig. 6.35: The differences among the actual robot, simulation before online learning, and simulation after online learning [178].

MAE, JLM, JTLM にはできること, できないことに差がある. JLM は筋張力項がないため力に関する情報を使うことはできないが, JTLM, MAE は力の情報, 筋の伸び等を考慮することができる. また, JTLM と MAE は筋張力と筋の伸びの関係を利用することで, 可変剛性制御が可能である. 筋長制御については, JLM, JTLM は指定した関節角度や筋張力に対応する筋長を計算することはできるものの, 筋張力指令に対する制約等を考慮することはできない. 一方, MAE は様々な条件下で制御入力を計算することが可能である. 関節角度推定については, JLM と JTLM は拡張カルマンフィルタと併用する必要があるものの [124], MAE はネットワークから直接関節角度推定が可能である. また, シミュレーションや異常検知は MAE においてのみ可能である.

関節角度-筋長マッピングと関節角度-筋張力-筋長マッピングの詳細は以下で述べる.

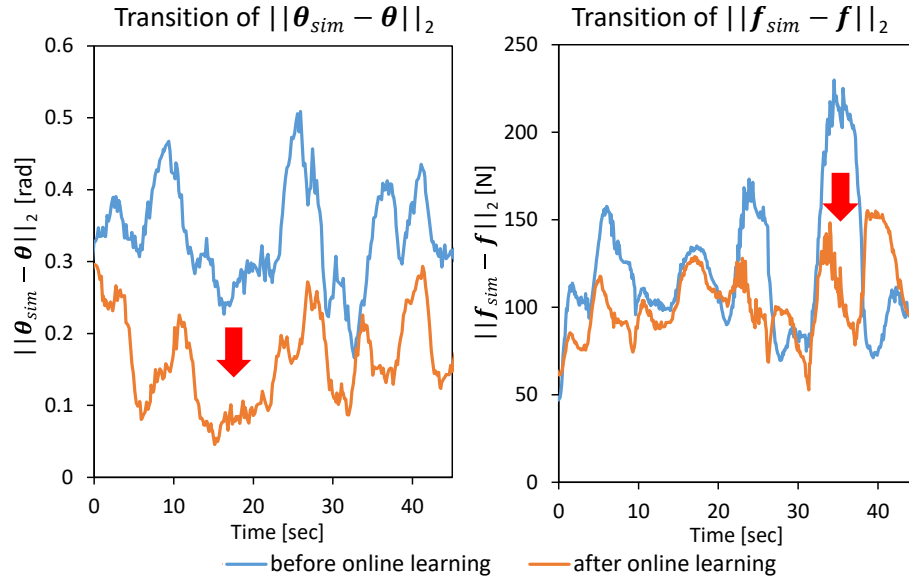


Fig. 6.36: The transition of $\|\theta_{sim} - \theta\|_2$ and $\|f_{sim} - f\|_2$ before and after online learning [178].

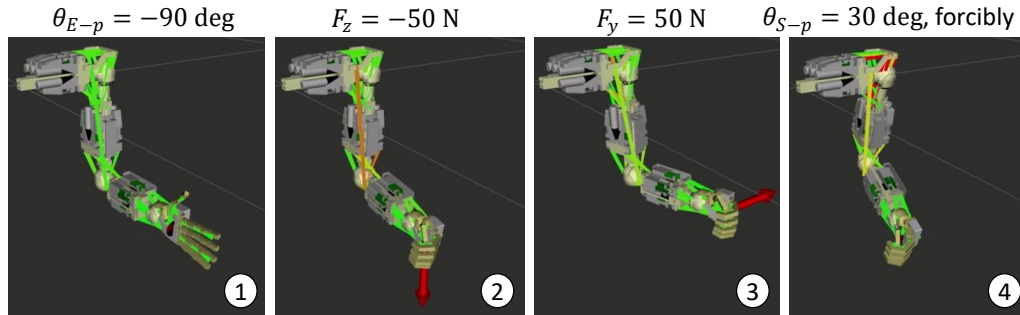


Fig. 6.37: The result of simulation experiment when changing the loss function [178].

関節角度-筋長マッピング

ここでは、関節角度-筋長マッピング $l = h(\theta)$ の学習方法と、これにより可能となる正確な制御・内力の減少について述べる。ここで言う関節角度-筋長マッピング [100] は厳密には $\theta \rightarrow l$ ではなく $\theta \rightarrow l^{ref}$ を表している。この関節角度-筋長マッピングを学習させるためには2つの更新則: Antagonism Updater と Vision Updater が必要である。Antagonism Updater は (θ, l) のデータを学習させることで、筋の緩みや筋の無駄な内力を減らしていく更新則である。一方、Vision Updater は (θ, l^{ref}) のデータを学習させることで、より正確に関節角度を実現できるようにする更新則である。Vision Updater だけでは拮抗関係は一切変化しないため、Antagonism Updater を使うことで筋の緩みや内力を防止している。

Antagonism Updater によるオンライン学習の有効性を、肘関節を用いた実験で示す。肘関節には主

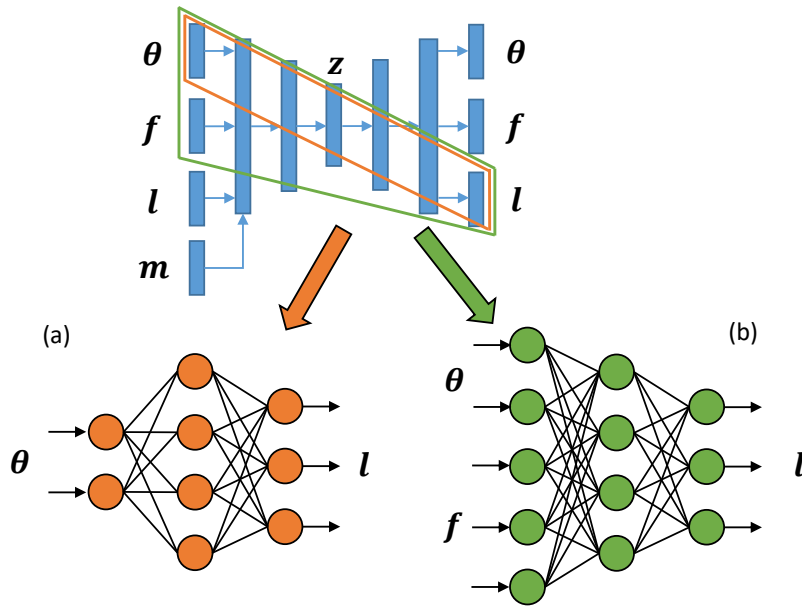


Fig. 6.38: The specific versions of musculoskeletal autoencoder.

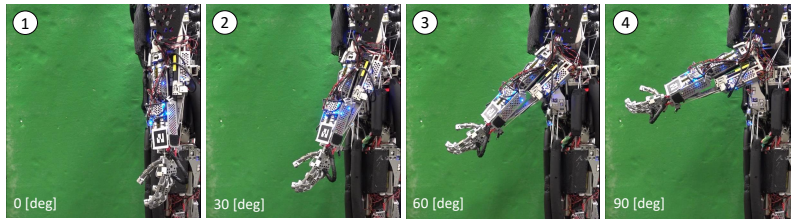


Fig. 6.39: The movement of Antagonism Updater experiment using the elbow joint. We repeated this movement 11 times [100].

に3つの筋が存在し、それは上腕筋、上腕二頭筋、上腕三頭筋である。動作としては、肘を0度、30度、60度、90度の順に2秒で動かして2秒止めるを繰り返すような動作を行う (Fig. 6.39)。肘関節を何度も動かすことによって、主動筋二本の筋張力が均等になりかつ拮抗筋の筋張力が減っていく様子を Fig. 6.40 に示す。11回の試行で最大張力は370 N から250 N まで下がっている。最初は発揮張力に大きな差があった主動筋である上腕筋と上腕二頭筋の張力であるが、試行ごとに段々と近くなっていることがわかる。一方、Fig. 6.39 に見るように、実際に動作させたい関節角度と実機の動作が大きく異なっていることがわかる。Antagonism Updater は拮抗関係の修正のみを行い、実機が指令した通りの動作を行うようになるためには、Vision Updater が必要である。

Vision Updater による視覚と指令筋長を用いたオンライン学習の有効性を示すため、腕を動作させ、その際の体を見ることで関節角度推定が正しくなっていく様子を示す。人間が Kengoro の首や肩に任

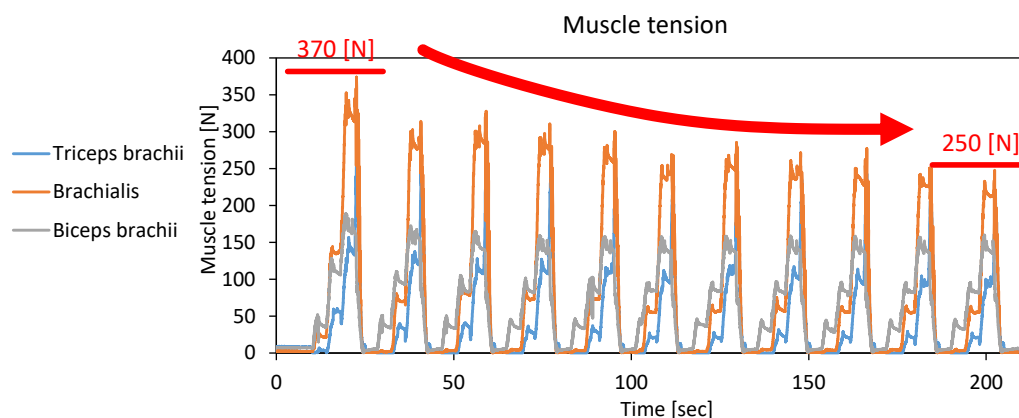


Fig. 6.40: The result of Antagonism Updater experiment using the elbow joint. This graph shows muscle tension change during the elbow flexion [100].

意の姿勢を連続的に送り、その際に頭に搭載したカメラによって手先についたマーカを認識し、それを用いてオンライン学習を行う。これにより、関節角度推定が正しくなる様子を Fig. 6.41 に示す。左図は手についたマーカの関節角度-筋長マッピングからの推定位置と、目から見たマーカの位置が正しくなっていく様子を示している。最初は二つのマーカがずれているが、最終的には重なるほど近い位置を示すようになっていく。また、右図は関節角度推定の値と目で見えたマーカから IK を解いた際の関節角度の値の差の RMSE の推移を表しており、その値が小さくなっていく様子を示している。肩関節を例にとると、最初は RMSE が約 16 deg だったのに対して、最終的に約 3 deg 程度まで下がっている。

この Antagonism Updater と Vision Updater を統合し、缶を見てそれを握らせるような物体把持実験を行う。その際の実験の様子を Fig. 6.42 に示す。また、実験中の RMSE と筋張力については Fig. 6.43 に示す。まず、AR マーカのついた缶を見て、それに対して逆運動学を解き、缶を掴む動作を行う。しかし、最初は幾何モデルと実機との誤差により正しい位置まで腕は動作せず、掴むことはできない。次に、様々な姿勢に体を動かしてみる。この間に視覚を用いた逐次的な学習によってある程度正しい関節角度-筋長マッピングが得られている。そして、もう一度缶に対してアプローチを行う。缶に対するアプローチは良くなったものの、まだ届いていないため、最後にもう一度周りを見渡してオンライン学習を行う。そして最終的に缶を掴み取ることができている。この際、缶に対して逆運動学を解いている位置は常に同じであるが、関節-筋空間マッピングが正しく修正されていっているため、手が動作する場所は変わっていき、それがわかる。また、Fig. 6.43 に示すように、RMSE が徐々に減り、必要な筋張力が減っていることがわかる。よって、二種類のオンライン学習が競合しないこと、関節角度-筋長マッピングを利用したシステムにより正確で内力の少ないマニピュレーションが可能となること

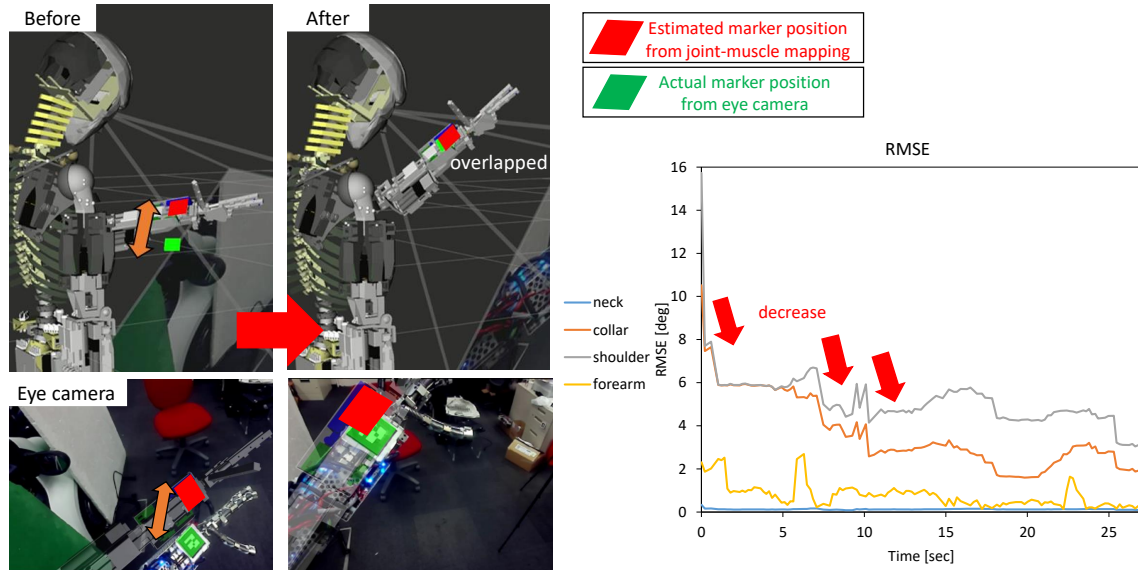


Fig. 6.41: The result of Vision Updater experiment. Left figure shows how this experiment was carried out; the right graph shows RMSE of the difference between the estimated joint angles from JLM and the actual joint angles calculated using the RGB camera [100].

が示された。

関節角度-筋張力-筋長マッピング

ここでは、関節角度-筋張力-筋長マッピング $l = h(\theta, f)$ により可能となる身体の筋経路変化補償と可変剛性制御について述べる。厳密には [124] は $(\theta, f) \rightarrow l^{ref}$, [125] は $(\theta, f) \rightarrow l$ を表すものの、その2つに大きな差はないため、区別なく述べる。

まず、身体の筋経路変化補償について述べる。重量物体(本実験では約 1.4 kg のダンベル)を指令関節角度で保持し続ける実験を行った。その際の実験の様子、肩ピッチと肘ピッチの θ^{ref} と実機の関節角度 θ の比較グラフを図 Fig. 6.44 に示す。はじめに Kengoro の肩ピッチを -30deg, 肘ピッチを -60deg とし、その状態でダンベルを握らせる。すると身体組織の柔軟性による筋経路変化によって腕は下方へ下がるが、この際に、Eq. 2.21 における指令関節角度と現在筋張力による筋長指令を繰り返す、筋張力変化に対するフィードバックループを組むことで、最終的に元の位置まで姿勢が修正されていることがわかる。つまり、関節角度-筋張力-筋長マッピングは筋張力が影響する身体の筋経路変化を考慮することが可能である。

次に、可変剛性制御について述べる。手先の作業空間剛性 K_w を導出するためには、関節ヤコビアン

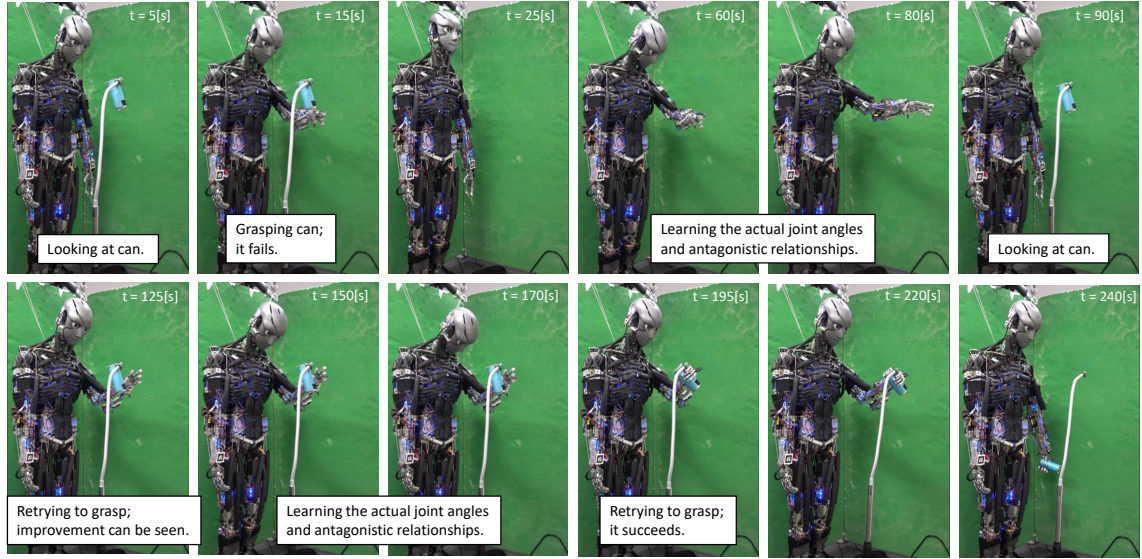


Fig. 6.42: The result of can grasping experiment. This figure shows how the experiment was carried out [100].

J , 筋長ヤコビアン G , 筋剛性 K_m の 3 つが必要となる. 関節ヤコビアン J は通常の軸関節型ロボット同様, 幾何モデルから直接得ることが可能である. 筋長ヤコビアン G はマッピング h を関節角度 θ で微分することで求めることができる. 筋剛性は現在の関節角度 θ における筋張力変化 Δf と h の出力から得られる筋長変化 Δl の間の線形回帰によって求めることができる ($\Delta f = K_m \Delta l$). よって, 関節剛性 K_j は $K_j = G^T(\theta) K_m G(\theta)$ から, 作業空間剛性 K_w は $K_w = J^T(\theta) K_j J(\theta)$ から計算することができる.

この作業空間剛性 K_w の計算を用いて可変剛性制御を行う. これは, 指令関節角度 θ^{ref} と指令作業空間剛性 K_w^{ref} を入力として, 指令筋張力 f^{ref} を計算し, 指令筋長 $l^{ref} = h(\theta^{ref}, f^{ref})$ を計算, 実機に送る制御である. まず, 現在の筋張力 f に, ある範囲内の (本節では -20 [N] から 20 [N]) のランダムな f_{rand} を加え f_{tmp} とし, それを以下の式に基づいて評価値 $E(\theta^{ref}, f_{tmp})$ を求めたデータを収集する.

$$\begin{aligned}
 f_{tmp} &= f + f_{rand} \\
 K_w(\theta^{ref}, f_{tmp}) &= J(\theta^{ref})^{-T} G(\theta^{ref})^T K_m(\theta^{ref}, f_{tmp}) G(\theta^{ref}) J(\theta^{ref})^{-1} \\
 \tau(\theta^{ref}, f_{tmp}) &= -G(\theta^{ref})^T f_{tmp} \\
 E(\theta^{ref}, f_{tmp}) &= \|(K_w^{ref})^{-1} K_w(\theta^{ref}, f_{tmp}) - I\|_2 + \alpha \|\tau(\theta^{ref}, f) - \tau(\theta^{ref}, f_{tmp})\|
 \end{aligned} \tag{6.17}$$

ここで, $K_w(\theta^{ref}, f_{tmp})$ は作業空間剛性, K_w^{ref} は指令作業空間剛性, $J(\theta)$ は関節ヤコビアン, $G(\theta)$ は筋長ヤコビアン, K_m は筋剛性, $\tau(\theta, f)$ は関節トルク, α は重みの定数を表す. 評価値 $E(\theta^{ref}, f_{tmp})$

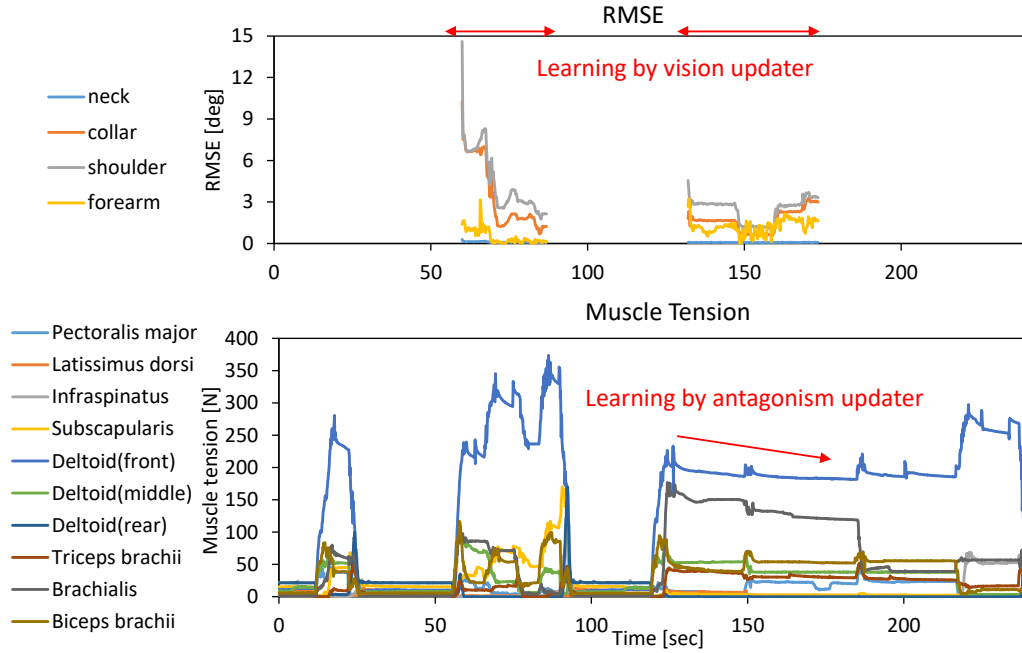


Fig. 6.43: The result of can grasping experiment. Upper graph shows RMSE of the difference between the estimated joint angles and the actual joint angles using the RGB camera; Lower graph shows the muscle tensions during this experiment [100].

は、 f^{imp} における作業空間剛性と指令作業空間剛性の誤差の評価値と、現状の関節トルクを維持するための評価値を $1 : \alpha$ の重みで足し合わせたものである。集めた N_{v1} 個のデータを E の値によって昇順に並べ、上から N_{v2} ($N_{v1} > N_{v2}$) 個のデータの f_{rand} の平均を f_{change} とする。このとき、 $E(\theta^{ref}, f + f_{change})$ が $E(\theta^{ref}, f)$ より小さい場合は f^{ref} を $f + f_{change}$ に置き換える。これを N_{v3} 回繰り返し、最終的に求めた f^{ref} を h に入力して所望の l^{ref} を得ることになる。ここで、 $N_{v2} = 1$ のときは山登り法であり、1 より大きい場合はより安定に剛性の探索が可能となる。本節では基本的に、 $\alpha = 0.02$, $N_{v1} = 10$, $N_{v2} = 2$, $N_{v3} = 50$ としている。

可変剛性制御に関する実験を3つ行う。まず、身体剛性の可変制御の実現性に関する評価を行う。 $(\theta_{S-r}, \theta_{S-p}, \theta_{S-y}, \theta_{E-p}, \theta_{E-y}) = (45, 0, 0, -90, 0)$ [deg] を θ^{ref} に設定し、剛性を変化させる実験を行った (S は肩, E は肘, rp は roll, pitch, yaw を表す)。この結果を Fig. 6.45 に示す。まず Sample 1 では、現状の身体剛性 (Current value) から長軸短軸に関して剛性が2倍になるような K_w^{ref} (Target value) を設定し、可変剛性制御によって現在の身体剛性を指令剛性に近づけるような探索を行った。探索の際の $E(\theta, f)$ の推移は Sample 1 の下図のようになっており、関節トルク誤差を抑えつつ、身体剛性を近づけることができています。その結果得られた身体剛性 (Calculated value) を実現する f^{ref} と関節角度-筋張

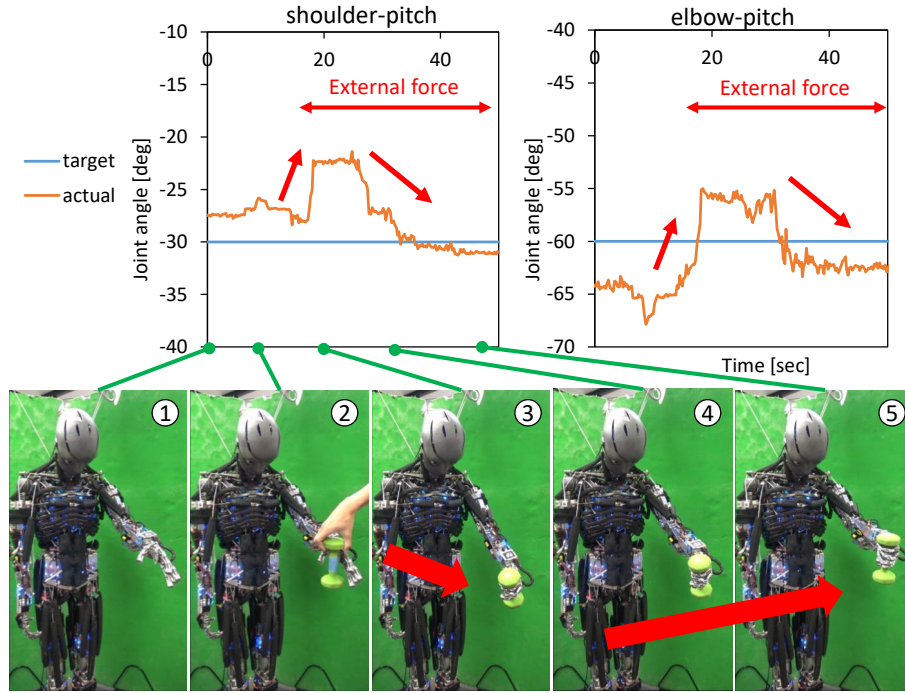


Fig. 6.44: Experiment of dumbbell grasping [124].

力-筋長マッピングを用いて実機を動作させた結果, 最終的な現状態での身体剛性 (Actual value) が観測される. Sample 1 では, 本手法によって, 指令した剛性楕円体に完全に一致した剛性楕円体を実現する f^{ref} は探索できなかったものの, 指令値に大きく近づくことに成功している (剛性楕円体は, $|F| = 10$ [N] のときの変位として記述している). また, その f^{ref} を指令として実機を動作させた後における剛性楕円体は, 探索された剛性楕円体とほぼ一致しており, マッピングの学習が正しくできていることがわかる. Sample 2 においては, 大きさと傾きを変えた剛性楕円体を指令値としたが, 探索の結果, 大きさは同等なものが実現できたが, 傾きには誤差が残ってしまった. これは, どんなに探索してもその指令剛性を実現できる解がなく, その中でも最良の選択をした結果だと考える.

次に, 可変剛性制御によって計算される作業空間剛性の理論値が現実とどの程度合致するかについて検証する. 身体剛性を測定しやすいよう, $(\theta_{S_r}, \theta_{S-p}, \theta_{S-y}, \theta_{E-p}, \theta_{E-y}) = (0, 30, 0, -60, 0)$ [deg] を θ^{ref} に設定し, 剛性が低い状態, 剛性が高い状態を可変剛性制御の手法により作り出す. このとき, xy 平面においてエンドエフェクタに対して, 周回 360 度を均等に分割した 8 方向から, フォースゲージの測定値が 10 N になるように力を加え, その時のエンドエフェクタの xy 平面における変位を計測した. この結果を Fig. 6.46 に示す. 剛性が高いときも低いときも, 理論値と実際の値がある程度合致していることがわかった. この結果における理論値と実際の値の誤差は, 学習されたモデルと実機の間に残っ

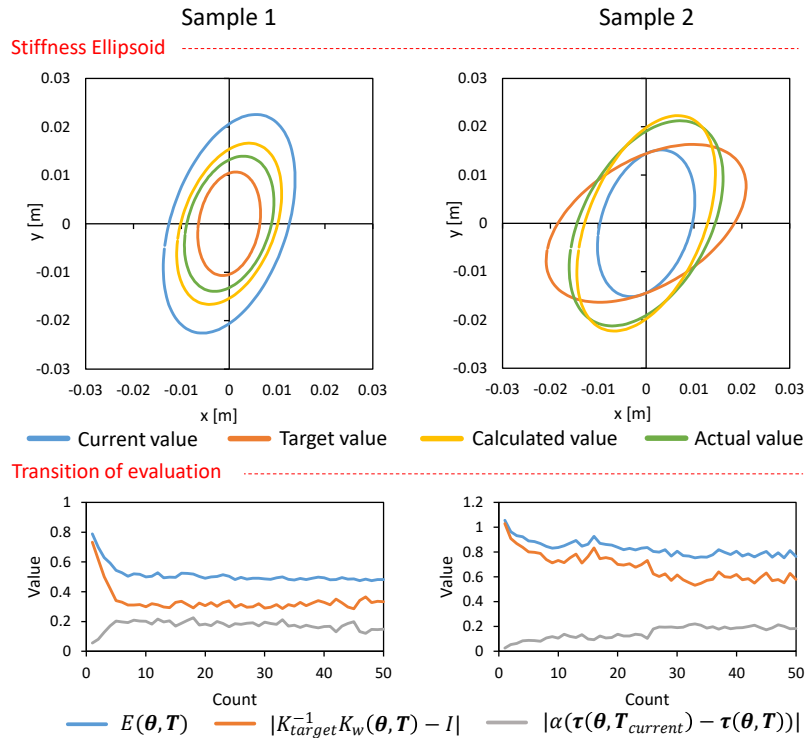


Fig. 6.45: Stiffness ellipsoid and transition of evaluation in the evaluation experiment of variable stiffness control using self-body image [125].

た誤差と、摩擦等によるヒステリシスが問題となっていると考えられる。

最後に可変剛性制御を用いた衝撃対応実験を行う。MusashiDarm の両手同士を握った状態において、肘上 1 m, 肘前 0.15 m から 50 N のボールを落とし、そのときの筋張力・関節角度の遷移を、低剛性・高剛性において比較実験する。実験の際の筋骨格双腕の動きを Fig. 6.47 に、左手における筋張力・関節角度の遷移を Fig. 6.48 に示す。Fig. 6.47 において、明らかに低剛性のときは大きく手が動き、高剛性のときは関節角度の変位が低剛性のときほど大きくないことがわかる。実際に、Fig. 6.48 の下図において、高剛性にするすることで、衝撃に対する Shoulder-p と Elbow-p の変位を小さくすることができる。また、その際の筋張力変化 (Fig. 6.48 の上図) においては、初期の筋張力が低剛性と高剛性で大きく違うため評価は難しいものの、低剛性では最大 150 N 程度、高剛性では最大 250 N 程度と、低剛性の方が衝撃を吸収できていることもわかる。

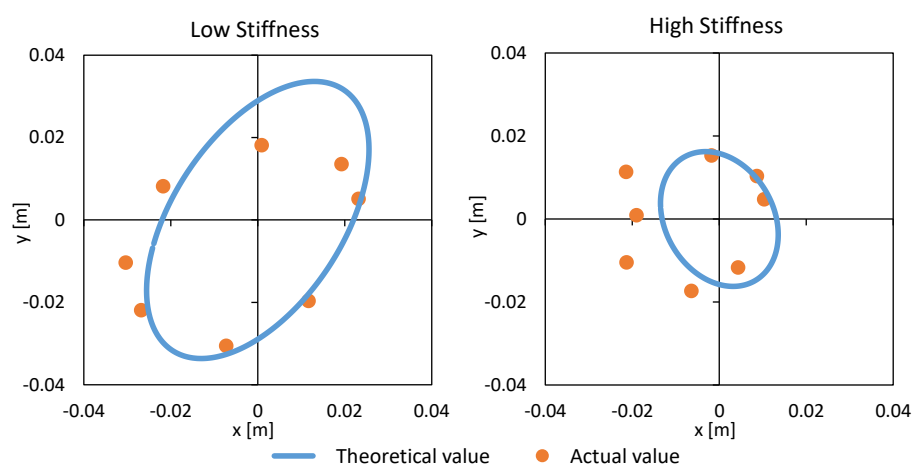


Fig. 6.46: Stiffness ellipsoid in the evaluation experiment of variable stiffness estimation using self-body image [125].

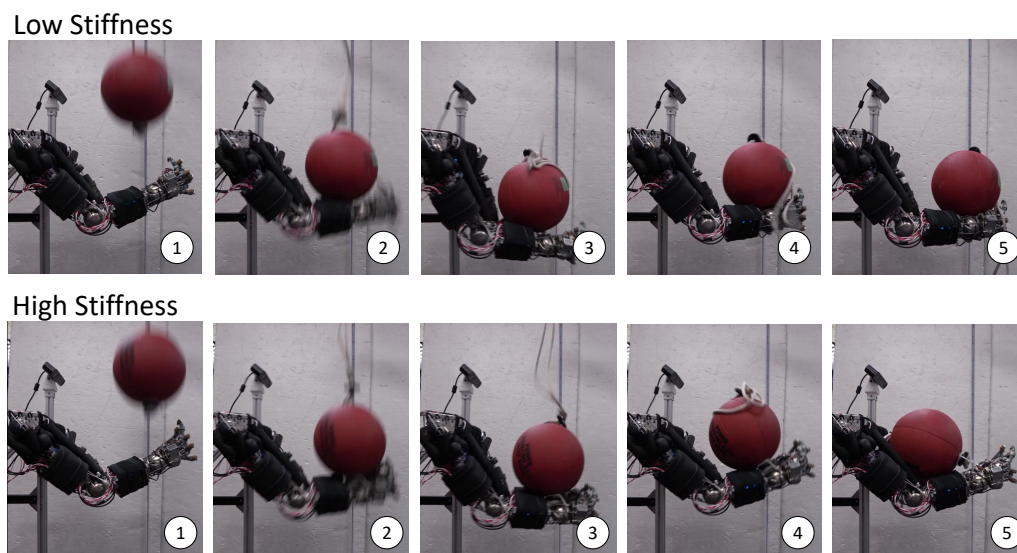


Fig. 6.47: Impact response experiment with variable stiffness control [125].

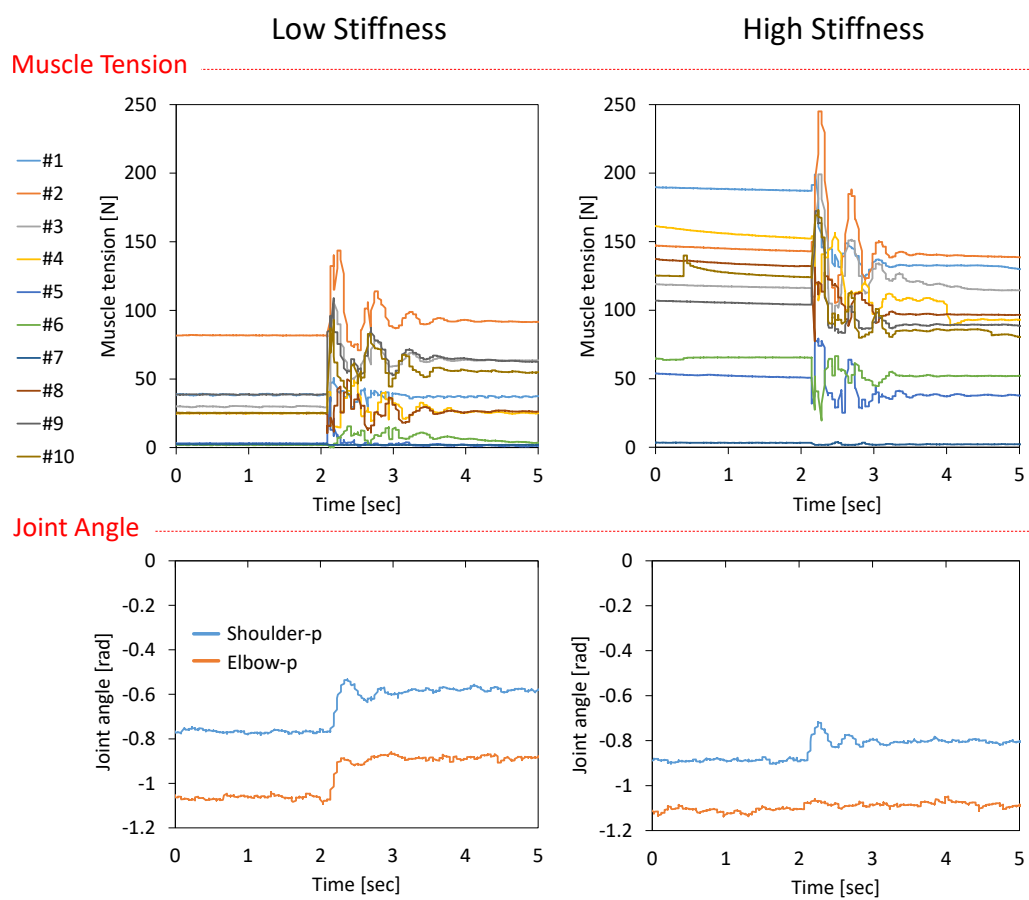


Fig. 6.48: Transition of muscle tensions and joint angles in the impact response experiment with variable stiffness control [125].

6.4 動的模倣動作学習 - 動作スタイルを考慮可能な模倣学習

6.4.1 概要と先行研究

ロボットには人間と同様に様々なタスクをこなすことが求められている。そのために様々なモデルベースな手法 [228] が開発されてきた一方、最近では深層学習を用いた手法が増え、強化学習 [50] や予測モデル [229] 等を使った手法が一般的になりつつある。この中でも、模倣学習 [230] は人間のデモンストレーションから直接動作を学習するため、効率よく人間のスキルを真似することができる。これまで、画像情報を使ったものや [23, 231], 力情報を組み合わせたもの等が開発されてきている [232]。また、Meta-Learning と組み合わせた研究 [233] や逆強化学習と Generative Adversarial Network を組み合わせた模倣学習手法 [37] 等も開発されている。

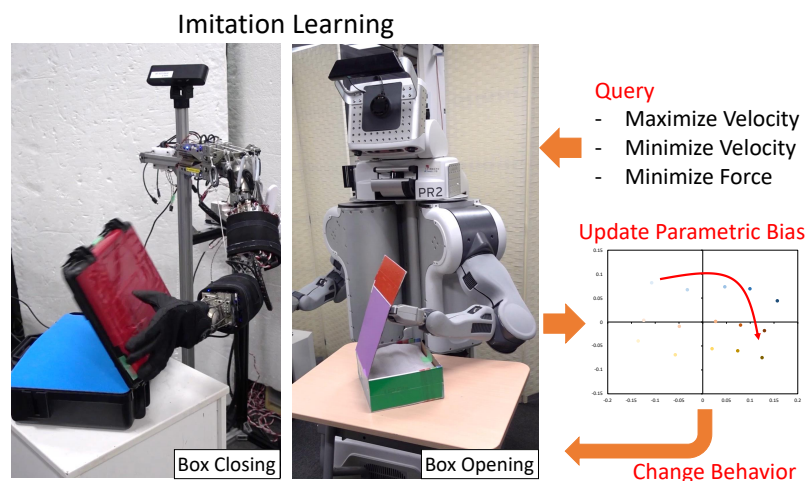


Fig. 6.49: Imitation learning with additional constraints on motion style [234].

この模倣学習は、人間の教示に依存して動作軌道や力の入れ具合等の動作スタイルが決定される。バラついた人間の教示の中でも、通常はその平均的な動作が再生されることになる。しかし、単純に平均的な動作を行うのみではなく、人間の動作のバラつきの中の偏りを再現したい場合がある。例えば、なるべく力がかからないようにしたり、速度・加速度を抑えたり、一部の関節をなるべく動かないようにしたり等である。これにより、その状況に合った形で振る舞いを変更していくことが可能となると考える。

動作スタイルの違いを模倣学習の中で考慮した手法とその問題点をまとめる。まず、Probabilistic Movement Primitives [49] や Dynamic Bayesian Network を用いた手法 [235], Probabilistic Principle Component Analysis を用いた手法 [236] 等のような、深層学習を用いない古典的な手法があり、動作速

度や障害物等の追加制約を考慮可能である。これらは感覚や制御入力 of 構造にある多くの仮定を置くため、少数のデモンストレーションから動作学習が可能である一方、生の画像や接触覚を含むような、より多次元でマルチモーダルな系にスケールしないという問題点がある。これらに対して、InfoGAIL [237] や OptionGAN [238], MSRD [239] 等は強化学習・逆強化学習の仕組みを用いて、生の視覚情報を扱った適応的な模倣学習が可能である。InfoGAIL は相互情報量を使って潜在空間に動作スタイルの違いを埋め込むことができ、OptionGAN はスタイルごとに Policy を、MSRD はスタイルごとに Reward を作成することで、動作スタイルを学習可能としている。一方、それら動作スタイルは離散的な値であるため、その動作スタイルを追加制約をもとに制御することはできない。また、強化学習を併用するため、デモンストレーションだけから模倣動作は学習できず、試行錯誤をしながら最適な Policy を獲得していく必要がある。その他にも、模倣学習ではないものの、Motion Style Transfer の文脈で動作スタイルを変化させることが行われてきている [240]。

そこで本研究では、Parametric Bias [72] を用いて追加制約を考慮可能な模倣学習手法を提案する (Fig. 6.49)。これまで Parametric Bias は複数のアトラクタダイナミクスを抽出する目的で使われてきた。模倣学習においては、それら複数のダイナミクスが訓練された後は、一度人間が動作を見せたり道具画像等を見せたりすることで、それに合致するようにダイナミクスを決定し、これを元に模倣動作を再生する。これらのようにただ模倣すべき提示された動きを模倣できるようにダイナミクスを変化させるのではなく、これに動作軌道や力の入れ具合等に関する追加の動作スタイル制約を考慮可能にする手法を本研究では開発する。通常の RNN を用いた模倣学習に Parametric Bias を追加し、複数の動作スタイルをこの Parametric Bias に埋め込むことで、動作スタイルを自在に制御することが可能となる。本研究ではこの模倣学習を 1 自由度腱駆動シミュレータ、筋骨格ヒューマノイド MusashiLarm [87], PR2 により検証し、その有効性を示す。本研究のネットワークは、動的な一般化多感覚相関モデルによって模倣学習を行った一例である。

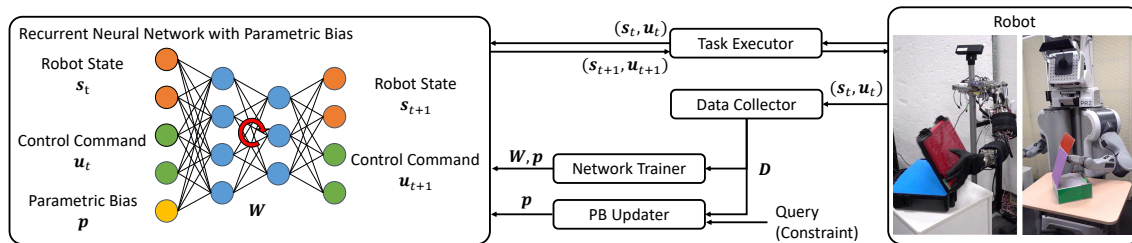


Fig. 6.50: The overall system of imitation learning with additional constraints [234].

6.4.2 動作スタイルに追加制約を考慮可能な模倣学習

本研究の全体システムを Fig. 6.50 に示す.

ネットワーク構造

本研究で扱うネットワークである Recurrent Neural Network with Parametric Bias (RNNPB) は, 以下のように式で表せる. 基本のネットワーク構造は [72] と同じである.

$$(s_{t+1}, u_{t+1}) = h(s_t, u_t, p) \quad (6.18)$$

ここで, t は現在のタイムステップ, s はセンサ状態, u は制御入力, p は Parametric Bias (PB), h は RNNPB を表す. s や u はロボットごとに異なり, 軸駆動型のロボットであれば u は関節角度指令値 θ^{ref} やトルク指令値 τ^{ref} が一般的に使われる. 腱駆動型のロボットによる模倣学習はこれまでに例がないが, 本研究では筋長指令値 l^{ref} を u として用いている. s は画像情報の他, 関節トルク τ や筋張力 f を入れても良い. 本研究では, 画像情報を入れる際は一度 AutoEncoder [227] を通して次元を圧縮し, s に用いる. Parametric Bias p は一つのデモンストレーション内では同一だが, 異なるデモンストレーション間では異なる学習可能な変数である. Eq. 6.18 には表していないが, 実際には 6.1 節の実験で得られたマスク変数 m が入力として存在している. それぞれのセンサ値は全て得られたデータを使って正規化してからネットワークに入力している.

本研究において RNNPB は 10 層とし, 順に 4 層の全結合層, 2 層の LSTM 層 [241], 4 層の全結合層からなる. ユニット数については, $\{N_u + N_s + N_p, 500, 300, 100, 100$ (LSTM のユニット数), 100 (LSTM のユニット数), $100, 300, 500, N_u + N_s\}$ とした (なお, $N_{\{u,s,p\}}$ は $\{u, s, p\}$ の次元数とする). 活性化関数は Tanh, 更新則は Adam [46] とした. 画像を圧縮する際は, 96×96 の RGB 画像について, カーネルサイズが 3, スライドが 2 の畳み込み層を 5 回適用し, 全結合層で順にユニット数 256, 12 まで次元を削減したあと, 同様に全結合層・逆畳み込み層によって画像を復元していく形を取っている. 最終層以外については Batch Normalization [45] が適用され, 活性化関数は最終層以外については ReLU [138], 最終層は Sigmoid, 更新則は Adam [46] とした. p の次元は, デモンストレーションの数よりも十分に小さいことを条件として, 実験ごとに適当に設定した. また, Eq. 6.18 の実行周期は 5Hz とする.

学習

構築した RNNPB の学習方法について述べる. シミュレーションについては人間が予め決めた動作を行い, 実機については VR 機器等を用いて人間がロボットに動作を教示し, その際の s, u のデータを取

得していく. 一回の試行 k について, データ $D_k = \{(s_1, u_1), (s_2, u_2), \dots, (s_{T_k}, u_{T_k})\}$ を得る ($1 \leq k \leq K$, K は全試行回数, T_k はその試行 k に関する動作ステップ数とする). そして, 学習に用いるデータ $D_{train} = \{(D_1, p_1), (D_2, p_2), \dots, (D_K, p_K)\}$ を得る. p_k はその試行 k に関する Parametric Bias であり, その一回の試行中については共通の値で, 異なる試行については別の値となる変数である. このデータ D_{train} を用いて RNNPB を学習させる. 通常の模倣学習のようにネットワークの重み W のみを更新するのではなく, 同時に p_k も更新していく. なお, 学習の際の損失関数は平均二乗誤差であり, 全 p_k は初期値を 0 として最適化される.

通常の模倣学習では学習された中における平均的な動作が実行されるが, p_k にはその試行における人間の動作スタイルが反映されることになる. 例えば, 冗長性ゆえに手先で物体を操作するときには肘の角度は人それぞれであり, その動作の実行速度も人それぞれである. よって, p を変化させることで, デモンストレーションのばらつきの範囲内ではあるが, 追加の制約を考慮することが可能となる. なお, 実際にはある試行 k 内でも途中で動作スタイルが変わることがあるが, 本研究ではそれらをまとめて一つのスタイルとしてしまっている.

また同様に p によって, 本研究の本題ではないが, 筋骨格ヒューマノイドのような柔軟身体で発生しやすい, 初期化の再現性の無さによる状態の違いや経年変化等 [146] についても考慮することができる. これにより, ロボットの逐次的変化を考慮して正確に動き続けることが可能になる. ロボットの身体状態を変化させながらデータを取得していくことで, それらの変化を Parametric Bias に埋め込むことができる. タスク実行時には, p を変化させることで, ネットワークを現在のロボット状態に合致させる制約を追加することが可能となる.

タスク実行と Parametric Bias の更新

タスクの実行方法はシンプルである. 現在のセンサ状態 s_t と制御入力 u_t を取得し, Eq. 6.18 を順伝播することで u_{t+1} を得て, これを実機に指令することを繰り返すのみである.

このタスク実行と同時にオフラインまたはオンラインで Parametric Bias の更新も行う. 本研究では, 前述のロボット状態の変化による Parametric Bias の違いを (A), 人間のデモンストレーションにおける動作スタイルのばらつきによる Parametric Bias の違いを (B) とする. (A) については, ロボットの状態変化によって状態遷移モデル $((s_t, u_t) \rightarrow s_{t+1})$ が変化するため, 実際に動作したときの (s, u) と合致するように, p を更新する. (B) については, 筋張力最小化や速度最大化等, 追加の制約を考える必要がある. これは, (A) はタスク達成に直接影響するのに対して, (B) はタスク達成に関する Null Space であるということを表す. よって, 一度 $p = \mathbf{0}$ でタスクを実行してデータ

$D = \{(s_1^{data}, \mathbf{u}_1^{data}), (s_2^{data}, \mathbf{u}_2^{data}), \dots, (s_T^{data}, \mathbf{u}_T^{data})\}$ が得られた時に, A と B について以下のように損失関数を定義することで \mathbf{p} を更新していく.

$$L = \|\mathbf{s}_{2:T}^{data} - \mathbf{s}_{2:T}^{pred}\|_2 + \alpha L_{constraint}(\mathbf{x}_{2:T}^{pred'}) \quad (6.19)$$

ここで, \mathbf{x} は $\begin{pmatrix} \mathbf{s}^T & \mathbf{u}^T \end{pmatrix}^T$, $\mathbf{x}_{a:b}$ は $[a, b]$ の間の \mathbf{x} を並べたもの, $\mathbf{s}_{2:T}^{pred}$ は $\mathbf{x}_{1:T-1}^{data}$ を順に \mathbf{h} に通して得られた \mathbf{s} の値, α は重みの係数, $L_{constraint}$ は (B) の追加制約に関する損失関数を表す. また, $\mathbf{x}_{2:T}^{pred'}$ は \mathbf{x}_1^{data} から始め, \mathbf{h} に通し, その出力を \mathbf{h} に入力するという自己回帰を繰り返した際に得られた \mathbf{x} の値である. Eq. 6.19 の右辺第一項は (A) に関する損失であり, Eq. 6.18 の出力 \mathbf{u}_{t+1} を無視し, \mathbf{s} の予測モデル誤差を最小化するように \mathbf{p} を更新する. Eq. 6.19 の右辺第二項は (B) に関する損失であり, \mathbf{p} が変化することによって入力の \mathbf{u} が変化するため, データ D は初期値のみしか用いず, 自己回帰によって計算を進める必要がある. もし (B) のみに関して損失を取るのであれば, 一度動作を試行する必要はなく, 現在状態 $(s_1^{data}, \mathbf{u}_1^{data})$ のみ与えても良い. $L_{constraint}$ には様々な定義が可能だが, その一例を以下に示す.

$$L_{constraint}^{tension} = \|\mathbf{f}_{2:T}^{pred}\|_2 \quad (6.20)$$

$$L_{constraint}^{lvelocity} = \|\mathbf{l}_{3:T}^{pred} - \mathbf{l}_{2:T-1}^{pred}\|_2 \quad (6.21)$$

$$L_{constraint}^{jvelocity} = \|\boldsymbol{\theta}_{3:T}^{pred} - \boldsymbol{\theta}_{2:T-1}^{pred}\|_2 \quad (6.22)$$

ここで, \mathbf{f}^{pred} は \mathbf{x}^{pred} が筋張力 \mathbf{f} を含む場合のその値, $\boldsymbol{\theta}^{pred}$ は \mathbf{x}^{pred} が関節角度 $\boldsymbol{\theta}$ を含む場合のその値である. 筋張力を最小化したい場合は α を正として Eq. 6.20 を用い, 筋張力を高めにして剛性を保ちたい場合は α を負とすれば良く, これは速度についても同様である. この他にも, Eq. 6.19 に制約として $\|\mathbf{p}\|_2$ の項を加えることで, \mathbf{p} が意図しない値に収束しないように制約することもできる. \mathbf{p}_k は基本的に $\mathbf{0}$ 周辺に集まり, $\mathbf{p} = \mathbf{0}$ のときが最も平均的な動作になるため, そこから大きく外れないように最小化制約を追加する. 実際に \mathbf{p} を更新する際は, ネットワークの重み W を固定して \mathbf{p} のみ Momentum SGD [139] で更新する. オフラインで Parametric Bias を更新する際は, 一度 $\mathbf{p} = \mathbf{0}$ の状態で動作を行い, その際に得られたデータ D を用いて \mathbf{p} を更新する. オンラインで Parametric Bias を更新する際は, データ数 N_{online} が閾値 N_{thre}^{online} を超えてから, 蓄積したデータ D を用いて \mathbf{p} を更新する. なお, N_{max}^{online} を超えたデータは古いものから破棄していく.

本研究では Eq. 6.20–Eq. 6.22 にあるように, 速度や力の最小化・最大化のみを扱っているが, 実際にはある指令値を与え, これに速度や力が近くなるように \mathbf{p} を更新することも可能である. しかし, 例えば箱を開けるときはその一つの動作でも常に速度は一定ではなく様々に変化するため, ある指令値

を与えても、それが実現できるとは限らず、全体的に見たときの速度が概ね指令値に近い、という程度にしかならない。ゆえに、本研究では最小化・最大化によって値を操作し、そのときの係数 α の大きさによって、大まかに速度に関する指令値を与えるという形にしている。

本研究では、 α や $L_{\text{constraint}}$ は実験によって異なり、 $N_{\text{thre}}^{\text{nonline}} = 10$, $N_{\text{max}}^{\text{nonline}} = 20$ とした。

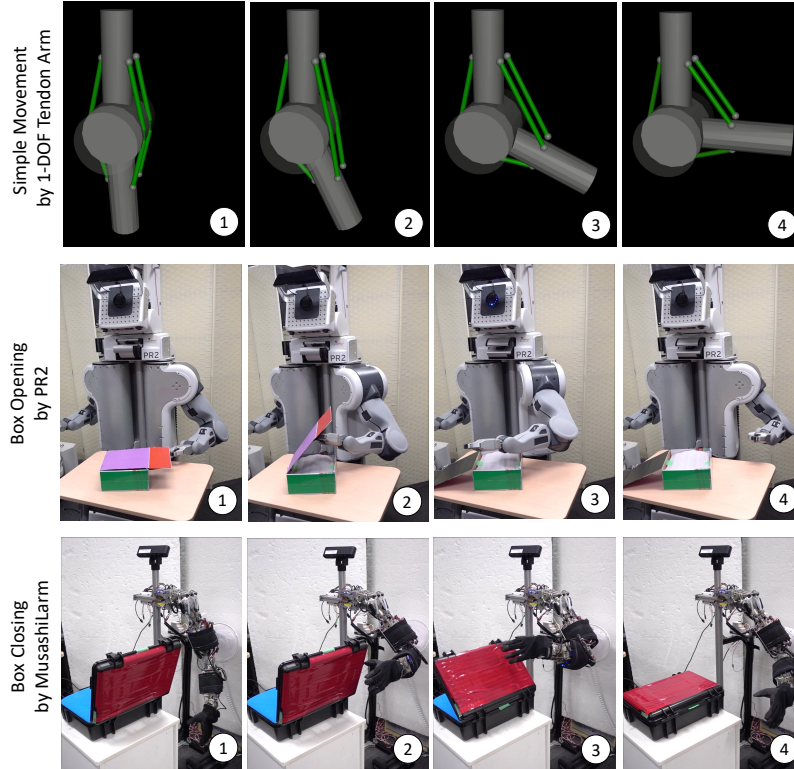


Fig. 6.51: Experimental setup [234]: simple movement by 1-DOF tendon arm, box opening by PR2, and box closing by MusashiLarm.

6.4.3 実験

実験セットアップ

Fig. 6.51 にそれぞれの実験のセットアップを示す。

まず、1 自由度 3 筋の簡易的な筋骨格ロボットのシミュレーションにより、基本的な性能を示す。本シミュレーションは筋に非線形弾性ばねの要素と摩擦の影響を入れ、モデル化を困難にしている。タスクは関節角度を 0 から -90 度まで動かすという簡単な動きである。デモンストレーションの動作は、 $\theta_{i+1}^{ref} \leftarrow \theta_i^{ref} + \beta(\theta^{task} - \theta_i)$, $f_{i+1}^{ref} \leftarrow f_i^{ref} + \beta(f^{style} - f_i^{ref})$ という形で指令関節角度と指令筋張力

を更新し, $\boldsymbol{l}^{ref} = \boldsymbol{h}_{bodyimage}(\boldsymbol{\theta}^{ref}, \boldsymbol{f}^{ref})$ の形で指令筋長を送ることを繰り返すフィードバック制御である ($\boldsymbol{\theta}^{task} = -90$ [deg] は実現したい関節角度, $\boldsymbol{\theta}_t$ は測定された現在関節角度, \boldsymbol{f}^{style} は教示のスタイルとしての筋張力指令, β はフィードバックの割合を表す係数を表す). ここで, $\boldsymbol{h}_{bodyimage}$ は [125] によって学習される指令関節角度・指令筋張力 \boldsymbol{f}^{ref} から筋長への変換関数である. なお, このモデルは静的な関係しか考慮しておらず, 直接 $\boldsymbol{l}^{ref} = \boldsymbol{h}_{bodyimage}(\boldsymbol{\theta}^{task}, \boldsymbol{f}^{ref})$ を送っても $\boldsymbol{\theta}^{task}$ を実現できないため, このようなフィードバック制御を行っている. β と \boldsymbol{f}^{style} を任意に選ぶことで速度や筋張力を調整でき, これを変化させて本手法の検証を行う (\boldsymbol{f}^{ref} はそれぞれの筋について独立に設定でき, 重力補償可能な筋張力を設定すべきであるが, 本研究では簡単のため全筋に対する指令筋張力を同じ値としている. これらは, PB 更新における (B) に相当する). また, 1 自由度関節の半径 r を変化させることで, PB 更新における (A) のロボット状態の変化を作り出すことが可能であり, これも検証に用いる. なお, RNNPB において, $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{\theta}^T & \boldsymbol{f}^T \end{pmatrix}^T$, $\boldsymbol{u} = \boldsymbol{l}^{ref}$ であり, \boldsymbol{p} は 2 次元とした (本実験では, \boldsymbol{p} のプロットをわかりやすくするため, デモンストレーションの動作スタイルの次元と一致させている).

次に, 台車型双腕型ロボット PR2 について実験を行う. タスクは, 7 自由度の左腕を使って, 机の上の箱を開ける動作である. 右手を重力補償トルクのみをかけた状態にし, 右手と対称な関節角度を左手に指令するような制御を行う. 人間が PR2 の右手を動かすことで PR2 の左手によりデモンストレーションを行い, 模倣学習を行う. その際, 箱を様々な方向に回転させてデータを取る. なお, RNNPB において, $\boldsymbol{s} = \boldsymbol{z}$, $\boldsymbol{u} = \boldsymbol{\theta}^{ref}$ であり (\boldsymbol{z} は画像を AutoEncoder により圧縮した値とする), \boldsymbol{p} は 3 次元とした.

最後に, 筋骨格ヒューマノイド MusashiLarm [87] について実験を行う. タスクは, 7 自由度 18 筋の左腕を使って, 机の上の箱を閉じる動作である. 人間が操作する HTC Vive のコントローラから手先座標を取得し, これに対して逆運動学を解いて指令関節角度 $\boldsymbol{\theta}^{ref}$ を求め, $\boldsymbol{l}^{ref} = \boldsymbol{h}_{bodyimage}(\boldsymbol{\theta}^{ref}, \boldsymbol{f}^{ref})$ により指令筋長を送ってデモンストレーションを実行する (ここでは $\boldsymbol{f}^{ref} = 10$ [N] で一定としている). その際, 箱を様々な方向に回転させてデータを取る. なお, RNNPB において, $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{z}^T & \boldsymbol{f}^T \end{pmatrix}^T$, $\boldsymbol{u} = \boldsymbol{l}^{ref}$ であり, \boldsymbol{p} は 3 次元とした.

一自由度腱駆動シミュレーションによる単純動作実験

まず, 1 自由度関節の半径 r , 動作の筋張力 \boldsymbol{f}^{style} を変化させて実験を行った. $r = \{0.03, 0.035, 0.04\}$ [m], $\boldsymbol{f}^{style} = \{10, 50, 100, 150, 200\}$ [N] の 15 の組み合わせについてそれぞれ約 60 ステップのデータを取得し, これを用いて RNNPB を学習させた. 訓練時に得られた \boldsymbol{p}_k を PCA を通して 2 次元で表示したものを Fig. 6.52 に示す. r と \boldsymbol{f}^{style} の大小に沿って, \boldsymbol{p}_k が規則的に整列していることがわかる. シミュレータを $r = 0.04$ で固定し, $\boldsymbol{p} = \mathbf{0}$ の状態で一度動作してデータ D を取得して, 追加の制約が

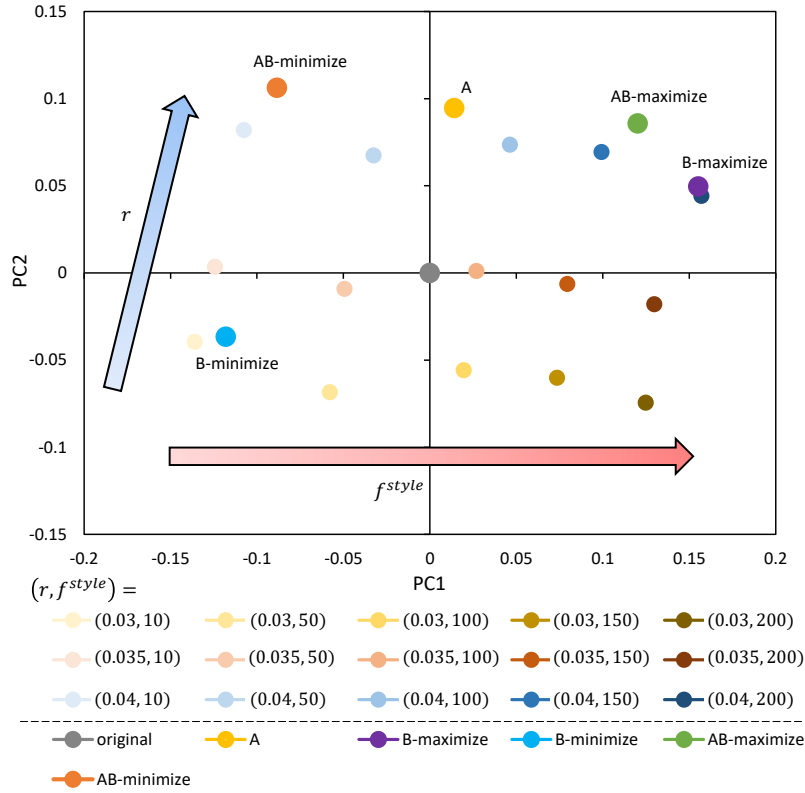


Fig. 6.52: Simple movement experiment by simulated 1-DOF tendon arm when changing r and f^{style} [234]: parametric biases trained in data collection phase and those updated from additional constraints regarding “A”, “B-maximize”, “B-minimize”, “AB-maximize”, and “AB-minimize”.

らオフラインで更新した p の値も同様に Fig. 6.52 に示す。ここでは、 r の変化が PB 更新の (A) に、 f^{style} の変化が (B) に対応するため、(A) の合致制約を入れるか否か、(B) の制約について最大化するか・最小化するか・制約を入れないかの組み合わせである、A, B-maximize, B-minimize, AB-maximize, AB-minimize について p を示している。 p の更新の際は $\alpha = 0.1$ (最大化の際は符号を逆にする)、学習率は 0.01、エポックは 30 とし、(B) については Eq. 6.20 の制約式を用いている。また、original は $p = 0$ を表す (なお、PCA によって変換されるため、 $p = 0$ がグラフの原点にあるとは限らない)。(A) を考慮したものは全て、学習時に得られた $r = 0.04$ のときの p_k に近い値を取っており、現在の r の値を正しく認識できていることがわかる。また、(B) を考慮したものは、最大化であれば $f^{style} = 200$ [N] の p_k 付近に、最小化であれば $f^{style} = 10$ [N] の p_k 付近に p が移動していることがわかる。ここで得られた p を使ってタスクを行ったときの $\|\theta^{task} - \theta_t\|_2$, $\|f_t\|_2$ を Fig. 6.53 に示す。なお、以降の実験では、オンライン学習実験の間は p は連続的に更新される一方、Fig. 6.53 のような評価実験の際は、更

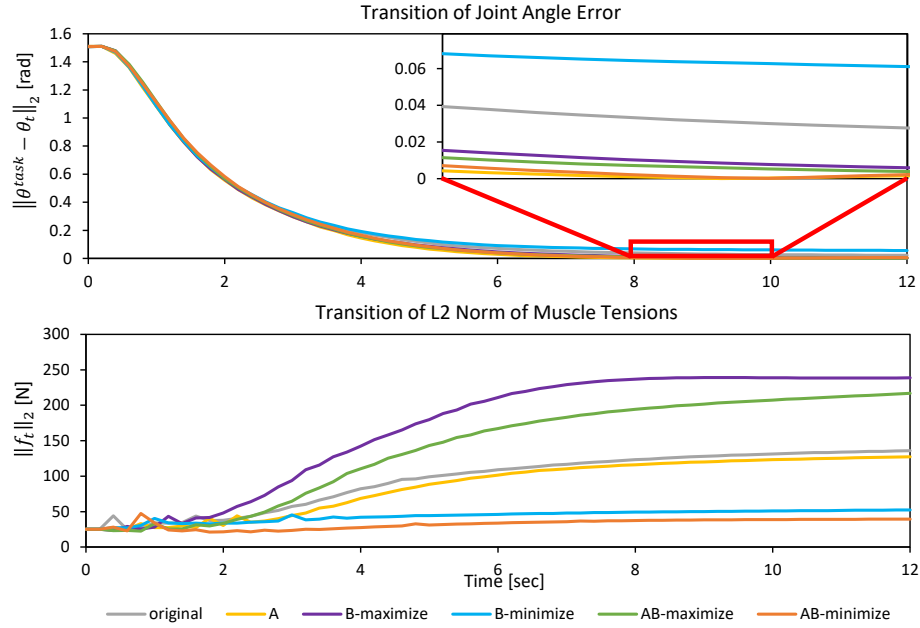


Fig. 6.53: Evaluation of simple movement experiment by simulated 1-DOF tendon arm when changing r and f^{style} [234]: transition of $\|\theta^{task} - \theta_t\|_2$ and $\|f_t\|_2$.

新する前と後の p を固定したうえで、オンライン学習を行わない状態で実験することで、一貫性のあるグラフを得ている。 $r = 0.04$ の際の p_k から遠い、original と B-minimize の p では、 θ_t が θ^{task} まで到達しておらず、タスク達成に誤差がのってしまっていることがわかる。また、(B) を minimize した p では f_t が小さく、(B) を maximize した p では f_t が大きく、(B) に対して制約をかけていない p では、その中間程度の f_t を示していた。B-maximize については、(A) の制約は入っていないが、偶然にも $r = 0.04$ の p_k 付近の p が得られたため、AB-maximize と同じような挙動になっている。

次に、 $r = 0.04$ で固定し、動作の筋張力 f^{style} と、動作速度を表す係数 β を変化させて実験を行った。 $f^{style} = \{10, 50, 100\}$ [N]、 $\beta = \{0.05, 0.10, 0.15\}$ の 9 の組み合わせについてそれぞれ約 60 ステップのデータを取得し、これを用いて RNNPB を学習させた。訓練時に得られた p_k を PCA を通して 2 次元で表示したものを Fig. 6.54 に示す。 f^{style} と β の大小に沿って、 p_k が規則的に整列していることがわかる。先の実験と同様に追加の制約からオフラインで更新した p の値も Fig. 6.54 に示す。ここでは、 f^{style} と β について、Eq. 6.20 と Eq. 6.21 の制約式を用いて、それぞれについて最大化、最小化を行う 4 つの組み合わせについて実験を行った。 p の更新の際は $(\alpha_1, \alpha_2) = \{(0.02, 0.3)$ (なお、 f^{style} に関する重みを α_1 、 β に関する重みを α_2 とする)、学習率は 0.01、epoch は 30 とした。なお、以降の実験では基本的に (B) の制約についてのみ考えるが、Eq. 6.19 の (A) の制約式も含めて p を更新している。

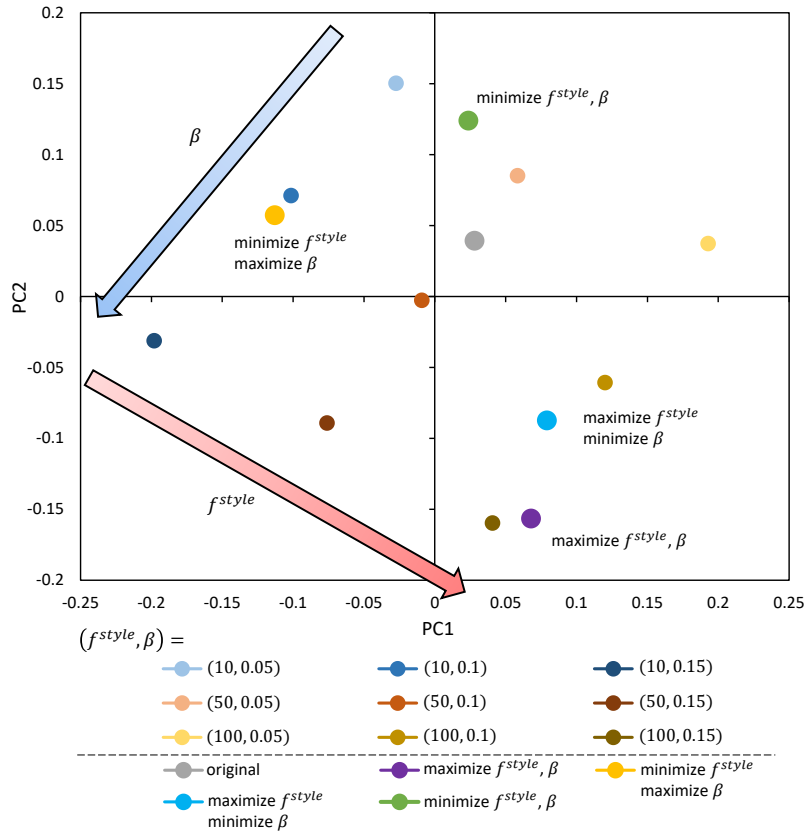


Fig. 6.54: Simple movement experiment by simulated 1-DOF tendon arm when changing f^{style} and β [234]: parametric biases trained in data collection phase and those updated from additional constraints.

f^{style} の方向については最小化・最大化によって比較的意図した通りの方向に \mathbf{p} が更新された。それに対して、 β の方向については、一部意図した通りの方向に \mathbf{p} は更新されていない。最適化ごとに α_1 と α_2 を調整することで所望の結果を得ることはできるものの、2つ以上の (B) の制約を考慮するためには、注意深い重みの比率の調整が必要ながわかった。得られた \mathbf{p} を使ってタスクを行ったときの $\|\theta^{task} - \theta_t\|_2$, $\|f_t\|_2$ を Fig. 6.55 に示す。Fig. 6.54 における f^{style} の軸に関する \mathbf{p} の配置結果と同じく、 $\|f_t\|_2$ はほとんどの場合において最小化・最大化によって意図した通りに制御できているが、 f^{style} と β を両方最小化した場合は f^{style} が最小化し切れていなかった。 $\|\theta^{task} - \theta_t\|_2$ については、 f^{style} を最小化した場合、 f^{style} を最大化した場合のそれぞれ同じ条件下では、速度最大化・最小化によって筋長速度が正しく変更できている。なお、もともと f^{style} を大きくすると拮抗が強まり、 f^{style} が小さい場合と比べると速度は出ない。

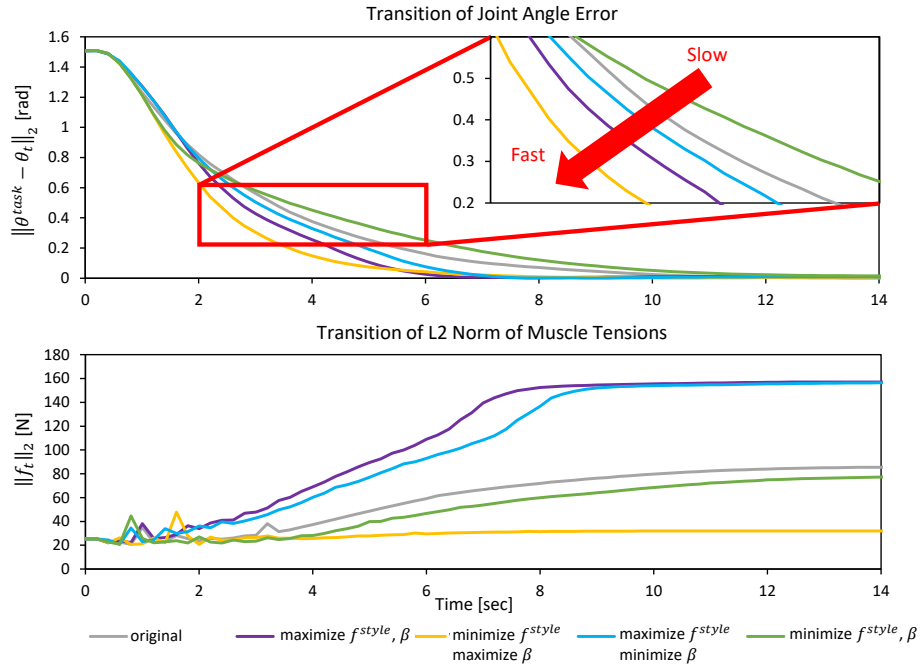


Fig. 6.55: Evaluation of simple movement experiment by simulated 1-DOF tendon arm when changing f^{style} and β [234]: transition of $\|\theta^{task} - \theta_t\|_2$ and $\|f_t\|_2$.

PR2 による箱開け実験

得られた 42 試行分のデータを使って学習させた際の動作の様子を Fig. 6.56 に、訓練された p_k を Fig. 6.57 に示す. Fig. 6.56 では、箱を開けると手を引き、閉じると開けようとする、また、外乱を加えてもタスクを行うことができた. このとき、別々の箱の角度において、速度を最大化 (1)、最小化 (2) する制約を加えて p をオンライン更新したときの p の軌跡を Fig. 6.57 に示す. 最小化・最大化によって、別方向に p が動いていることがわかる. この (1), (2) で更新された p を使ってタスクを行ったときの

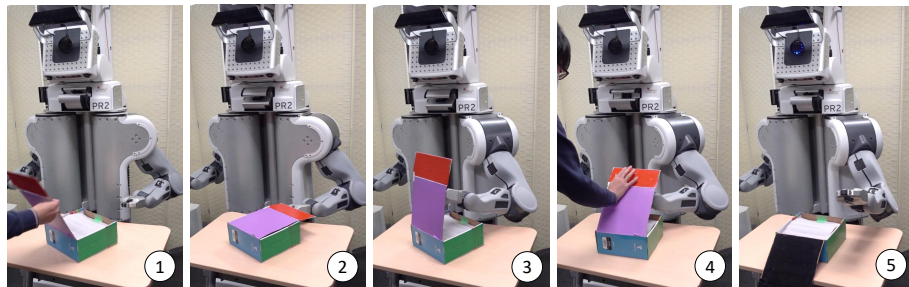


Fig. 6.56: The trained behavior of box opening by PR2 [234].

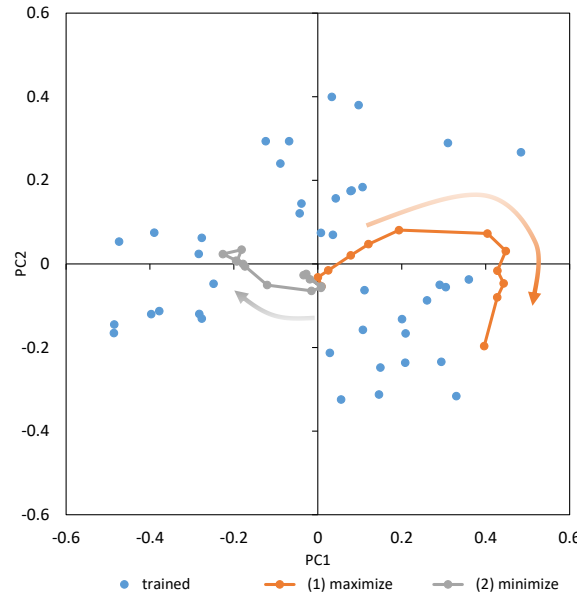


Fig. 6.57: Box opening experiment by PR2 [234]: parametric biases trained in data collection phase and their trajectories updated from additional constraints regarding (1) and (2).

$\|\theta_t - \theta^{init}\|_2$ の遷移を Fig. 6.58 に示す. なお, 全ての動作において, 箱を開けるタスクは完遂できていた. ここでは, \mathbf{p} を更新する前と後について, 3 回の動作の軌道 (“before-{1, 2, 3}” と “after-{1, 2, 3}”) と, その平均 (“before-ave” と “after-ave”) を示している. なお, 動作の都合上, 箱を開いた状態を 3 秒見せてから箱を閉じて動作を始めているため, $\|\theta_t - \theta^{init}\|_2$ の初期値は 0 rad ではない. (1) では, \mathbf{p} の更新前の, 途中でゆっくり停滞しながら動く動作が消え, 箱を開けてすぐに初期姿勢に戻るような動作が生成された. (2) では, 最初の動き出しから箱を開けるまでの動作が, 更新前に比べて非常にゆっくりになった.

筋骨格アーム MusashiLarm による箱閉じ実験

得られた 40 試行分のデータを使って学習させた際の動作の様子を Fig. 6.59 に, 訓練された \mathbf{p}_k を Fig. 6.60 に示す. Fig. 6.59 では, 箱を閉めると手を引き, 開けると閉めようとする, また, 外乱を加えてもタスクを行うことができた. このとき, それぞれ別々の箱の角度で速度を最大化 (v1, v2), 筋張力を最小化 (f1, f2) する制約を加えてオンライン学習させたときの \mathbf{p} の軌跡を Fig. 6.60 に示す. (v1) と (v2) は同じ方向へ \mathbf{p} が動いていることがわかるが, (f1) と (f2) は $\mathbf{p} = \mathbf{0}$ からそれほど大きく動いていない. (v1) と (v2) で得られた \mathbf{p} を使ってタスクを行ったときの $\|\mathbf{l}_t - \mathbf{l}^{init}\|_2$ の遷移を Fig. 6.61 に示す. PR2 による実験と同様に, \mathbf{p} を更新する前と後について, 3 回の動作の軌道と, その平均を示してい

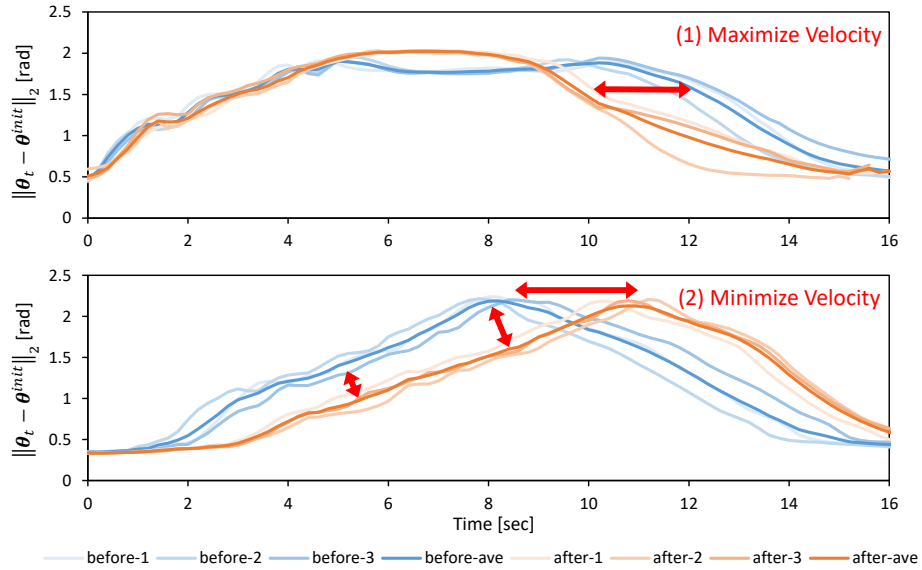


Fig. 6.58: Evaluation of box opening experiment by PR2 [234]: transition of $\|\theta_t - \theta^{init}\|_2$ when using p before and after online update regarding (1) and (2).

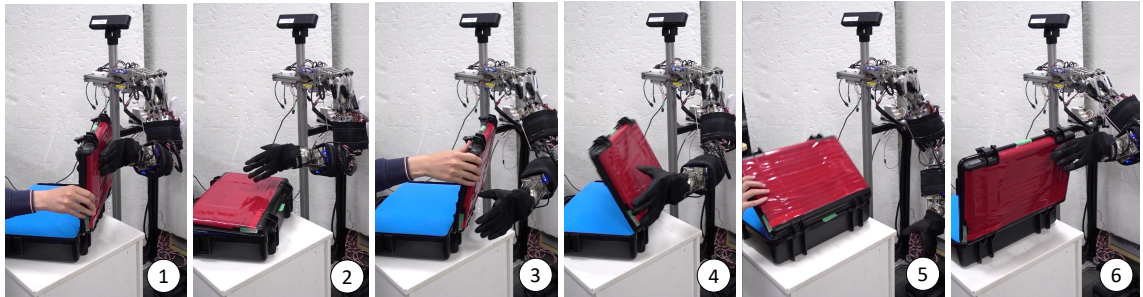


Fig. 6.59: The trained behavior of box closing by MusashiLarm [234].

る. 大幅にタスク実行速度が速くなり, 正しく制約が加えられていることがわかる. また, (f1) と (f2) で得られた p を使ってタスクを行ったときの $\|f_t\|_2$ の遷移を Fig. 6.62 に示す. (f1) では更新前に比べて筋張力が多少小さな動作に変更されたが, (f2) では更新前とあまり変わらない結果が得られた.

6.4.4 議論

まず, 1 自由度腱駆動ロボットシミュレーションから, 本手法によって, 筋張力や動作速度に関する最小化・最大化制約を考慮できること, また, 関節半径等のロボット状態の合致制約等も同時に考慮可能なことがわかった. 得られたデモンストレーションのデータから RNNPB を学習させることで, p_k が規則的に自己組織化される. 動作スタイル・ロボット状態に関する制約を考慮して p を更新するこ

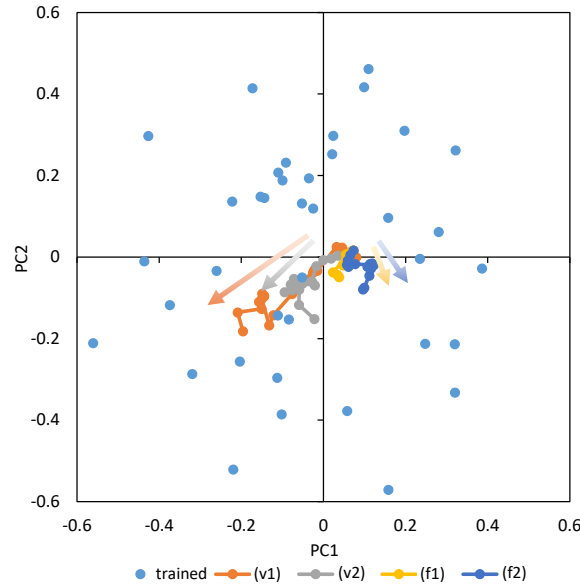


Fig. 6.60: Box closing experiment by MusashiLarm [234]: parametric biases trained in data collection phase and their trajectories updated from additional constraints regarding (v1), (v2), (f1), and (f2).

とで、意図した動作スタイル・ロボット状態の p を求めることができる。実際にこの更新された p を用いることで、タスク実行時の筋張力・速度を意図した通りに制御できることがわかった。同時に、動作スタイルに関する制約が2つ以上ある場合は、それらの重みの比率の調整が難しいということもわかった。

次に、PR2 による箱開け動作により、多自由度の軸駆動型ロボットにおいても本研究が適用可能であること、 p はオフラインだけでなくオンラインでも更新可能であることがわかった。速度最大化・最小化によって p が逆方向に更新されることから、それら動作スタイルに関する情報が正しく自己組織化されていると考えられる。実際に得られた p を使ってタスクを実行することで、速度が意図したように調整出来ており、本研究の有効性が示された。

最後に、MusashiLarm による箱閉め実験により、多自由度の腱駆動型ロボットにおいても本研究が適用可能であることがわかった。異なる箱の角度において、速度最小化により p が同じ方向に更新されることから、PR2 の箱開け実験と同様に、動作スタイルに関する情報が正しく自己組織化されていると考えられる。実際に得られた p を使ってタスクを実行することで、速度が意図したように調整出来ており、本研究の有効性が示された。一方、筋張力に関しては速度ほど意図した調整ができなかったが、これはデモンストレーション内で、動作速度に関しては様々なバリエーションがあったのに対して、筋張力についてはそれほど違いが得られなかったからであると考えられる。本研究では明示的に

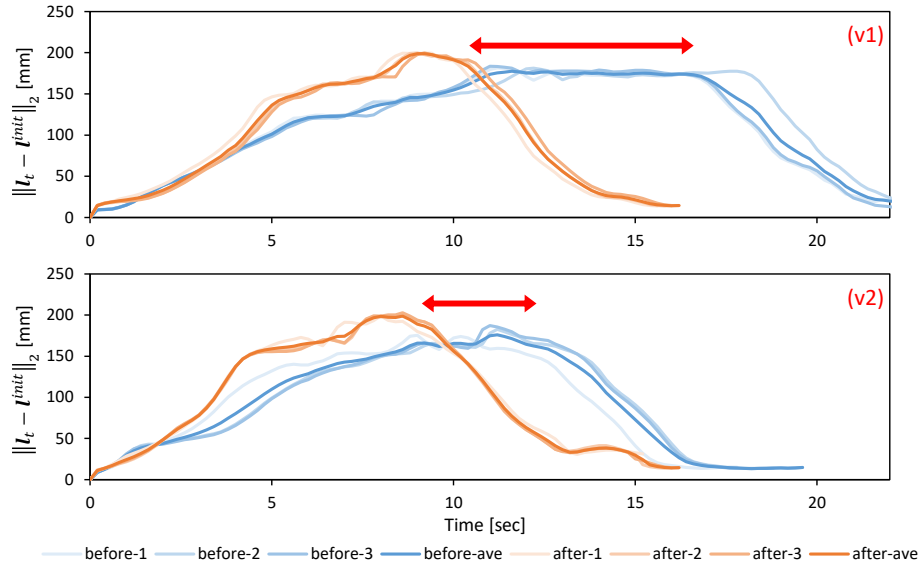


Fig. 6.61: Evaluation of box closing experiment by MusashiLarm [234]: transition of $\|l_t - l^{init}\|_2$ when using p before and after online update regarding (v1) and (v2).

f^{ref} について変化を加えなかったが、例えば人間の筋電からロボットの筋張力入力を変化させるようにすると、明示的に f^{ref} が変化し、より明確に筋張力の制約を考慮可能になると考えられる。また、本研究の目的とは直接は関係ないが、冗長な筋を持つ腱駆動型による模倣学習は初めてであり、柔軟な身体を活かした模倣学習が今後より期待される。

本研究の限界について述べる。まず、本研究はあくまでデモンストレーション中に得られたばらつきの範囲内ではしか動作スタイルに制約を加えることができない。そのため、ほとんど同じスタイルでしか実行できないような曖昧性のないタスクへの適用は難しく、また、デモンストレーションのばらつき以上に速度を遅くしたり、力を弱めたりすること等はできない。デモンストレーション時にバラエティーに富んだ動作を行うような操作デバイス側の工夫等が必要な可能性がある。また、無理なオンライン学習により p が $\mathbf{0}$ から大きく外れた場合、如何にタスク達成の挙動が変化するのか、人間による干渉や邪魔等による適応性はどの程度変化するのかについても検証が必要である。次に、本研究では、ネットワークから出力されない値には損失関数を定義できないため、定量化できない動作スタイルに適用できない点が挙げられる。一方、定量化できない動作スタイルに対してアノテーションが行われていれば、アノテーションされた動作スタイルの指令値を決め、その周辺の Parametric Bias を使うことで、これを扱うことは可能になる。最後に、一回のデモンストレーション時の動作スタイルを一つの Parametric Bias に埋め込むため、途中でスタイルが変わる場合等を考慮できないという点であ

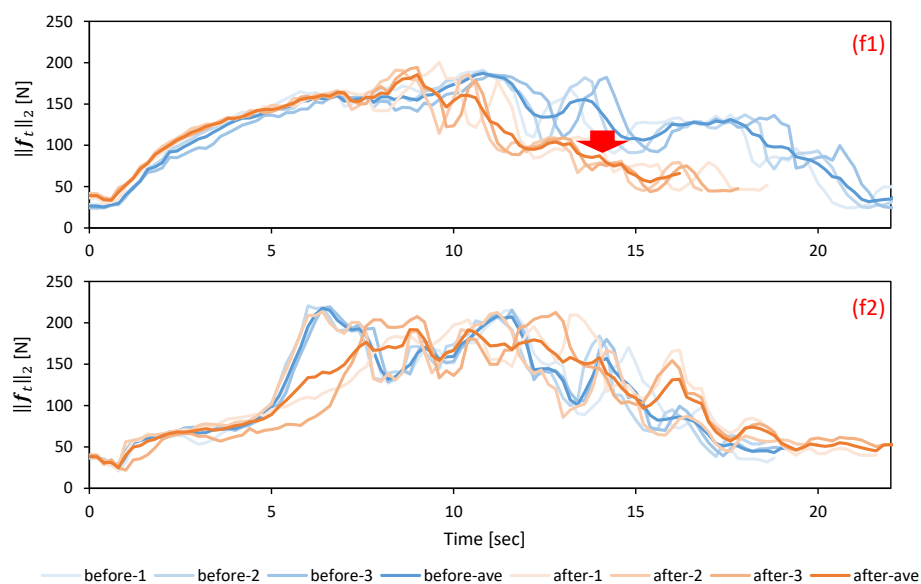


Fig. 6.62: Evaluation of box closing experiment by MusashiLarm [234]: transition of $\|f_t\|_2$ when using p before and after online update regarding (f1) and (f2).

る. 今後より複雑な動作を実行する際は, 学習時における Parametric Bias の入れ方, 動作データの分割等について, 考察していく必要がある.

6.5 動的筋骨格ハンド学習制御 - 関節角度-筋張力-筋長-接触力関係への適用

6.5.1 概要と先行研究

これまで、様々なロボットハンドが開発されてきた [242, 243, 244, 245, 116]. [242, 243] のような多自由度で剛なロボットハンドが多く存在する一方, [244, 245, 116] にあるような柔軟ハンドが, 適応的な把持や衝撃吸収の観点から注目を浴びている. [244] はゴムで形成された構造が空気圧により駆動されており, [245] は柔軟ブリーを使った, 高度な生物模倣型ハンドである. 本節では, それらに比べて多数のセンサを持ち, 指自体が切削ばねにより構成される柔軟かつ力強い把持が可能な筋骨格柔軟ハンド [116] を用いる. しかし, これら柔軟なハンド [244, 245, 116] は様々な利点の一方で, 把持による物体認識や接触検知, 動的制御等に現在もいくつかの問題点が存在する.

まず, モデリングが難しいため, 幾何モデルを直接扱うような制御が難しいという点である. この問題は現在は広く知られており, 学習型の把持物体認識や制御が盛んに研究されている. [246] は把持成功率の予測と強化学習による再把持を実現している. [247] は 5 指ハンドのシミュレーションにおいて, 様々なマニピュレーションタスクを実行可能な視覚運動方策を学習する模倣学習アプローチをとっている. これらの研究は剛なリンクを持つハンドで行われているが, 強化学習を用いていることもあり, 実機を長時間動かす, または柔軟身体をシミュレーションするという課題がクリアできれば柔軟ハンドにも適用可能である. 一方, 以下のように柔軟ハンドを使った学習型手法の例も存在する. [62] は劣駆動的なロボットハンドに対して接触センサを用いて強化学習を適用し, 二次元平面上で二指による In-hand マニピュレーションを実行している. [248] は空気圧駆動の劣駆動柔軟ハンドによる把持物体の分類を行っている. しかし, 2 次元平面上での制御であったり, 把持分類のみであったり, その適用先は限られている.

次に, 柔軟ハンドはモデル化が難しいだけでなく, そのモデルが常に変わっていく点である. ゴムなどの柔らかい素材を用いている場合 [244] は特に経年劣化が顕著に現れ, その他の素材の場合も劣悪な使用により劣化することが多い. また, 柔軟なハンドは関節角度センサ等を入れられないような場合が多く [244, 245, 116], ハンドの制御入力の初期化が難しい場合がある. 本節で用いる [116] のハンドも, 関節角度センサがないため, 人間が適当に指を伸ばした位置がゼロ点となる. このような場合, 初期化のずれによってモデルが変化してしまう. この問題に対して, 筋骨格ヒューマノイドではオンラインでセンサ間の写像を学習し続ける手法 [100, 124, 125] が提案されているが, 学習する次元が多くなるとオンライン学習は困難となり, 静的な動作でのみ成功している. そのため, ネットワーク全体

ではなく、その一部のみを変化させることでモデルを修正・適応させていくような方法が必要と考えられる。

最後に、これまで述べたように柔軟ハンドの把持物体認識・接触検知・動的制御等、それらのコンポーネントはそれぞれ別々に設計されてきている [246, 247, 62, 248]。それぞれ個別のネットワークを作る方法はそれぞれのタスクに特化することができるという利点がある一方、本節でターゲットとなる比較的低レイヤのコンポーネントにはたった一つの状態遷移を表すネットワークさえあれば事足りると考えた。これにより、あるネットワークの更新が他のネットワークに影響を与えることができ、それぞれのネットワークを管理する必要もない。そのため、一つのネットワークを学習し、そのネットワークのみを用いて様々なタスクができるようになることが望ましいと考える。

そこで本節では、Parametric Bias を含む再帰的ニューラルネットワーク [72] によって表されたセンサ値の状態方程式（本節では、Hand Dynamics Network, HADYNET）を構築し、これを用いて把持物体認識・シミュレーション・接触検知・把持安定化制御を総合的に行う (Fig. 6.63) [146]。Parametric Bias のみを更新することで現在の状態に適応し、経年劣化や初期化ずれ等にも対応する (Fig. 6.63 の PB Updater に対応する)。本手法を筋骨格ヒューマノイド Musashi [87] の左手 [116] に適用することで、その有効性を示す。これは、一般化多感覚相関モデルにおける動的身体図式のマスク変数を含まない一例である。6.1 節で得られた実行可能な m_{init} と m_{seq} をそれぞれ 1 種類のみとして、変数 m は入力から除いている。

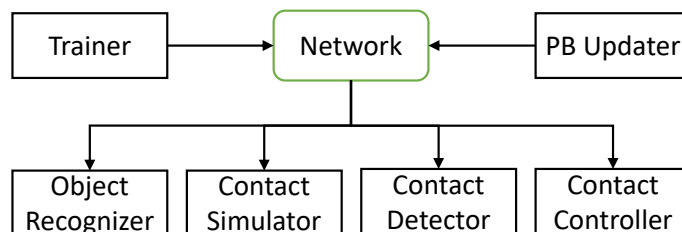


Fig. 6.63: Overview of the developed system [146].

6.5.2 筋骨格ハンド

本節で用いるハンドは筋骨格ヒューマノイド Musashi [87] に搭載された柔軟五指ハンド [116] である。Musashi [87] の前腕には、8 本の筋アクチュエータ [115] が存在し、手首に 3 本、指に 5 本が割り当てられている。指の 5 本の腱のうち 2 本はそれぞれ人差し指と中指、薬指と小指を駆動しており、2 つの指をプーリで分岐することで制御している。また、うち 2 本は親指を駆動している。最後の一本は

張力を高めることで切削ばねを圧縮し、指の剛性を変化させることができる。

それぞれの指の先端、手のひらには9つの接触センサとしてのロードセルが分布している。また、筋アクチュエータ [115] からは筋長、筋張力を測定することができる。このロードセル値を $f_{contact}$ 、エンコーダから得られる現在筋長を l 、筋張力値を f とする。加えて、手首には関節モジュール [87] があり、指の関節角度は測れないものの、手首の関節角度 θ は測定することができる。

l, f は 8 次元, $f_{contact}$ は 9 次元, θ は 2 次元である。また、一部で指に直接関係のある 4 つの筋のみを用いる場合は、これを f_{finger}, l_{finger} と表し、それぞれ 4 次元とする。

6.5.3 Hand Dynamics Network

まず、Hand Dynamics Network (HADYNET) のネットワーク構造、その学習、Parametric Bias の更新、これを用いた把持物体認識について説明する。その後、HADYNET を用いた接触シミュレーション、接触検知、接触制御について述べていく。全体のシステム図を Fig. 6.65 に示す。

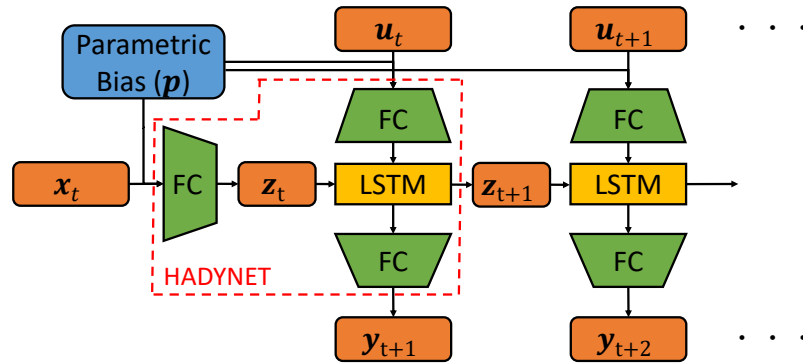


Fig. 6.64: Network structure of hand dynamics network (HADYNET) [146].

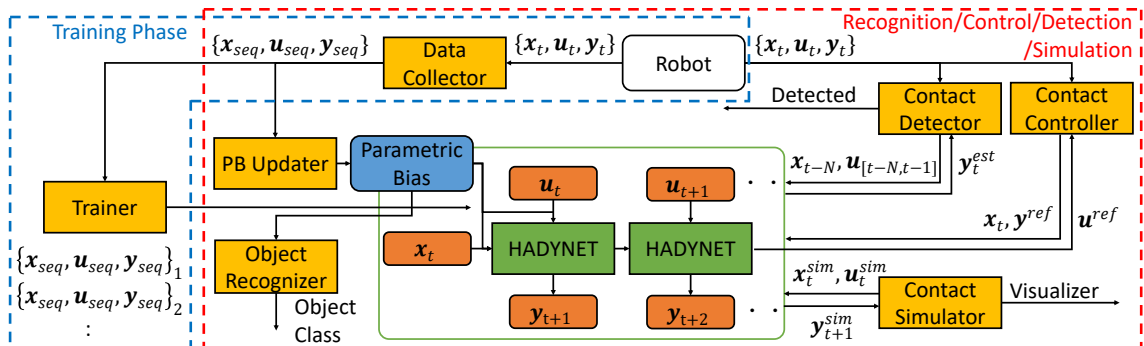


Fig. 6.65: The overall system using the hand dynamics network [146].

ネットワーク構造

HADYNET は、一般化多感覚相関モデルによる動的な身体図式において、 m_{init} と m_{seq} をそれぞれ一種類に制限したものと同一である。本手法では Parametric Bias に対して、把持物体や初期化、経年劣化等の様々な状態におけるアトラクタを埋め込む。

HADYNET を簡易的に関数 $h_{\{1,2\}}$ で表すと以下のようになる。

$$z_t = h_1(x_t, p) \quad (6.23)$$

$$y_{t+1} = h_2(z_t, u_t, p) \quad (6.24)$$

ここで x はハンドの初期状態、 z は隠れ状態、 u は制御入力、 y は観測状態、 p は Parametric Bias、 t は現在のタイムステップを指す。本節において、 $x^T = (l^T \ \Delta l^T \ f^T \ f_{contact}^T \ \theta^T \ \Delta \theta^T)$ ($\Delta\{l, \theta\}$ は前ステップからの差分、つまり速度を表し、 x は本節では 37 次元である) である。また、 $u = \Delta l^{ref}$ (l^{ref} は指令筋長を表し、 Δl^{ref} は本節では 8 次元である)、 $y^T = (l^T \ f^T \ f_{contact}^T \ \theta^T)$ (y は本節では 27 次元である) である。この Eq. 6.23, Eq. 6.24 を合わせたものを、本節では HADYNET と呼ぶこととする。つまり、HADYNET はハンドに備わる全センサの筋長速度指令に関する状態方程式を表す。後に説明するが、ネットワークの重みを W とすると、本手法における学習は W と p を、把持物体認識は p を、接触制御は u を、与えられた損失関数から誤差逆伝播法によって更新することに相当している。

具体的なネットワーク構造を Fig. 6.64 に示す。ここで、FC は Fully Connected Layer、LSTM は Long Short-Term Memory [241] を表す。LSTM のユニット数は 128、Parametric Bias は 8 次元とする。FC はそれぞれ入力と出力の値から次元数は一意に決定されており、Fully Connected Layer を通した後に、Batch Normalization [45]、ReLU [138] をかけている。また、特に記載がない場合は、LSTM における初期の cell state は 0 で初期化する。

本節ではタイムステップごとの間隔を 0.2 秒としている。

HADYNET の学習

訓練用データを蓄積し、HADYNET を学習させる。本手法では把持物体や初期化、経年劣化等の変化の様々な状態におけるアトラクタを包含する状態方程式を得るために、それぞれの状態を別のタスクとしてネットワークに学習させる。

まず、初期化・把持物体を変化させた、ある一つのハンド状態を用意し、これをタスク k とする。様々な動作を行い、その際の N ステップ分のデータ $D_k = \{x_{[0,N)}, u_{[0,N)}, y_{[0,N)}, k\}$ を取得する。本節では、2 種類のランダムな動作 (Random-1, Random-2) を行う。1 つめ (Random-1) は、指令関節角度に対応す

る指令筋長が得られるような幾何モデルを用意し、設定したランダムな指令関節角度を筋長に変換し、ランダムな秒数をかけて動かすことを繰り返す動作である。2 つめ (Random-2) は、指を曲げて物体を把持し、その状態で指に直接関係する \mathbf{l}_{finger} だけランダムに変化させる動作である (もし何も把持しないタスクの場合は適当な量指を曲げる)。後者は後に説明する把持安定化制御において重要となるデータであり、握り方の違いによる接触の変化を得ることができる。

次に、HADYNET で何ステップ先まで予測をさせるかを決める N_{train} を決定する。そして、 D_k から 1 ステップごとに連続する N_{train} ステップ分のデータを取り出し、このデータを $D'_k = \{\{\mathbf{x}_0, \mathbf{u}_{[0, N_{train}]}, \mathbf{y}_{[1, N_{train}+1]}, k\}, \{\mathbf{x}_1, \mathbf{u}_{[1, N_{train}+1]}, \mathbf{y}_{[2, N_{train}+2]}, k\}, \dots\}$ とする。なお、 \mathbf{x} は最初のステップのみ必要なため、はじめのタイムステップだけ取り出している。この試行を様々なハンド状態について行い、 $D'_{k=\{1,2,\dots,K\}}$ 用意する (K は行ったタスクの数を表す)。

最後に、このデータを用いて HADYNET を学習させる。得られたデータを全てシャッフルし、バッチ数を C_{batch}^{train} 、エポック数を C_{epoch}^{train} として学習させる。このとき、Parametric Bias \mathbf{p} の値は \mathbf{p}_k を用い、タスクごとに共通であるがタスク間では異なるパラメータとし、これを HADYNET と同時に暗黙的に学習させる。

本節では、 N はタスクごとに異なり、 $N_{train} = 10$, $C_{batch}^{train} = 100$, $C_{epoch}^{train} = 200$ として学習させている。

Parametric Bias の更新

HADYNET, Parametric Bias が学習されたが、実際に HADYNET を使用するときは、訓練時に学習された Parametric Bias は破棄する。改めてハンドに道具を持たせたり、初期化をしたりしたときに、もう一度 D' のデータを取得し (この状態におけるデータのみで良い)、HADYNET の重み \mathbf{W} は固定して、Parametric Bias のみ学習させる。全体ではなく、Parametric Bias という一部を学習させることによって、過学習せずに適切かつ素早くネットワークを現在の状態に適合させることができる。訓練時に、様々な物体把持、初期化、経年劣化等の状態について学習をしておけば、それらの状態が Parametric Bias に構築される。その Parametric Bias 内部を現在のデータから探索することで、現在のダイナミクスの状態を得ることができるのである。このときのバッチサイズを C_{batch}^{update} 、エポック数を C_{epoch}^{update} とし、更新則は MomentumSGD を用いる。

また、上記のように一度 D' を取得してから学習させるのが最も安定しているが、過学習を避けられるため、オンラインでの更新も可能である。オンライン学習を始めるデータ数 C_{thre}^{online} と、オンラインで学習が完了する程度の蓄積する最大データ数 C_{max}^{online} を決める。データを上記と同じように貯めていき、 C_{thre}^{online} を超えたらバッチサイズを C_{batch}^{online} 、エポック数を C_{epoch}^{online} として学習し始める。データ

が C_{max}^{online} を超えたら、一番古いデータを削除することを繰り返す.

本節では, $C_{batch}^{\{update, online\}} = N$ (N は D' のデータ数とする), $C_{epoch}^{update} = 20$, $C_{epoch}^{online} = 3$, $C_{thre}^{online} = 100$, $C_{max}^{online} = 200$ とする.

把持物体認識

Parametric Bias の更新時において得られた \mathbf{p} を用いることで, 把持物体の認識を行うことができる. 訓練時に用いた K 個のタスクに対して, 学習された Parametric Bias \mathbf{p}_k と, それぞれのタスクにおいて把持している物体の名前を保存しておく. \mathbf{p}_k の値を主成分分析により 2 次元に落とし, 2 次元平面にプロットしておく. Parametric Bias の更新時において学習された \mathbf{p} に, 先の PCA で得られた変換をかませ, 同様に 2 次元平面にプロットする. これにより, 現在の \mathbf{p} の位置から, 現在把持している物体がどの物体に近いかを可視化することができる. また, 最近傍法により, 最も \mathbf{p} に近い \mathbf{p}_k を現在把持している物体として, 認識することが可能である.

接触シミュレーション

接触のシミュレーションは HADYNET の順伝播のみで行うことができる. 初期状態 \mathbf{x}_t^{sim} を設定し, コマンド \mathbf{u}_t^{sim} を指令することで, \mathbf{y}_{t+1}^{sim} を得ることができる. \mathbf{f} は筋の色で, $\mathbf{f}_{contact}$ は力の矢印の大きさで, θ は関節角度で描画することで, ハンドの動作指令における力や位置のシミュレーションを行うことができる. また, Parametric Bias を変更することで, 物体の有無や種類によるハンドのダイナミクスの変化を観察し, 強化学習等にも用いることができると考える.

接触検知

HADYNET における予測誤差から, ハンドにおける接触検知 (異常検知) を行うことができる. まず, 接触検知を行う前に Parametric Bias の更新を行う. そのあと, HADYNET で予測するタイムステップ数 N_{detect} ($N_{detect} \leq N_{train}$) を決め, データ D' に対して, N_{detect} ステップ後における予測誤差の平均と共分散行列を計算しておく. つまり, ある一つデータの終点を i として, $\mathbf{x}_{i-N_{detect}}$ と $\mathbf{u}_{[i-N_{detect}, i]}$ を取り出し, それを用いて HADYNET により予測された \mathbf{y}_i^{est} とデータセット内の \mathbf{y}_i の誤差を計算し, 全データについてこの平均 μ と共分散行列 Σ を計算するというのである.

次に, 実際の接触検知について説明する. 現在のタイムステップ t から N_{detect} 前までの \mathbf{x} と \mathbf{u} を常に保存しておく. ここで, 現在 t において, $\mathbf{x}_{t-N_{detect}}$ と $\mathbf{u}_{[t-N_{detect}, t]}$ を取り出し, HADYNET により,

\mathbf{y}_t^{est} を予測する. ここで, \mathbf{y}_t を得て, 以下のマハラノビス距離を計算する.

$$d = \sqrt{(\mathbf{y}_t - \mu)^T \Sigma (\mathbf{y}_t^{est} - \mu)} \quad (6.25)$$

この d が閾値 C_{detect} を超えたとき, 予想とは違う動きが発生, つまり接触が検知されたと見なす. この C_{detect} はデータ D' における d の分散の 3σ 等を用いることも可能だが, 余計な接触が検知されないよう, 本節では高めの値を設定している.

本節では $C_{detect} = 100$ とする.

接触制御

HADYNET による予測を指令値に近づけるように制御指令を最適化することで, 指令した接触状態を実現することができる.

まず, HADYNET で予測するタイムステップ $N_{control}$ ($N_{control} \leq N_{train}$), 接触状態の指令値 \mathbf{y}^{ref} を決める. 次に, 最適化前の制御指令値 $\mathbf{u}_{[t, t+N_{control})}^{opt}$ (以降では \mathbf{u}_{seq}^{opt} とする) を決定する. 現在のタイムステップ t における \mathbf{x}_t を取得し, \mathbf{u}_{seq}^{opt} とともに HADYNET に入力することで, 予測値 $\mathbf{y}_{[t+1, t+N_{control}+1)}^{est}$ を得る. そして, 損失関数 h_{loss} により損失 L を計算し, 以下のように \mathbf{u}^{opt} を最適化していく.

$$L = h_{loss}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) \quad (6.26)$$

$$\mathbf{g} = \partial L / \partial \mathbf{u}_{seq}^{opt} \quad (6.27)$$

$$\mathbf{u}_{seq}^{opt} \leftarrow \mathbf{u}_{seq}^{opt} - \alpha \mathbf{g} / \|\mathbf{g}\|_2 \quad (6.28)$$

ここで, \mathbf{y}_{seq}^{est} は $\mathbf{y}_{[t+1, t+N_{control}+1)}^{est}$ を, \mathbf{y}_{seq}^{ref} は \mathbf{y}^{ref} を縦に $N_{control}$ 個並べたベクトル, α は更新率を表す. 実際には, $C_{batch}^{control}$ 種類の α を使って \mathbf{u}_{seq}^{opt} を更新し, そのバッチの中で最も L が小さかったものを採用し, また勾配を計算することを $C_{epoch}^{control}$ 回繰り返す. また, $\mathbf{u}_{control}^{seq}$ は前ステップで最適化された $\mathbf{u}_{[t-1, t+N_{control}-1)}^{opt}$ を一つ左ヘシフトし, 最終項を複製した $\mathbf{u}_{\{t, \dots, t+N_{control}-1, t+N_{control}-1\}}^{opt}$ を用いる. 最適化後, \mathbf{u}_t^{seq} が実機に送られる.

ここで, 損失関数 h_{loss} の設計について考える. 本節では, 主に把持安定化について考え, 道具を把持したときの初期の接触を, 常に保ち続けるような制御を行うことを目的とする. つまり, \mathbf{y}^{ref} は道具を把持した際の初期の値 \mathbf{y}^{init} となる. 例えば, ハンマーで打撃動作を行うと, 徐々に接触状態が変化し, 持ち方が変わってしまうことがあるが, これを抑制するような制御である. この場合, 損失関数は以下

のように設計する.

$$h_{loss}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) = h_{loss,1}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) + h_{loss,2}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) \quad (6.29)$$

$$h_{loss,1}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) = \|\mathbf{w}_1(\mathbf{f}_{all,seq}^{est} - \mathbf{f}_{all,seq}^{ref})\|_2 \quad (6.30)$$

$$w_1[i] = \begin{cases} 1.0 & (\mathbf{f}_{all,seq}^{est}[i] \geq \mathbf{f}_{all,seq}^{ref}[i]) \\ \beta & (otherwise) \end{cases}$$

$$h_{loss,2}(\mathbf{y}_{seq}^{est}, \mathbf{y}_{seq}^{ref}) = w_2\|\boldsymbol{\theta}_{seq}^{est} - \boldsymbol{\theta}_{seq}^{ref}\|_2 + w_3\|\mathbf{l}_{seq}^{est} - \mathbf{l}_{seq}^{ref}\|_2 \quad (6.31)$$

ここで, $\mathbf{f}_{all,seq}^{\{est,ref\}}$ は $\mathbf{y}_{seq}^{\{est,ref\}}$ から \mathbf{f} と $\mathbf{f}_{contact}$ を抜き出して縦に並べたもの, $\{\boldsymbol{\theta}, \mathbf{l}\}_{seq}^{\{est,ref\}}$ は同様に \mathbf{y} から $\{\boldsymbol{\theta}, \mathbf{l}\}$ を抜き出したものを表す. また, $w_{\{2,3\}}$ は重み, w_1 は重みのベクトル, $w_1[i]$ はその i 番目の要素, β は定数を表す. この w_1 の設計は, 接触センサの性質を考慮したものである. 接触センサや筋張力センサの値は, マイナス方向には 0 までしか変化しないが, プラス方向には定格まで大きく変化する. つまり, $\beta = 1.0$ のとき, 最適化の結果として, 接触がマイナス方向に変化しやすくなってしまい, 接触が維持できない. そこで, β を 1.0 以上に設定することで, 接触を常に保ち続けるような制御指令を生成する. また同時に, 関節角度, 筋長に対しても制限をかけることで, 接触を維持しつつも, 筋長・関節角度が大きく変化しないような制御指令を得ることができる. ただ, 初期状態を保つような把持安定化ではなく, 狙った接触力を出したいような場合においては, $\beta = 1.0$ が好ましい.

本節では, $N_{control} = 8$, $C_{batch}^{control} = 4$, $C_{epoch}^{control} = 3$, $\beta = 3.0$, $w_2 = 1.0$, $w_3 = 1.0$ とする.

6.5.4 実験



Fig. 6.66: Grasped objects in this study: Hammer, Hammer-S (hammer with soft cover), Cylinder, Gripper, and Ball [146].

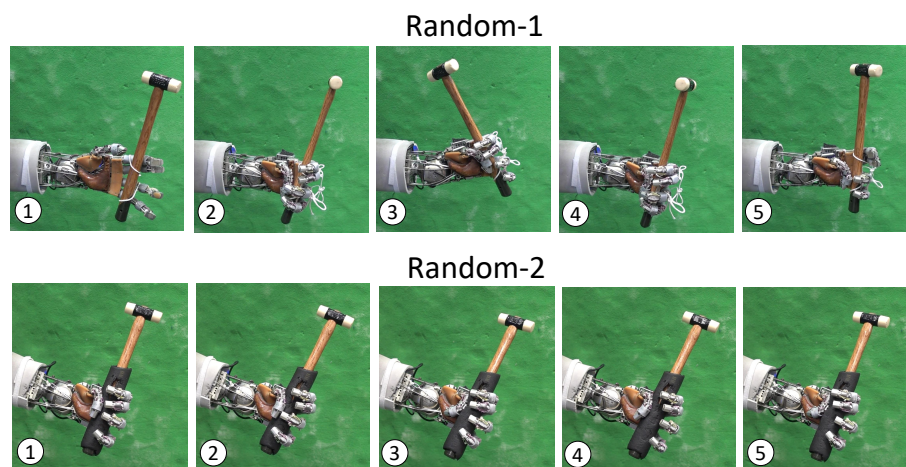


Fig. 6.67: Motion sequences of the data collection phase of Random-1 and Random-2 [146].

HADYNET 学習実験

本節で用いる把持物体は, Fig. 6.66 に示すように, Hammer, Hammer-S (hammer with soft cover), Cylinder, Gripper, and Ball の 5 つを用いる. また, 何も把持しない状態として None を加え, 全部で 6 つの把持物体について扱う.

これら 6 つの把持物体を持たせ, Random-1, Random-2 の動作をそれぞれ約 500 ステップ (100 秒) 行う. その様子を Fig. 6.67 に示す. Random-1 では把持物体が手から落ちないように紐やテープで軽く括りつけているが, Random-2 では把持状態から大きく動作しないため必要ない. これを, ハンドのセットアップをやり直しながら 3 回行い, 一つの把持物体で計 6 回, 全把持物体で 36 回の試行を行った. 集まったデータは計約 18000 個となり, これを用いて HADYNET を学習させた. Parametric Bias を固定して学習させた場合と, タスクごとに別の値を暗黙的に学習させた場合 (This Study) の loss の遷移の比較を Fig. 6.68 に示す. 前者と後者には loss の下がり方に差があり, 後者の方が約 20% loss が小さいことがわかる. ただし, 何も掴んでいない瞬間の None と同じデータがそれぞれの把持物体について多く含まれるため, そこまで大きな変化ではない. つまり, Parametric Bias を入れることで, 把持物体の違いや初期化の違いを暗黙的に学習させることができる. 本実験では初期化・把持物体という 2 つの違いにのみ対応しているが, これを時間間隔を空けて繰り返すことで, 経年劣化等にも対応できるようになる. 経年劣化は主に筋の伸びやバネの癖がつき初期点が変わってしまう等の影響を持つが, これらは十分に初期化の違いにも現れうる違いであるため, 初期化の違いを扱うことができれば, 経年劣化も扱うことができると考える. また, PB の違いが経年劣化と初期化どちらによる違いかを区別することは難しく, 本節では直接的には経年劣化に関する実験は行わない.

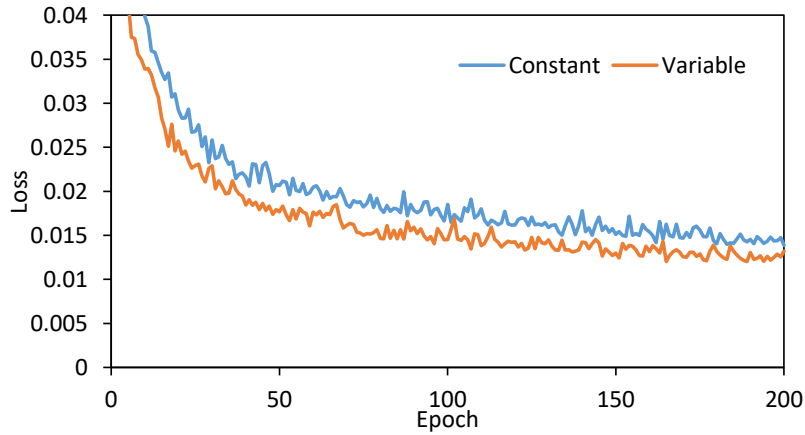


Fig. 6.68: Comparison of the training results of HADYNET between with fixed parametric bias (Constant) and with variable parametric bias (Variable, this study) [146].

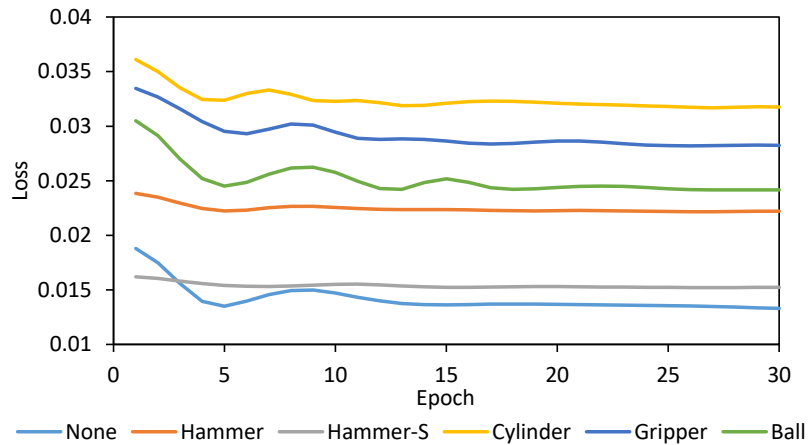


Fig. 6.69: Transition of loss when updating only parametric bias [146].

把持物体認識実験

ハンドの初期化を行い新しく6つの把持物体のデータを得た。これを用いて Parametric Bias のみの更新を行った際の loss の変化を Fig. 6.69 に示す。全ての把持物体について、Parametric Bias のみを更新することで loss が6–28%下がっていることがわかる。つまり、Parametric Bias を変更することで、様々な把持物体・初期化を再現できている。

訓練時に得られた Parametric Bias と、新しいデータによって得られた Parametric Bias を、6.5.3 節で説明したように2次元平面上にプロットする (Fig. 6.70)。それぞれの把持物体の Parametric Bias が、正しくグルーピングされていることがわかる。四角で囲まれたデータが新しいデータであるが、それぞ

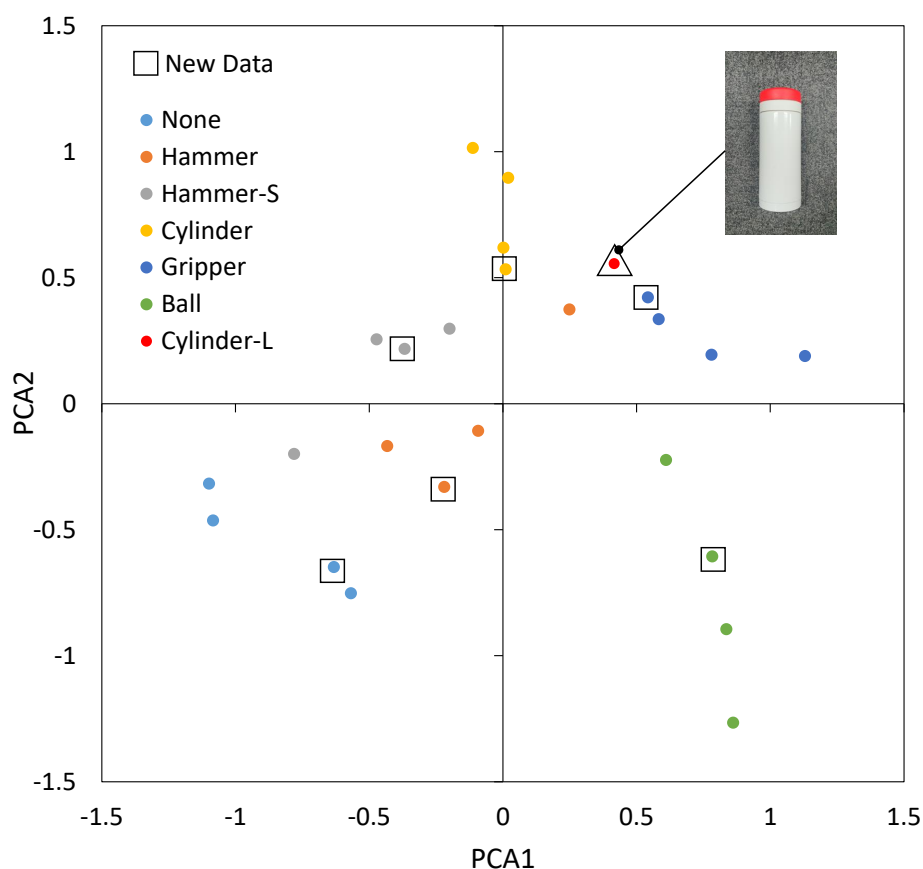


Fig. 6.70: Results of the object recognition [146]. The marks surrounded by squares are new data to recognize what the hand is grasping. The mark surrounded by a triangle is a new object, the large cylinder (Cylinder-L).

れのデータについて最近傍法によって把持物体を正しく識別できている。つまり、対象物体を把持し、それをランダムに握ったり離したりすることで、その物体が何であるかを、視覚を使わずに判断することができる。

また、新しい物体 Cylinder-L に関してもデータを取り、Parametric Bias を学習させた。ここで、Cylinder の直径は 32 mm, Cylinder-L の直径は 68 mm である。また、学習に使用した物体で最も把持部分の大きい Gripper は、把持部分の幅が 60 mm である。Fig. 6.70 に獲得された Cylinder-L の Parametric Bias の値 (Triangle に囲まれている) を示すが、これは Cylinder と Gripper のクラスの間が存在している。Cylinder の円筒という形の特徴と Gripper の大きさの特徴を併せ持った Cylinder-L の位置としては妥当である。つまり、このように新しい物体に関しても、学習の際のデータを参照することで、物体の特徴を掴むことができる。

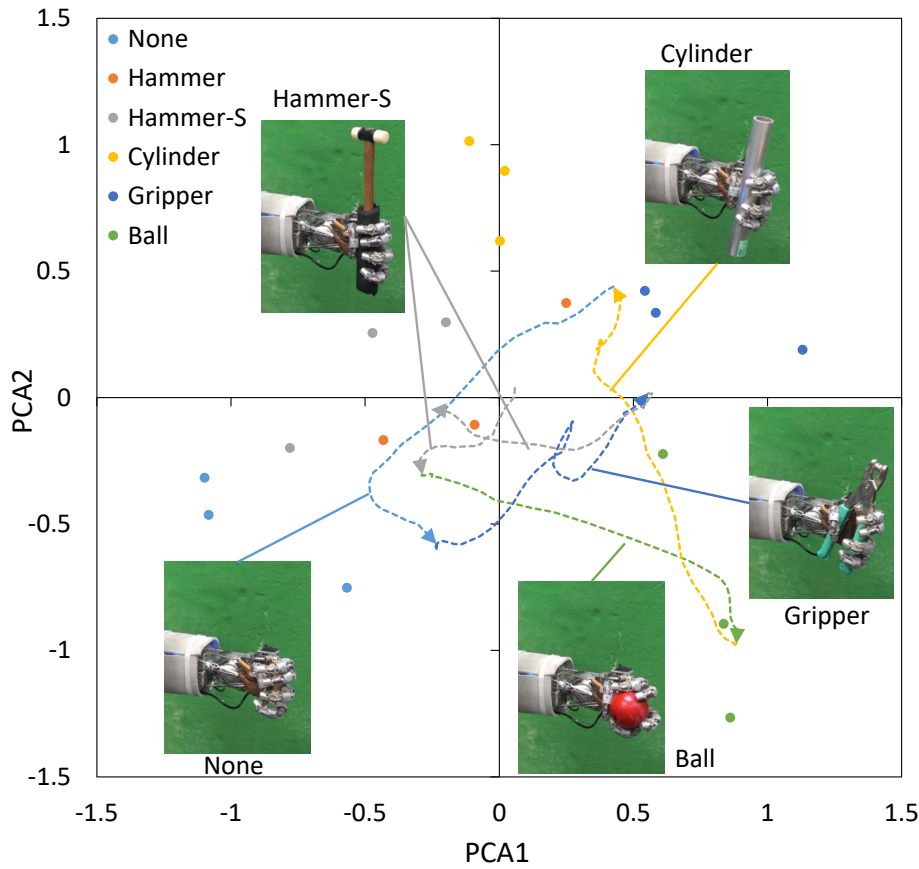


Fig. 6.71: Online update experiment of parametric bias [146]. The graph shows the transition of parametric bias when updating it online.

最後に、6.5.3 節で説明した、Parametric Bias をオンラインで更新することを試みる。Hammer-S, Ball, Cylinder, None, Gripper, Hammer-S の順で物体を約 1 分半ずつ握って 6.5.3 節の Random-2 を実行する。この間常にオンライン更新を実行しておき、その際の Parametric Bias を 2 次元平面上に投影したときの遷移を Fig. 6.71 に示す。矢印の方向に Parametric Bias は移動しており、学習時に得られた Parametric Bias のプロットの色と遷移の線の色は一致している。図から、Fig. 6.70 のように一度に学習させる場合に比べると精度は落ちるが、現在の Parametric Bias が、今把持している物体のクラスの方へ動いていることがわかる。特に、Ball は Parametric Bias が正確に遷移しているのに対して、その他は、現在把持物体のクラスの近傍までは遷移するものの、正確に最後まで遷移することはできていない。これは、最大で N_{max}^{online} までのデータしか用いず、常にデータが変わっていくため、loss が下がりがきいていないためだと思われる。しかし、このようにオンラインで Parametric Bias を更新することで、徐々に把持物体の傾向を掴むことができることがわかった。

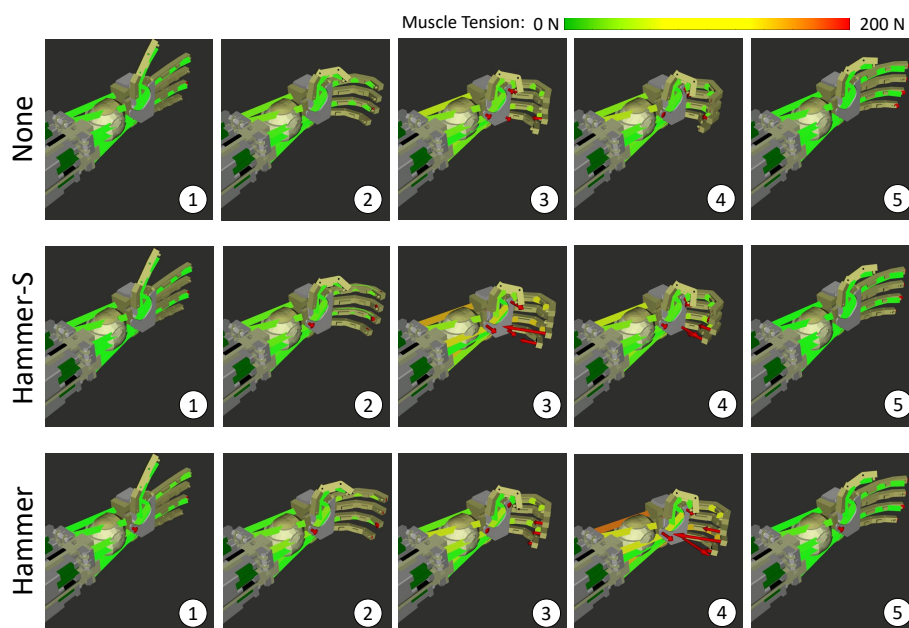


Fig. 6.72: Contact simulation using parametric biases of None Hammer-S, and Hammer [146].

接触シミュレーション実験

HADYNET を使ったシミュレーションの様子を, None, Hammer-S, Hammer について Fig. 6.72 に示す. それぞれにおいて, 訓練時に得られた Parametric Bias をセットしている. ここでは, 矢印が大きいほど接触力が大きく, 筋の色が緑から赤くなるほど筋張力が高いことを表している. None の場合は接触力・筋張力ともに大きな変化は現れていない. それに対して, Hammer-S の場合は指先・手のひらのロードセルに大きな力が加わり, 筋張力も高まっていることがわかる. Hammer-S と Hammer を比べると, Hammer-S は soft cover によって径が Hammer よりも大きくなっているため, 把持の早い段階で接触力・筋張力の高まりが見られる. Hammer は soft cover がないため把持の際の衝撃が吸収できず, 接触が生じた瞬間に突然 Hammer-S よりも大きな力が出ていることもわかる. このように, Parametric Bias の変化のみでハンドのダイナミクスを変化させることが可能となった. これは, モデリング困難な柔軟ハンドのシミュレーション上での強化学習に用いることができると考える.

接触検知実験

まず, 様々な物体を握らせた時に, 接触検知の値 d がどのような挙動をするのかを観察する. 現在の初期化状態で一度 Parametric Bias を None に適合させ, None, Cylinder, Hammer-S, Ball を持たせたと

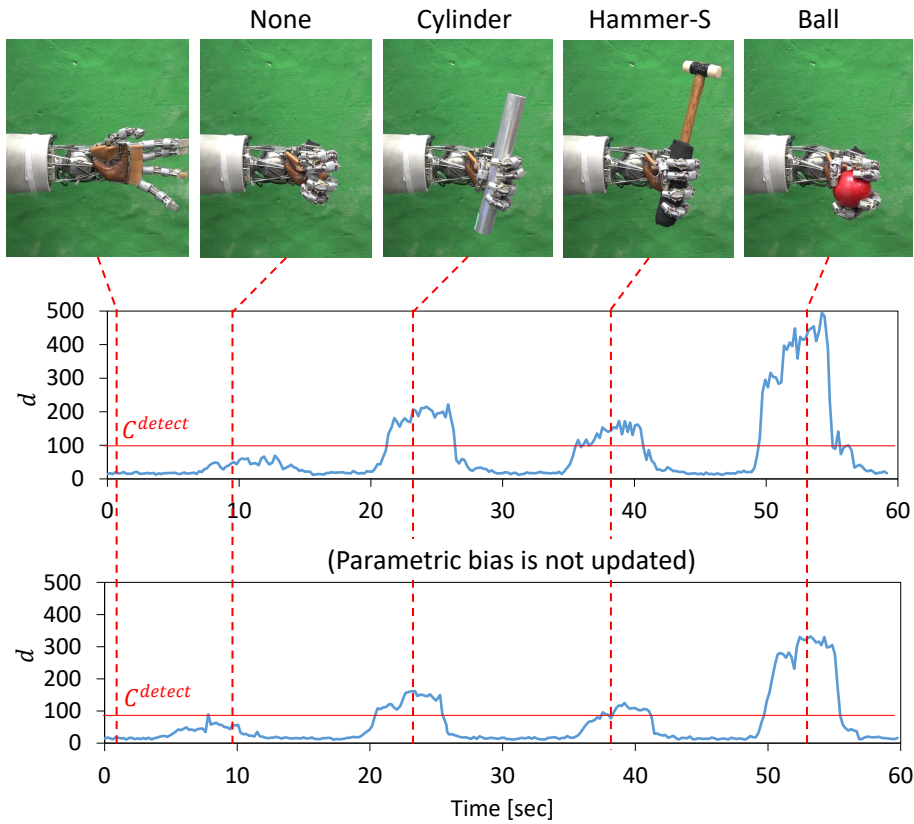


Fig. 6.73: Contact detection experiment when grasping various objects [146]. The middle graph shows d when updating parametric bias and the lower shows d when using parametric bias trained previously.

きの d の遷移を Fig. 6.73 の中図に示す. 本節では $C^{detect} = 100$ であり, None は検知されず, Cylinder, Hammer-S, Ball を握ったときに正しく接触が検知出来ていることがわかる. Fig. 6.70 と照らし合わせると, None から値が遠い Ball や Cylinder の方が, None から近い Hammer-S に比べて d が大きな値となっていることがわかる. これに対して, 学習時に得られた None の Parametric Bias を用いて接触検知を行った結果を Fig. 6.73 の下図に示す. 物体を把持したときの d にメリハリがなくなり, None のときは 89 という C^{detect} に近い値が, Hammer-S のときも最大で 114 という C^{detect} に近い値が計測された. これでは多少の誤差で接触が検知されたりされなかったりという現象が起きてしまう. つまり, Parametric Bias は初期化の状態を含み, もしハンドを初期化し直したなら, 把持物体が同じであっても一度 Parametric Bias を更新することが望ましい.

次に, 接触を検知するだけでなく, 接触を適切に保つような動作も可能である. 接触を保つ, つまり d が C^{detect} になるように制御しつつ, 布巾で掃除をするタスクを行った. 具体的には, 接触が検知されたら接触面から離れる方向に, 検知されなければ接触面に押し付ける方向に逆運動学を解いて少し

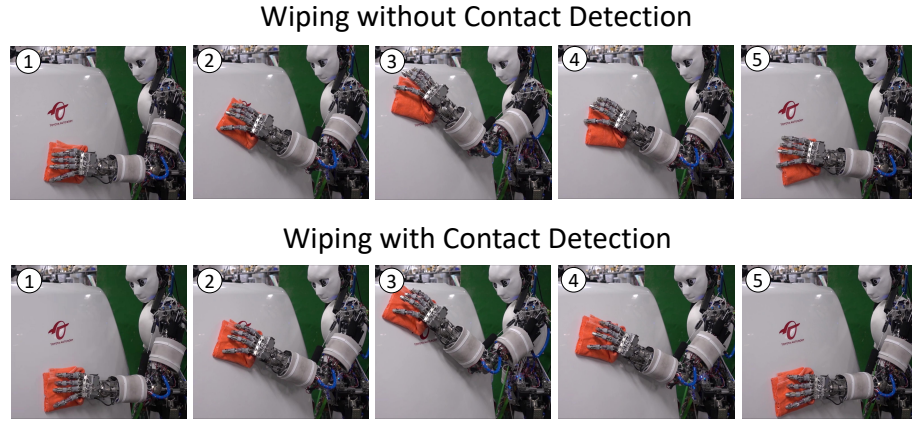


Fig. 6.74: Wiping experiments with and without contact detection [146].

ずつ動かしていく。Fig. 6.74 に接触保持を入れない場合と入れた場合の掃除の様子を、Fig. 6.75 にその際の d の遷移を示す。Fig. 6.74 では、接触保持を入れない場合は布巾が手からずれ落ちてしまっているのに対して、接触保持を入れた場合はしっかりと面をなぞれていることがわかる。Fig. 6.75 の左図に示すように、この掃除する面は上に行くほど離れるような斜めの構造になっている。つまり、接触保持を入れない場合は手が直上に動き、徐々に接触面から手が離れて布巾がずれてしまっているのである。その際の d は、一様でほとんど変化がないことがわかる。これに対して、接触保持を入れた場合は手を直上に動かすだけでなく、徐々に接触面方向にも動かされるため、面に沿って動作ができている。手を上に動かすときは接触面の傾きと接触保持による動作がほとんど同じなため d に大きな変化は見られないが、手を下に動かす際には逆に接触力が上昇する方向のため、 d が上昇し、 C^{detect} 付近で変化している。よって、接触検知を能動かつ受動的に活用することができる。

接触制御実験

6.5.3 節で説明した把持安定化戦略を、ハンマー操作の際に適用し、接触の安定化が可能かどうかを評価する。まず、使用する Hammer-S に関して D' を取得して Parametric Bias を更新する。次に、把持安定化戦略を使用した場合と使用しない場合について、Fig. 6.76 の上図のようにハンマーで机を叩く動作を 30 秒間行い、その際の評価値 E の遷移を確認する。 \mathbf{y}^{ref} は最初に Hammer-S を握ったときの値 \mathbf{y}^{init} とする。本節では道具把持初期の接触を常に保つような安定化を行うため、 E は $h_{loss,1}$ を用い、これは小さいほうが把持が安定していると言える。実験結果を Fig. 6.76 の中図・下図に示す。把持安定化戦略を用いない場合は、徐々に E が大きくなり、初期の把持が崩れてしまっていることがわかる。これに対して、把持安定化戦略を用いたときは、試行を重ねても E は 0.3 付近から大きく変化せ

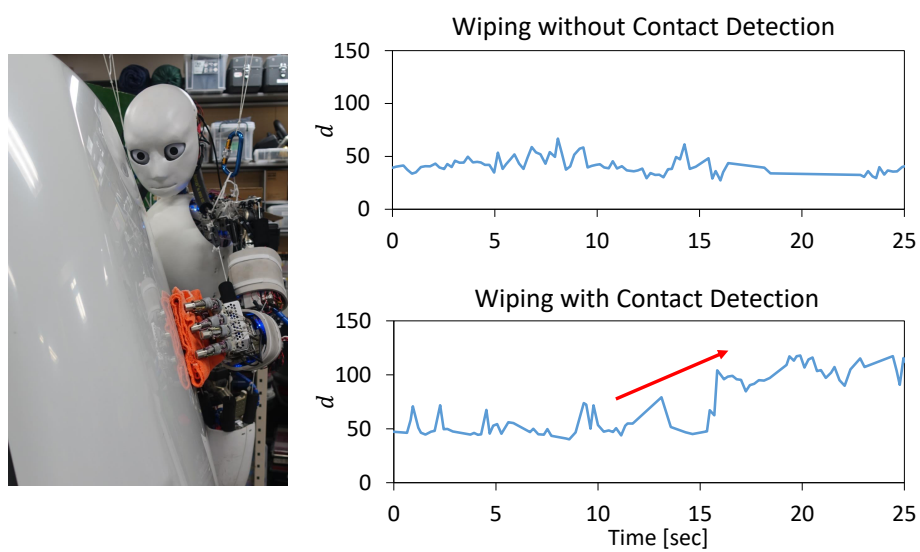


Fig. 6.75: The transitions of d when wiping with and without contact detection [146]. The left figure shows the inclination of the target surface to be wiped.

ず, 大きな値になっても, すぐに元に戻っている. つまり, HADYNET を用いることで, 指令した接触状態を実現するような制御が実行可能である.

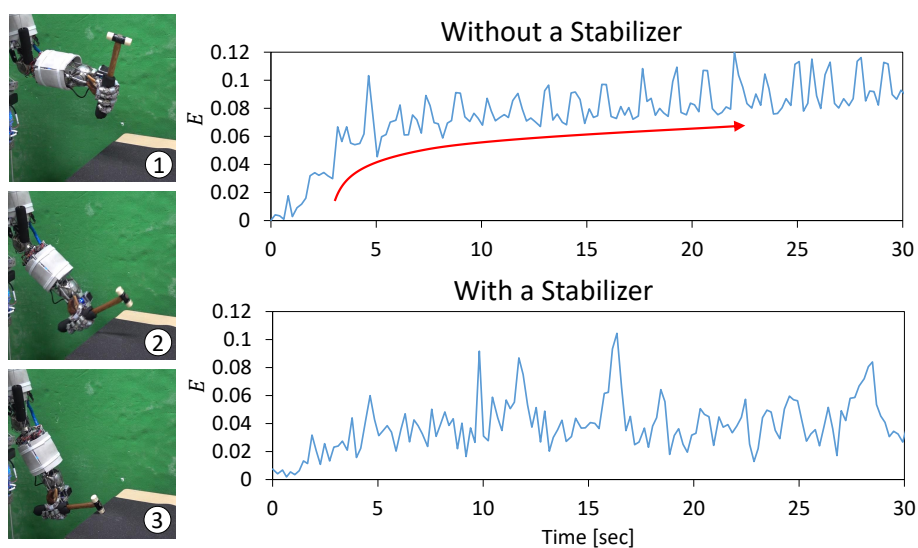


Fig. 6.76: Hammer hitting experiments with and without the grasping stabilizer [146]. The graph shows the transition of the evaluation value E .

6.6 本章のまとめ

まず、得られたデータから一般化多感覚相関モデルの入出力を自動で決定する実験を行った。得られたネットワーク構成とマスク変数の集合が合理的であることを示し、本手法の有効性を確認した。

次に、静的身体図式学習の例として、身体の制御入力から道具先端位置を推定するネットワークを構成し、誤差逆伝播を用いて道具先端位置を制御する手法を開発した。Parametric Bias を含めることで、その中に把持状態に関する暗黙的な変数を埋め込み、これをオンライン学習することで、逐次的な把持状態変化に適応することができる。また、線形変換ではなくニューラルネットワークにより表現することで弾性変形するような道具・軸駆動や筋骨格などの異なる身体も同様に扱うことができることを確認した。

次に、静的身体図式学習の例として、筋骨格ヒューマノイドにおける状態推定・制御・シミュレーションを統一的に扱う Musculoskeletal AutoEncoder について提案した。筋骨格ヒューマノイドに一般的に備わる関節角度・筋長・筋張力センサの関係性を一般化多感覚相関モデルで記述し、これを実機センサ情報からオンラインで更新していく。関節角度の状態推定は、現在の筋長・筋張力を本ネットワークに通すことで行うことができる。関節の位置制御は、指令関節角度と筋張力最小化の観点から誤差逆伝播を通して指令筋長を求め、実行することができる。関節角度と筋張力のシミュレーションは、指令筋長と必要関節トルクの観点から誤差逆伝播を通してリアルタイムに行うことができる。また、Musculoskeletal AutoEncoder の微分により筋長やコピアンを求めることで、関節トルク制御、可変剛性制御も同様に実行することができる。本手法により、筋骨格ヒューマノイドの静的な動作に関する必須の要素群を統合し、かつ実機情報をオンラインでモデルに取り込んでいくことが可能となった。

次に、動的身体図式学習の例として、学習したタスクを模倣する以外に追加の動作スタイル制約を考慮可能な模倣学習手法の提案を行った。人間の教示による動作のばらつきを Parametric Bias に埋め込んだ。予測状態や制御変数に関する追加制約の損失関数を定義し、この Parametric Bias をオフラインまたはオンラインで更新することで、追加制約を考慮することが可能となった。本研究を柔軟で冗長かつ多数のセンサを持つ筋骨格ロボットや PR2 に適用し、筋張力最小化や筋長・関節速度変化等を行うことに成功した。

最後に、動的身体図式学習の例として、柔軟ハンドの把持物体や初期化のずれ、経年劣化等のダイナミクスの変化に対応した、把持物体認識・接触シミュレーション・接触検知・接触制御が可能な HADYNET を開発した。Parametric Bias を用いた動的身体図式学習により、様々な状況におけるダイナミクスの違いを暗黙的に学習させることが可能となる。また、Parametric Bias の違いによる把持物

体の認識, ネットワークのフォワーディングによる接触シミュレーションが可能である. 予測誤差を用いた接触検知, 予測結果を指令値に近づけるような制御指令値を誤差逆伝播法により求める接触制御法を開発した. 様々なコンポーネントが一つのネットワークで構築可能であり, そのネットワークの一部を更新するのみで様々なダイナミクスを表現できることを確認した.

第7章

身体図式学習の応用実験

本章の概要を Fig. 6.1 に示す。また、身体図式学習の応用実験範囲に関する詳細図を拡大したものを Fig. 7.2 に示す。本章では、一般化多感覚相関モデルの応用として、(a)–(g) の要素を考える。これらは、第 6 章で網羅できていない要素に関する身体図式の応用である。まず、(a) と (b) はネットワーク入出力である x の減少と増加について扱う。これは、柔軟性と冗長性の利点を活かすための身体図式の応用である。(a) は 7.1 節の筋破断を考慮した適応的身体図式学習において、動作実行中におけるセンサ・アクチュエータの損傷による欠落を異常検知し、 x に対する損傷状態を表すマスク変数 r の導入に基づく制御や状態推定の変化を扱う。(b) は 7.2 節の筋追加を考慮した適応的身体図式学習において、 x を追加した場合におけるネットワークの再利用に向け、重みコピーと変化前ネットワークを併用した再学習を行う。次に、(c) と (d) は x に関する確率表現と平均分散表現の導入について扱う。これは、空間的柔軟性の欠点を補完するための身体図式の応用である。(c) は 7.3 節の筋骨格構造における危険回避学習において、ネットワーク出力に確立表現を導入することで、確立の最小化に基づく動作修正や危険回避行動を実現する。(d) は 7.4 節の環境適応型台車制御学習において、ネットワーク出力に平均分散表現を加え最尤推定に基づく損失関数により学習することで、分散最小化による安定な制御が実現可能であることを示す。また、(e) は一般化多感覚相関モデルの自動分割を扱う。7.5 節の筋

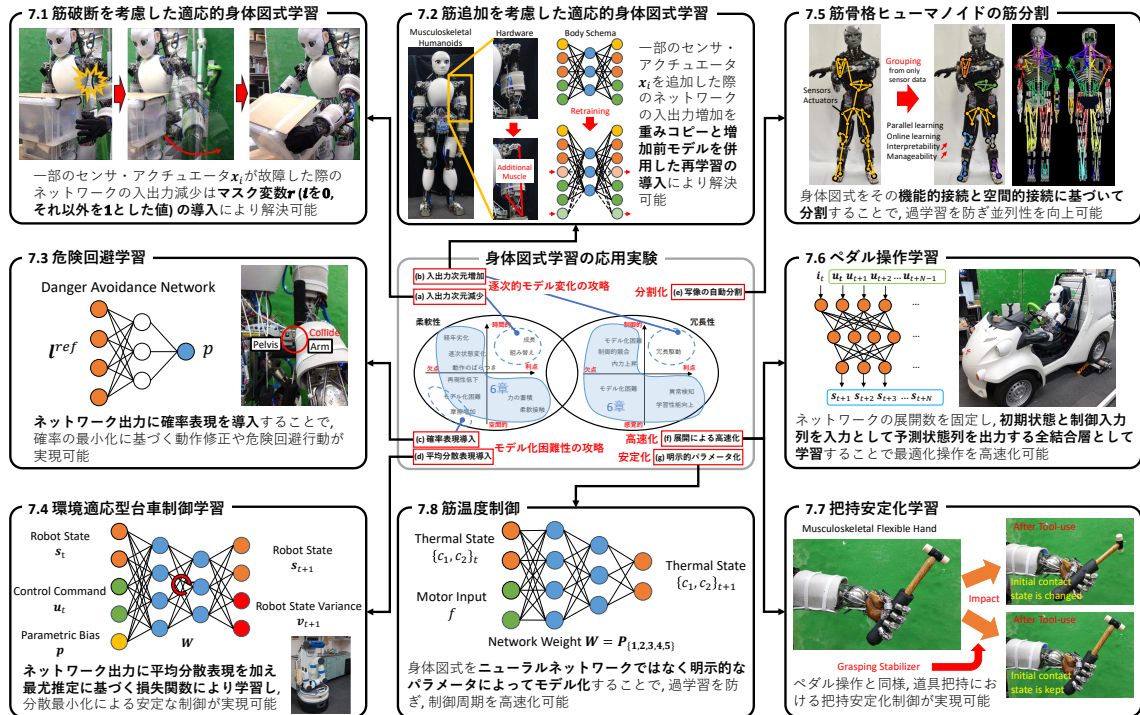


Fig. 7.1: The overview of this chapter.

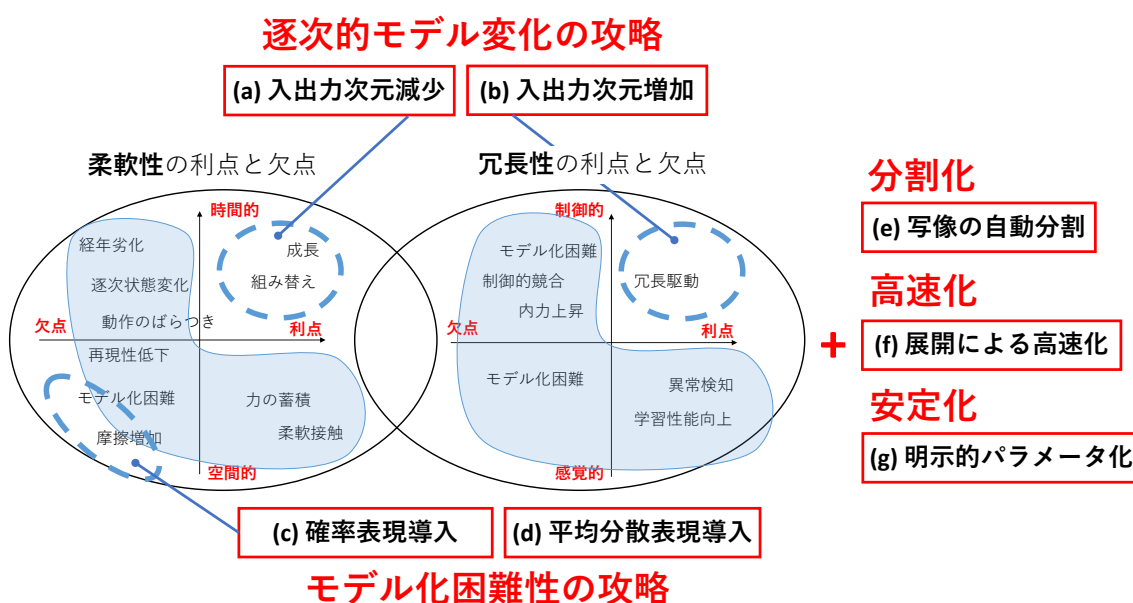


Fig. 7.2: The overview of the applications of generalized multisensory correlational model.

骨格ヒューマノイドにおける筋分割では、強い機能的接続と空間的接続を持つセンサ・アクチュエータをまとめ、それぞれのグループについて一般化多感覚相関モデルを適用することで、解釈性・並列実行性を高める。(f)は動的な一般化多感覚相関モデルの時系列方向展開による高速制御計算を行う。7.6節のペダル操作学習と7.7節の把持安定化学習では、ネットワークの展開数を固定し、初期状態と制御入力列を入力として予測状態列を出力する、全結合層として身体図式を学習させることで、より安定して高速に制御入力を計算可能になることを示す。(g)は明示的パラメータを用いた身体図式学習の安定化について扱う。7.8節の筋温度制御では、身体図式をニューラルネットワークではなく明示的なパラメータによってモデル化することで、過学習を防ぎ、制御周期を高速化可能とする。これらの実験から、身体図式学習によりロボットの身体・道具・動作環境における様々な相関複雑性と時間的変容に対応することが可能であることを示す。

7.1 身体図式ネットワーク入出力の減少 - 筋破断を考慮した適応的身体図式学習

7.1.1 概要と先行研究

本節では、筋骨格ヒューマノイドの特性である、筋の冗長性に着目した制御戦略について述べる。制御対象である関節角度に対して制御入力(筋長)が冗長である筋骨格構造においては、筋が一本切れた

としても、それを補償してこれまでと同じように動作し続けることが可能であり、よりロバストなロボット構成へと繋がる (Fig. 7.3). この戦略を、柔軟でモデル化が困難な筋骨格ヒューマノイドに適用するため、本研究では学習型制御を用いた新しい枠組みを提案する. 筋破断検知・筋長制御・状態推定とそのオンライン学習, 筋破断時の戦略について述べ、この筋冗長性を活かした動作を筋骨格ヒューマノイド Musashi の実機において実現する. 実験により、筋破断時には状態推定・筋長制御・オンライン学習のアルゴリズムを筋破断情報を元に変更することで、現状のネットワーク構造を壊さずに現在の身体状態に適応可能であること、また、途中で筋が一本切れても一連の継続的動作が可能となることを示した.

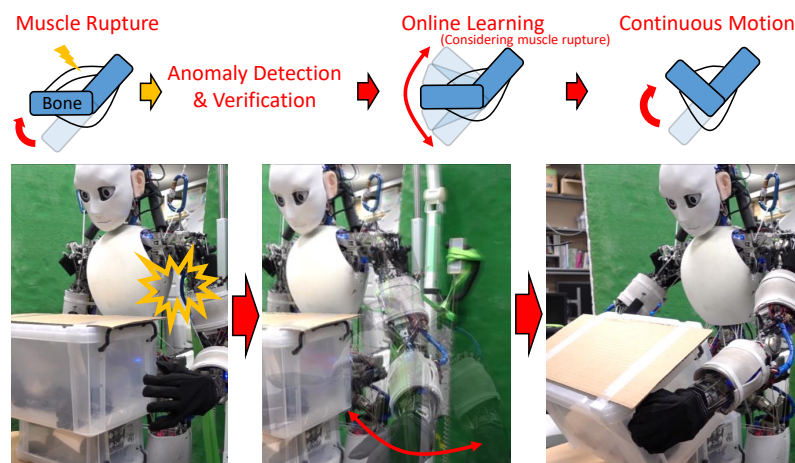


Fig. 7.3: The overall concept of this study: anomaly detection for muscle rupture, verification of the muscle rupture state, online learning of intersensory networks considering muscle rupture, and robust continuous motion control using them.

筋骨格構造の冗長性

これまで筋の冗長性は主に、可変剛性 [125] や二関節筋 [166] という点に対して焦点が当てられてきた。また、大抵の場合は冗長性は制御の障害となるため、冗長性を排除する方法が多く提案されてきた [249, 250]。一方、その他の筋の冗長性の特徴として、筋が一本切れても動く、という重要な特徴があるが、これに関してはこれまでほとんど研究されてこなかった。[123] では筋張力制御において、筋が切れているかどうかの情報を含めることが可能とあるが、それらについての評価はなく、筋が切れているかどうかの判断 (異常検知) や破断後の身体変化の再学習等に関する議論もない。よって本研究では、筋破断検知, 筋破断後の身体モデルのオンライン学習, それを使った状態推定や制御について議論

する必要がある。

柔軟な筋骨格構造に向けた制御・状態推定・異常検知

これまで、柔軟でモデリング困難な筋骨格構造を制御するために多くの研究が成されてきた。筋張力制御手法として [226, 188, 123] の研究が存在する。これらは、筋長ヤコビアンとして、実機により学習された関節-筋空間マッピング [122] を利用している。しかし、球関節や背骨など多くの部分と筋が接触し、摩擦・ヒステリシスが大きいいため意図したように制御することは難しく [123]、本研究では筋長制御を行う。筋長制御手法として、茂木らは、手先位置と筋長を対応付けるテーブルを構築し、正確な手先位置を実現する手法を提案している [127]。また、水内らは、モーションキャプチャデータから関節と筋の関係をニューラルネットワークで表現し、身体制御を行っている [128]。河原塚らは、関節と筋の関係を表すニューラルネットワークをオンラインで学習し身体制御を行う手法 [100]、さらにそれを拡張し、筋張力の影響による身体組織の柔軟性を考慮した手法を開発している [124, 125]。一方、[127] は手先位置と筋長のデータテーブルを直接書き換えるため、筋が切れた場合には全データを一切利用できなくなってしまうので不向きである。[128, 100] は筋張力を扱うことができず、また制御しか扱うことができない。[124, 125] は筋張力を扱うことができるものの、一度のニューラルネットワークの順伝播によって筋長指令値を求めるため、筋が切れた際の制御指令値変化を扱うことができない。

この他にも、1,2 自由度程度のシンプルな空気圧ロボットを、Gaussian Process Regression (GP) [131] や Neural Network・Sequential Quadratic Programming [130] を使って動的に制御する方法が開発されているが、低い自由度でのみ実験されている (多くの自由度を持つロボットでは、関節ごとに異なるコントローラを作っている)。また、筋破断のような身体変化・オンライン学習等については扱うことができていない (特に GP 等のモデルはオンライン学習や身体変化等について柔軟に扱うことが難しい)。筋骨格以外の冗長系についても Dynamic Neural Network や Dynamic Movement Primitives, Reinforcement Learning 等の選択肢があるものの [251, 252]、それぞれ身体モデルに多くの仮定を置いており、身体変化後の再学習・オンライン学習が難しい、複雑な系の実機においては現実的ではないほど学習に時間がかかる等の問題がある。

本研究の貢献

本研究では、[124, 125] の研究をさらに拡張した、6.3 節の Musculoskeletal AutoEncoder [178] を中核に用いる。この手法は、関節角度・筋張力・筋長の相互関係を用いて、状態推定、制御、シミュレーションを統一的行う方法を提供する。損失関数の最小化に基づく最適化により状態推定や制御を行うた

め, 損失関数の定義を変化させることで筋破断を扱うことが可能になる. また, 静的な身体モデルに限ることで, 動的モデルには難しいオンライン学習を現実的な時間で可能としている. 異常検知についてはこれまで [253, 254] 等の手法が提案されており, それらを用いることも可能であるが, 状態推定や制御とは個別にモデルが開発されてきている. 本手法は状態推定・制御・異常検知等を統一的に一つのネットワークで行うため, 個別のモデル管理が必要なくなる点で有効である. 本研究では, 状態推定と制御の枠組みを改良して用い, 筋破断検知の追加, 冗長性を活かした動作戦略へと繋げ, 機能を制限・拡張しているため, Redundant Musculoskeletal AutoEncoder (RMAE) と呼ぶこととする.

本研究は, これまで多く扱われてこなかった, 筋の冗長性を陽に利用した新しいロバスト動作戦略について議論する. 筋が切れたことを検知, 筋破断状態を認識し, 筋破断後のモデルを再学習して, これまでと同じように動作を再開するという一連の動作を行う. また, この動作のために必要なハードウェア・ソフトウェアの構成, ネットワーク構成等まで含め, 包括的な議論を行う.

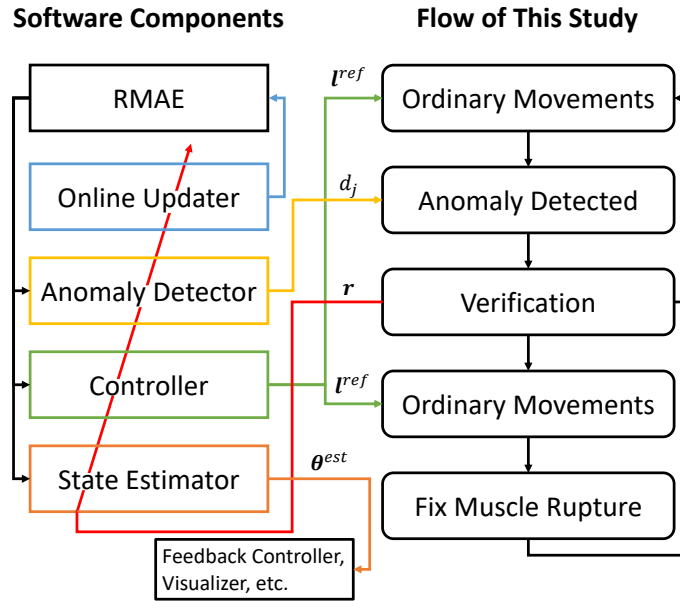


Fig. 7.4: Overview of software components and flow of this study. The muscle rupture information r from Verification (red line) changes each component of online updater, anomaly detector, controller, and state estimator.

7.1.2 筋破断を考慮した適応的身体図式学習

本研究の全体の流れを Fig. 7.4 に示す. 柔軟身体を持つ筋骨格ヒューマノイドが, 筋が切れても動き続けるロバストな動作戦略のために必要な要件は以下であると考ええる.

- 筋が切れたことが検知できる.
- 筋が切れた状態における身体センサ間関係を再学習する.
- 容易に筋を交換できる.

まず、筋の破断を検知できる必要があるが、一口に筋破断と言っても、ロボットにおける筋破断には様々な状態があり得る。本研究では、筋破断や筋が切れたとは、ロボットにおいて、摩擦や外部からの力によって筋が切れた状態、回路やモータ等の故障により筋を能動的に動かせなくなった状態等を総合的に表すものとする。筋の破断が検知できれば、一旦動作を止め、筋の状態を確認して、その後の動作戦略を練ることができる。次に、その筋が切れた状態で動作をする必要があるが、このとき、柔軟な身体における筋長や筋張力、関節の関係が変わってしまっているため、この関係を再学習する必要がある。これにより、筋が切れる前の状態と同じように身体が動かせるようになるべきである。最後に、筋破断をそのままにしていれば、さらに他の筋が切れた際に、動作が継続できなくなってしまう。そのため、最終的には人間が交換するが、この筋の交換が容易である必要がある。

一方、この他にもいくつかの戦略が考えられる。一つ目は、再学習等はせず、筋を交換するまで動作を停止するという方法である。これは今までのアプローチであり、アルゴリズムが非常に単純になる一方、ロボットを動作限界まで使い切れていない。二つ目は、筋破断等は検知せず、常に身体モデルを学習し続けるという方法である。この方法は一つ目と同様にアルゴリズムがシンプルになる一方、筋破断に関する情報を含めずに身体モデルを学習し続けると、モデルが大きく変化し意図した通りに制御・状態推定ができなくなってしまうことが実験章から明らかになっている。三つ目は、筋を交換せずとも人間のように自己修復するという方法である [255]。このアプローチを本研究に適用できれば人間が介入しない完全なシステムが構築可能である一方、現状の技術ではユニットが大きい、自己修復後に伝達可能な張力が破断前より大幅に減少する等の問題点がある。そのため、本研究では筋破断検知、筋破断情報を含めた身体センサ間関係のオンライン学習、その後の筋の交換というアプローチによる、冗長性を使ったロバストな継続的動作戦略に関する手法について述べる。

ハードウェア・ソフトウェアの必要条件

ハードウェアについては、筋の交換容易性のため、4.5 節で述べたセンサドライバ統合型筋モジュール [152]、骨構造一体小型筋モジュール [115] の2つの筋モジュールが開発されている。これらは、モータ・モータドライバ・温度センサ・筋張力測定ユニット等の要素を一つのパッケージに収めることで、信頼性・メンテナンス性を向上させた筋モジュールである。これまでは、筋アクチュエータ [256] はモジュール化されず、モータやモータドライバがそれぞれ別の場所にある形であり、交換の際には

多大な労力を要した。これに対して, [152, 115] はネジを4つ外して交換するのみであり, 非常に容易である。実験したところ, 熟練した研究者で, 約4分で交換可能であった。本研究では, これらの筋モジュールを搭載した筋骨格ヒューマノイド Musashi [87] を実験で用いる。

ソフトウェアについては, 本研究を遂行するにあたり必要な要素は主に, 筋破断に関する異常検知と制御, 状態推定である (状態推定は本研究では直接は必要ないが, 大抵の場合他タスクを実行する上で重要である)。また, これらはある可変なネットワーク構造 (本研究では RMAE) を介してオンラインで学習され, 常に現在の身体状態に適応していく必要がある。筋破断は異常検知をただだけでは完全には判断できないため, その後, 筋破断状態を確認する手順も重要である。最後に, 筋破断時には一部のセンサ値が通常あり得ない値を取ってしまうため, オンライン学習・異常検知・制御・状態推定の全コンポーネントをそれに応じて変化させる必要がある。

以降では, 可変なネットワークである RMAE の構造, RMAE の初期学習, RMAE のオンライン学習, RMAE を用いた制御, RMAE を用いた状態推定, RMAE を用いた異常検知, 筋破断状態の確認を順に述べていく。

Redundant Musculoskeletal Autoencoder

本研究の中核となる Redundant Musculoskeletal AutoEncoder (RMAE) の構造について簡単に再掲する。その使い方やアルゴリズム等は異なるものの, この構造は 6.3 節における Musculoskeletal AutoEncoder と同等のものである。

RMAE は冗長なセンサ群の相互関係を表現するが, 本手法では筋骨格ヒューマノイドで通常得られる関節角度 θ , 筋張力 f , 筋長 l に着目する。これらの相互関係をニューラルネットワークにより可変かつ微分可能な形で表現することで, その関係を更新したり制御や状態推定等に利用したりすることができる。これらの間には, (i) $(\theta, f) \rightarrow l$, (ii) $(f, l) \rightarrow \theta$, (iii) $(l, \theta) \rightarrow f$ という関係が存在する。つまり, (θ, f, l) の3つのセンサ情報のうち, 2つから残り1つを求めることができる, 言い換えれば, (θ, f, l) のうち2つのセンサ情報から, 3つ全てのセンサ情報を取得することに相当する。3つのうち2つのセンサから潜在変数 z が計算でき, この z が3つのセンサに関する情報を持っているのである。よってこの3つの関係を一つのネットワークに縮約すると, (θ, f, l) とマスク変数 m を入力し, 同様の (θ, f, l) を出力する AutoEncoder 型 [227] のネットワークとなる。RMAE は, (θ, f, l) のうち2つのセンサ情報から3つのセンサ情報全てを取得できるという性質を用いて, 得られるセンサ情報を (θ, f) に絞った場合は $(\theta, f, \mathbf{0}, \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T)$ を入力として, (θ, f, l) を出力する。(ii), (iii) も m を変更させながら同様に実行し, 入力と出力が近くなるように学習を行う。まず関節と筋の関係を含む幾何モ

デルから RMAE を初期学習させる。その後、実機センサデータを取得し、このネットワークをオンラインで更新していくことになる。

また、本研究で使用するいくつかの関数を再掲する。

$$z = h_{enc}(\theta, f, l, m) \quad (7.1)$$

$$h_{enc,1}(\theta, f) = h_{enc}(\theta, f, \mathbf{0}, \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T) \quad (7.2)$$

$$h_{enc,2}(f, l) = h_{enc}(\mathbf{0}, f, l, \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^T) \quad (7.3)$$

$$h_{enc,3}(\theta, l) = h_{enc}(\theta, \mathbf{0}, l, \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T) \quad (7.4)$$

$$(\theta, f, l) = h_{dec}(z) \quad (7.5)$$

$$\theta = h_{dec,\theta}(z) \quad (7.6)$$

$$f = h_{dec,f}(z) \quad (7.7)$$

$$l = h_{dec,l}(z) \quad (7.8)$$

RMAE を取り巻くソフトウェアの詳細な全体システムを Fig. 7.5 に示す。この RMAE を中心として誤差逆伝播を利用し、それぞれのコンポーネントが動作している。

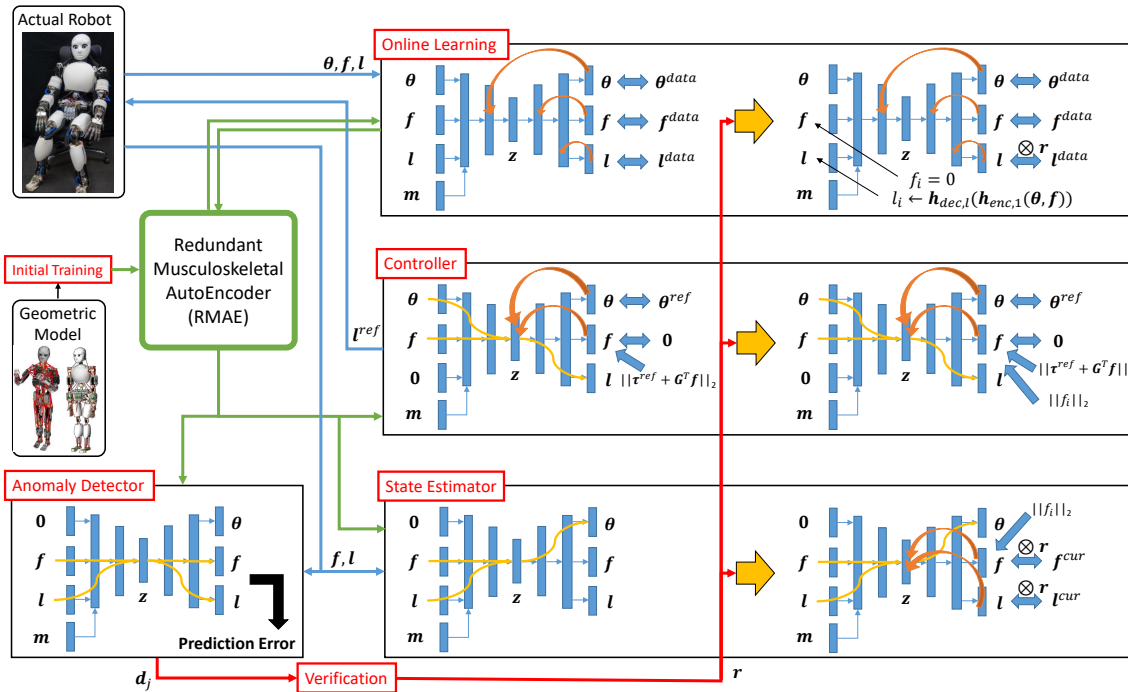


Fig. 7.5: The detailed overall software system.

RMAE の初期学習

初期学習については 6.3 節と同様のため損失関数のみ述べる。幾何モデルから得られたデータにおいて、 (θ, f, l) のうち一つをランダムに $\mathbf{0}$ にし、対応する m を加えたデータを RMAE に入力し、 $(\theta^{pred}, f^{pred}, l^{pred})$ を出力する。その後、損失関数 $L_{initial}$ を $\{\theta, f, l\}$ について以下のように計算し、RMAE を学習させる。

$$L_{initial} = w_{\theta} \|\theta - \theta^{pred}\|_2 + w_f \|f - f^{pred}\|_2 + w_l \|l - l^{pred}\|_2 \quad (7.9)$$

ここで、 $\|\cdot\|_2$ は L2 ノルムを表す。

本研究では学習の際のバッチ数を $N_{batch}^{init} = 100$ 、エポック数を $N_{epoch}^{init} = 30$ 、 $w_{\theta} = 1.0$ 、 $w_f = 10.0$ 、 $w_l = 100.0$ とし、最適化手法は Adam [46] を用いる。

RMAE のオンライン学習

実機センサデータを用いたオンライン学習について述べる。

まず、実機からセンサデータ $(\theta^{cur}, f^{cur}, l^{cur})$ を取得する。ただし、これは関節角度か筋張力が前回の学習の際のデータからある閾値以上離れ、かつ静止している場合に学習データを作成する。

次に、このデータを蓄積・拡張し、RMAE をオンラインで更新していく。データ $(\theta^{cur}, f^{cur}, l^{cur})$ を蓄積していき、データ数が N_{thre}^{online} を超えたところで、更新を開始する。蓄積したデータの中から N_{data}^{online} ($N_{data}^{online} \leq N_{thre}^{online}$) 個、そして最新のデータ 1 個を取得する。また、 $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ というデータも一つ加える。これらデータに対して、それぞれ (i) (ii) (iii) の関係に対応する 3 種類の m を加え、計 $3 \times (N_{data}^{online} + 2)$ 個のデータを用いて、損失関数 $L_{online} = L_{initial}$ として前述の初期学習と同様にネットワークを更新する。初期学習により全体的なネットワーク構造が構築されるため、オンライン学習ではその構造を崩さずに得られたデータに滑らかに合致するようになる。

ここで、筋破断に関する情報 r (切れた筋を 0、切れていない筋を 1 とした M 次元ベクトル) をもとに、オンライン学習の方法を変化させる必要がある。筋が切れた場合はその筋の f は常に 0 付近となり、 l もほとんど変化しなくなるため、そのまま学習を続けると、関係性が大きく崩れて RMAE の構造が破壊されてしまう。そこでまず、得られたデータセット (θ, f, l) に対して、切れた筋 i の f, f_i を全て 0 とする。次に、切れた筋 i の l である l_i を直接使うことはできないため、 $l^{pred} = h_{dec, l}(h_{enc, 1}(\theta, f))$ のように l^{pred} を推論する。最後に、 l_i を l_i^{pred} に置き換え、オンライン学習を実行させる。このとき、

筋 i の \mathbf{l} に関する損失は意味を成さないため、損失関数から抜く必要がある。よって、損失関数は以下のようになる。

$$\begin{aligned} L'_{online} = & w_\theta \|\boldsymbol{\theta} - \boldsymbol{\theta}^{pred}\|_2 \\ & + w_f \|\mathbf{f} - \mathbf{f}^{pred}\|_2 + w_l \|\mathbf{r} \otimes (\mathbf{l} - \mathbf{l}^{pred})\|_2 \end{aligned} \quad (7.10)$$

ここで、 \otimes は要素ごとの掛け算を表す。

本研究では、更新の際のバッチ数を $N_{batch}^{online} = 10$ 、エポック数を $N_{epoch}^{online} = 10$ とし、その他定数は $N_{thre}^{online} = 10$ 、 $N_{data}^{online} = 10$ とした。 N_{epoch}^{online} は大きすぎると得られたデータに適合し過ぎてしまうため、初期学習に比べて十分小さく設定する必要がある。

RMAE による身体制御

RMAE を用いた制御手法について説明する。まず、現在の筋張力 \mathbf{f}^{cur} 、指令関節角度 $\boldsymbol{\theta}^{ref}$ から、潜在状態 $\mathbf{z} = \mathbf{h}_{enc,1}(\boldsymbol{\theta}^{ref}, \mathbf{f}^{cur})$ を求める。次に以下の工程を繰り返す。

- (1) $(\boldsymbol{\theta}^{pred}, \mathbf{f}^{pred}, \mathbf{l}^{pred}) = \mathbf{h}_{dec}(\mathbf{z})$ を求める。
- (2) 損失 $L = \mathbf{h}_{control}(\boldsymbol{\theta}^{pred}, \mathbf{f}^{pred}, \mathbf{l}^{pred})$ を計算する。
- (3) 誤差逆伝播 [140] により \mathbf{z} を更新する。

(2) では、以下のように、筋張力最小化・指令関節角度実現・必要関節トルク実現に関する損失 L を計算する。

$$\begin{aligned} L = \mathbf{h}_{control}(\boldsymbol{\theta}, \mathbf{f}, \mathbf{l}) = & w_1 \|\mathbf{f}\|_2 + w_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}^{ref}\|_2 \\ & + w_3 \|\boldsymbol{\tau}^{ref} + \mathbf{G}^T(\boldsymbol{\theta}^{ref}, \mathbf{f}^{cur})\mathbf{f}\|_2 \end{aligned} \quad (7.11)$$

ここで、 $\boldsymbol{\tau}^{ref}$ は幾何モデルから計算された $\boldsymbol{\theta}^{ref}$ を保つのに必要な関節トルク値、 $\mathbf{G}(\boldsymbol{\theta}, \mathbf{f})$ は筋長ヤコビアンである。(3) では、以下のように誤差逆伝播法 [140] を元に \mathbf{z} を最急降下法で更新する。

$$\mathbf{g} = \partial L / \partial \mathbf{z} \quad (7.12)$$

$$\mathbf{z} = \mathbf{z} - \gamma \mathbf{g} / |\mathbf{g}| \quad (7.13)$$

ここで、 \mathbf{g} は \mathbf{z} に関する L の勾配、 γ は学習率を表す。そして最後に、 \mathbf{z} から得られた \mathbf{l}^{pred} 、 \mathbf{f}^{pred} を用いて筋剛性制御を考慮したうえで \mathbf{l}^{ref} を計算し、実機に指令する。

ここで、筋が切れたという情報 \mathbf{r} をもとに、制御手法を変化させる必要がある。この場合、切れた筋 i の筋張力 f_i は必ず 0 にならなければならない。ゆえに、Eq. 7.11 を一部変更し、以下のような損失を計算する。

$$L = \mathbf{h}'_{control}(\boldsymbol{\theta}, \mathbf{f}, \mathbf{l}) = \mathbf{h}_{control}(\boldsymbol{\theta}, \mathbf{f}, \mathbf{l}) + w_4 \|f_i\|_2 \quad (7.14)$$

ここで、 w_4 は係数であり、この値は w_1 に比べて十分大きくする必要がある。

本研究では、 $w_1 = 1.0$, $w_2 = 1.0$, $w_3 = 0.01$, $w_4 = 10.0$ とする。

RMAE による状態推定

状態推定の手法について述べる。筋骨格ヒューマノイドはこれまでに述べたように一般的には関節角度を測定することができないため、視覚を用いて実機関節角度を推定する。しかしこの場合、身体にマーカ等をつける必要や、身体を常に見続けなければならない等の制約を受けることになる。そこで、現在の筋張力・筋長情報から RMAE を通して関節角度を推定することで、自身の状態を常に知り続けることが可能となる。

方法は単純で、実機から現在筋張力 \mathbf{f}^{cur} 、現在筋長 \mathbf{l}^{cur} を取得し、これを (ii) の形でネットワークに入力し、関節角度推定値 $\boldsymbol{\theta}^{est} = \mathbf{h}_{dec, \boldsymbol{\theta}}(\mathbf{h}_{enc, 2}(\mathbf{f}^{cur}, \mathbf{l}^{cur}))$ を得るのみである。

また、筋が切れたという情報 \mathbf{r} をもとに、状態推定を変化させる必要がある。これまでと同様、切れた筋 i の f_i, l_i が信用できないため、直接 $\mathbf{h}_{enc, 2}$ により \mathbf{z} を求めてしまうと、不正確な値が出力されてしまう。これを解決する方法としては以下の2つが考えられる。

一つ目は、直前における推定関節角度 $\boldsymbol{\theta}^{est, prev}$ と現在の $\boldsymbol{\theta}^{cur}$ には大きな違いが無いという仮定を使う方法である (手法 A)。切れた筋 i の \mathbf{f}^{cur} である $f_i^{cur} = 0$ として、 $\mathbf{l}^{pred} = \mathbf{h}_{dec, \mathbf{l}}(\boldsymbol{\theta}^{est, prev}, \mathbf{f}^{cur})$ を計算し、 $l_i^{cur} = l_i^{pred}$ と更新する。その後、通常と同様に $\boldsymbol{\theta}^{est} = \mathbf{h}_{dec, \boldsymbol{\theta}}(\mathbf{h}_{enc, 2}(\mathbf{f}^{cur}, \mathbf{l}^{cur}))$ を計算するのみである。

二つ目は、前述の制御と同様の形で潜在変数 \mathbf{z} を繰り返し更新していく方法である (手法 A')。まず筋が切れていない状態と同様に $\mathbf{z} = \mathbf{h}_{enc, 2}(\mathbf{f}^{cur}, \mathbf{l}^{cur})$ を計算する。しかし、これは f_i, l_i が通常あり得ない値となっているため、 $\mathbf{h}_{dec}(\mathbf{z})$ により \mathbf{f}, \mathbf{l} を復元しても、同じ値が出るわけではない。そこで、以下の損失関数 L を計算し、前述の制御と同様の形で \mathbf{z} を更新していくことで、現在状態を表す \mathbf{z} を得ることができる。

$$\begin{aligned} L = \mathbf{h}_{estimate}(\mathbf{f}, \mathbf{l}) = & w_5 \|f_i\|_2 + w_6 \|\mathbf{r} \otimes (\mathbf{f} - \mathbf{f}^{cur})\|_2 \\ & + w_7 \|\mathbf{r} \otimes (\mathbf{l} - \mathbf{l}^{cur})\|_2 \end{aligned} \quad (7.15)$$

最後に, $\theta^{est} = \mathbf{h}_{dec}(\mathbf{z})$ により関節角度推定値を得る.

手法 A' の方が正確であることは明らかであるが, 繰り返しに時間がかかるため速い周期で実行することは難しい. それに対して手法 A は, 正確性には疑問が残るものの速い周期で実行することが可能である.

本研究では, $w_5 = 10.0$, $w_6 = 1.0$, $w_7 = 1.0$ とする.

RMAE による異常検知

オンライン学習の際のデータを蓄積していき, その個数が N_{data}^{detect} を超えたところから, データが貯まる度に異常検知を行うモデルを構築し直すことを繰り返す. 同時に, N_{data}^{detect} を超えたデータは古いものから捨てていく. 得られた全データ $(\theta, \mathbf{f}, \mathbf{l})$ に対して, $(\theta^{pred}, \mathbf{f}^{pred}, \mathbf{l}^{pred}) = \mathbf{h}_{dec}(\mathbf{h}_{enc,2}(\mathbf{f}, \mathbf{l}))$ のように, 予測値を計算する. それぞれの筋 j に対して, $\mathbf{v}_j = (f_j, l_j)^T$, $\mathbf{v}_j^{pred} = (f_j^{pred}, l_j^{pred})^T$ を作成する. この誤差 $\mathbf{e}_j = \mathbf{v}_j - \mathbf{v}_j^{pred}$ に関する全データの平均 μ_e と共分散行列 Σ_e を計算する.

実際の異常検知では, 現在の $\mathbf{f}^{cur}, \mathbf{l}^{cur}$ を常に取得し, 同様に $\mathbf{f}^{pred}, \mathbf{l}^{pred}$ を予測して, 以下の値をそれぞれの筋 j に対して計算する.

$$\mathbf{e}_j = \begin{pmatrix} f_j^{cur} & l_j^{cur} \end{pmatrix}^T - \begin{pmatrix} f_j^{pred} & l_j^{pred} \end{pmatrix}^T \quad (7.16)$$

$$d_j = \sqrt{(\mathbf{e}_j - \mu_e)^T \Sigma_e^{-1} (\mathbf{e}_j - \mu_e)} \quad (7.17)$$

この d_j の値が閾値 C_{thre}^{detect} を超えた時, 異常が検知されたと見なす. 異常が検知された際は, 一旦異常検知や RMAE のモデルの更新を停止する.

本研究では $N_{data}^{detect} = 50$, $C_{thre}^{detect} = 30.0$ とした. C_{thre}^{detect} は低すぎると検知されなくなり, 高すぎると異常が検知され続けてしまうため, 予備実験により値を決定している.

筋破断確認

異常が検知されたとき, 現在の筋状態を理解する必要がある. 本研究では筋の状態に, (a) 筋が切れた, またはモータの故障等によって力を伝達することができない状態, (b) 筋は切れたが切れる箇所によっては継続して使える状態, (c) 異常が検知されたものの筋は切れていなかった状態, を考える. (b) については, 筋の端点が固定点から取れる, または筋の端点から非線形性要素までの間のワイヤが切れることによって起こりうる. この場合, 筋を折り返す経路点ユニット等に外れた非線形性要素が引っ掛かり, 筋の 0 点はオフセットしてしまうものの, まだ利用することが可能である. これらを判断する手順を Fig. 7.6 に示す.

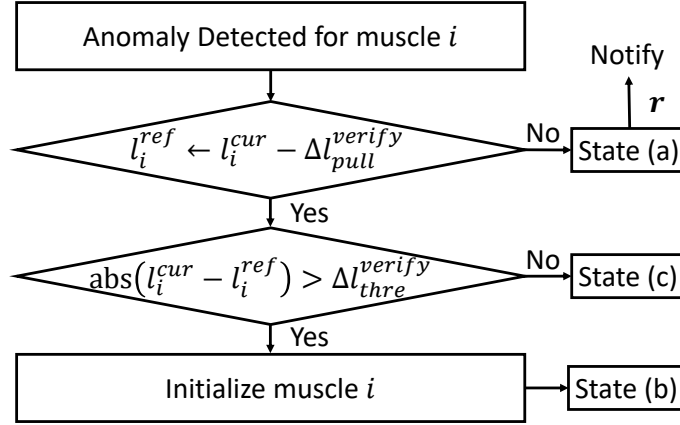


Fig. 7.6: The procedure of muscle rupture verification.

まず、筋は7.3.2節における安全機構と同様の形で最小の筋張力値を設定することで、緩まない状態にする。なお、筋が切れた場合に巻き過ぎるのを防止するため、 l^{cur} は l^{ref} よりも Δl_{max}^{verify} 以上は巻かないという制限を加えている。以降の操作を異常が検知された全ての筋に対して実行する。まず異常が検知された筋 i について、 l_i^{ref} を l_i^{cur} から Δl_{pull}^{verify} だけ巻いた位置に設定した時に、もし筋が切れておらず正しく働いていれば、筋張力が上昇する。つまり、筋張力の変化 Δf_i が Δf_{thre}^{verify} を超えた時、筋は正常に働いているとする。もし筋が正常に働いていない場合は、その筋は切れていると見なし、その筋 i の値を 0 とした r を全コンポーネントに通達し、それぞれの実行の仕方を変化させる。もし筋が正常に働いている場合は、 $\text{abs}(l_i^{cur} - l_i^{ref})$ の大きさに着目する。この値が Δl_{thre}^{verify} よりも大きかった場合は (b) の状態であり、それよりも小さかった場合は (c) であると見なす。

(b) の場合は、力は伝達するものの、筋長の原点がオフセットしてしまっているため、これを初期化する必要がある。ここでも、前述の制御と同様の形で潜在変数 z を繰り返し更新していく方法を用いる。まず $z = h_{enc,1}(\theta^{cur}, f^{cur})$ を計算する。そして、 l_i^{cur} 以外は正しい値のため以下の損失関数 L を計算し、前述の制御と同様の形で z を更新していくことで、現在状態を表す z を得ることができる。

$$\begin{aligned}
 L = h_{verify}(\theta, f, l) = & w_8 \|\theta - \theta^{cur}\|_2 + w_9 \|f - f^{cur}\|_2 \\
 & + w_{10} \|r^{offset} \otimes (l - l^{cur})\|_2
 \end{aligned} \tag{7.18}$$

ここで、 r^{offset} はオフセットしてしまった筋 i のみ 0 で残りは 1 となる M 次元のベクトルとする。最後に、 $l^{est} = h_{dec,l}(z)$ により筋長推定値を得て、 l_i^{cur} を l_i^{est} により初期化する。

また、(a) と (b) の場合はロボットの身体構造が変化してしまうため、それまでに学習された RMAE のモデルは別の場所に保存しておき、筋が換装された後にそのモデルを復帰させる。加えて、(a) と (b)

の場合は, RMAE と異常検知モデルの学習・構築のために蓄積されたデータは全て削除する. 異常検知された全ての筋についてこの操作が終わったら, 通常通りオンライン学習・異常検知モデルの更新等を再開する.

本研究では, $\Delta l_{max}^{verify} = 100$ [mm], $\Delta l_{pull}^{verify} = 10$ [mm], $\Delta f_{thre}^{verify} = 10$ [N], $\Delta l_{thre}^{verify} = 30$ [mm], $w_8 = 1.0$, $w_9 = 1.0$, $w_{10} = 1.0$ とする.

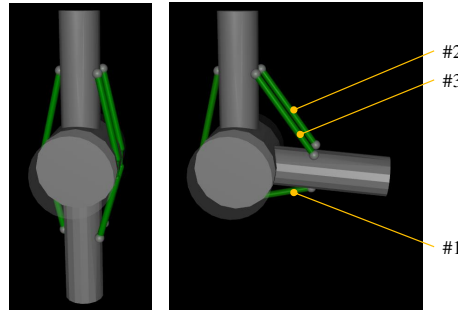


Fig. 7.7: 1 DOF simple joint model in MuJoCo [257].

7.1.3 実験

実験セットアップ

本研究では, 筋骨格ヒューマノイド Musashi [87] を用いる. 全身の筋は 74 本であり, 筋の末端には非線形弾性要素が配置されている. また, 通常の筋骨格ヒューマノイドでは関節角度は測定出来ないが, Musashi には関節モジュールという特殊な機構により, 実験評価のための関節角度センサが配置されており, 実機関節角度としても用いることができる. その中でも, 主に使用するのは腕の 5 つの関節自由度であり, これらを S-p, S-r, S-y, E-p, E-y とする (S, E はそれぞれ肩と肘を表し, rpy はそれぞれ roll, pitch, yaw を表す). これら 5 自由度に関係する筋は 10 本であり, 2 関節筋を一つ含む.

また, 本提案手法の動作をより詳細に確認するため, 1 自由度のシミュレーションを MuJoCo [257] で作成した (Fig. 7.7). 肘の構造を模しており, 筋が 3 本と冗長に配置されている. 実機に近くよう Musashi の筋と同様の特性を持つ非線形弾性要素を配置しており, また, 筋に摩擦損失を持たせている. 筋経路の幾何モデルは経由点を直線で結んだものであるのに対して, シミュレーションでは円筒に巻き付く形をしているため, 実機同様 RMAE の学習が必要な状況を作り出している.

本研究では, 筋破断時の各コンポーネントへの影響, また, これまで説明したコンポーネントの変化による筋破断への対処法について, シミュレーション・実機において順に実験を行う. また, これ

らコンポーネントを用いた、冗長な筋配置を活かした一連の動作実験を行い、本研究の有効性を示す。なお、状態推定については7.1.2節で得られる推定関節角度 θ^{est} と現在のロボットの関節角度 θ^{cur} の差である $RMSE_{est} = \|\theta^{est} - \theta^{cur}\|_2$ を、制御については、指令関節角度 θ^{ref} と θ^{cur} の差である $RMSE_{control} = \|\theta^{ref} - \theta^{cur}\|_2$ を測定し比較する。

1 自由度シミュレーションにおける状態推定実験

1 自由度のシミュレーションを用いて、状態推定・オンライン学習について実験を行う。本実験では、 θ は1自由度、 $\{l, f\}$ は3自由度である。

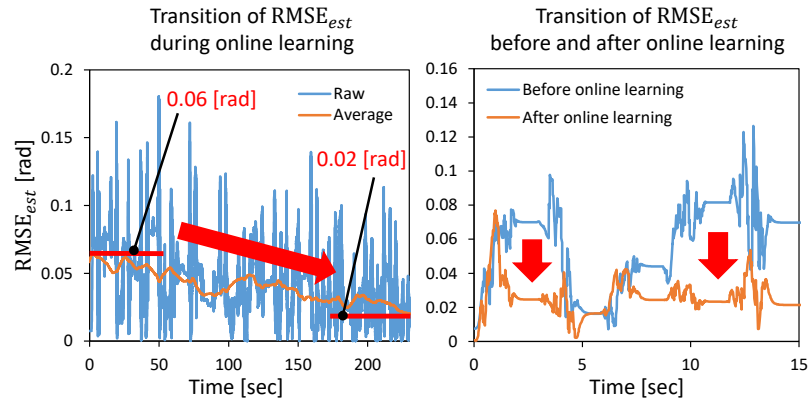


Fig. 7.8: 1 DOF simulation experiment of state estimation without any muscle ruptured: transition of $RMSE_{est}$ during online learning (left graph), and comparison of $RMSE_{est}$ between before and after online learning (right graph).

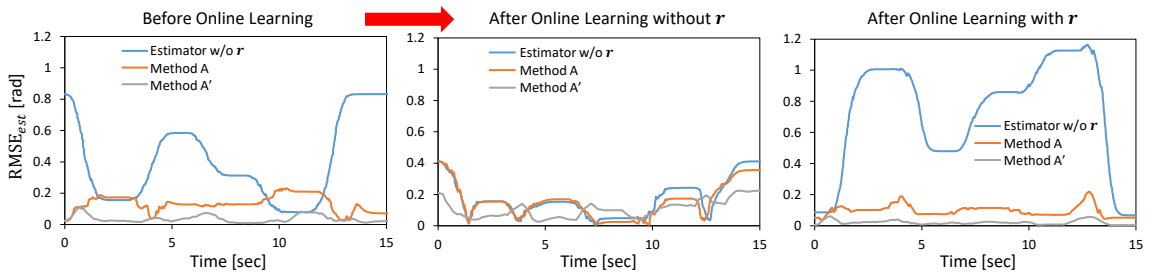


Fig. 7.9: 1 DOF simulation experiment of state estimation when muscle #2 is ruptured: comparison of transitions of $RMSE_{est}$ among without r , with method A, and with method A', before online learning (left graph), after online learning without r (center graph), and after online learning with r (right graph).

まず、オンライン学習を実行した状態で、関節角度範囲内のランダムな関節角度 θ^{rand} を指令することを繰り返す(これをオンライン学習実験動作と呼ぶ)。このとき、本研究の制御手法だけでなく、

$h_{dec,l}(h_{enc,1}(\theta^{rand}, f^{rand}))$ により得られた筋長を指令することで様々な状態を学習データとして得ることができる(f^{rand} はランダムな筋張力). 順に, 本節の制御手法で筋長を2秒で指令し0.5秒休み, 上記の方法で筋長を1秒で指令し0.5秒休むことを繰り返す. $RMSE_{est}$ の遷移をFig. 7.8の左図に示す. 20秒ごとの平均の値も表示している. $RMSE_{est}$ の平均値は徐々に減少し, 4分間かけて約0.06 radから0.02 radまで落ちていることがわかる. 次に, ランダムな5つの関節角度 θ^{rand} を指定し, これを順に本節の制御手法で2秒で指令し0.5秒休む動作を行った(これを評価実験動作と呼ぶ). このとき, オンライン学習前と後のRMAEを使った際の $RMSE_{est}$ の遷移をFig. 7.8の右図に示す. なお, 本実験からは評価実験中にオンライン学習は止めている. ほとんどの時間において, 学習後は学習前に比べて $RMSE_{est}$ が小さくなっていることがわかる. 以降の実験を含めた1DOFシミュレーションにおける状態推定の結果をTable 7.1に示す. 15秒間の本動作における $RMSE_{est}$ の平均, $RMSE_{est}^{ave}$ は, 学習前は0.056 rad, 学習後は0.026 radであった.

次に, オンライン学習後のRMAEを使って状態推定を行う際, 筋#2が切れ, 筋長が100 mmで一定となってしまった場合に, 前述の評価実験動作を行ったときの $RMSE_{est}$ の遷移をFig. 7.9の左図に示す. ここで, 7.1.2節における, 筋の破断情報 r が無かった場合, あった場合の二つの手法(AとA')における状態推定について比較を行う. 15秒間を通して手法A'が最も誤差が少なく, 一部逆転する箇所はあるものの, ついで手法A, 情報 r が無い場合の順で誤差が小さい. $RMSE_{est}^{ave}$ は, r がない場合は0.40 rad, 手法Aは0.13 rad, 手法A'は0.037 radであった. どの状態についても, 学習後の0.026 radよりは誤差が大きく, 筋の破断により学習されていない状態が生まれ, 誤差が増大していることがわかる.

最後に, その筋が切れた状態でオンライン学習を行った後に, 前述の評価実験動作を行ったときの $RMSE_{est}$ の遷移をFig. 7.9の中図・右図に示す. ここで, オンライン学習において, 筋の破断情報 r が無かった場合(Fig. 7.9の中図)とあった場合(Fig. 7.9の右図)について比較を行う. それぞれの場合について, 7.1.2節に関して筋の破断情報 r がない場合, あった場合の二つの手法(手法AとA')における状態推定について比較を行う. 破断情報 r がない状態でオンライン学習を行った場合は, 状態推定において r がない場合と, 手法AまたはA'を使った場合について, A'が多少誤差が小さいものの, 誤差の遷移に大きな差はない. 実際 $RMSE_{est}^{ave}$ は, r がない場合は0.17 rad, 手法Aは0.15 rad, 手法A'は0.11 radであった. よって, 破断情報 r がない状態でオンライン学習を続けると, 手法A, A'については学習前よりも悪い結果となってしまうことがわかった. これに対して, 破断情報 r がある状態でオンライン学習を行った場合について, その誤差は手法A', 手法A, r がない場合の順で小さい. $RMSE_{est}^{ave}$ は, r がない場合は0.73 rad, 手法Aは0.097 rad, 手法A'は0.021 radであった. よって, 筋が切れても r の情報を含めたオンライン学習を続けることで, 状態推定の誤差が小さくなることがわかった. な

お、これらの実験において、手法 A は 100 Hz 以上で実行できるのに対して、手法 A' は最大で 10 Hz 程度でしか実行できない。

Table 7.1: Results of evaluation experiments for state estimator in 1 DOF simulation: the averages of $RMSE_{est}$ during 15 seconds of the evaluation experiment, $RMSE_{est}^{ave}$.

Muscle state	w/o rupture		w/ #2 ruptured		
Online learning	Before	After	Before	After, w/o r	After, w/ r
Estimator w/o r [rad]	0.056	0.026	0.40	0.17	0.73
Method A [rad]	-	-	0.13	0.15	0.097
Method A' [rad]	-	-	0.037	0.11	0.021

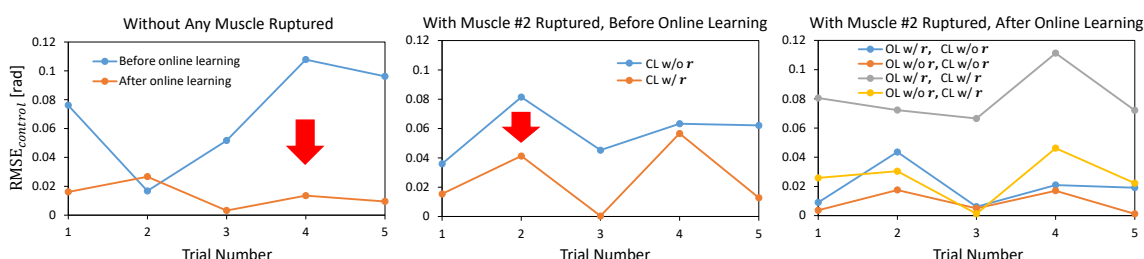


Fig. 7.10: 1 DOF simulation experiment of controller: comparison of transitions of $RMSE_{control}$ between before and after online learning without any muscle ruptured (left graph), between using controller with or without r with muscle #2 ruptured (center graph), and between after online learning with or without r and using controller with or without r with muscle #2 ruptured (right graph). OL means online learning and CL means controller.

1 自由度シミュレーションにおける筋長制御実験

1 自由度のシミュレーションを用いて、筋長制御・オンライン学習について実験を行う。

前述の評価実験動作と同様にランダムな 5 つの関節角度 θ^{rand} を θ^{ref} に指定し、これを順に本節の制御手法の方法で 2 秒で指令し 0.5 秒休む動作を行う。このとき、前述の状態推定実験のオンライン学習前と学習後の RMAE を使い、それぞれの動作後の静止時における $RMSE_{control}$ を計算する。その結果を Fig. 7.10 の左図に示す。2 つめの関節角度では多少逆転しているものの、おおむね学習後は学習前に比べて正確な関節角度を実現できていることがわかる。以降の実験を含めた 1DOF シミュレーションにおける制御の結果を Table 7.2 に示す。5 回の本動作における $RMSE_{control}$ の平均 $RMSE_{control}^{ave}$ は、学習前は 0.070 rad、学習後は 0.014 rad であった。

次に、前述の状態推定実験と同様に筋#2 が切れてしまった場合について、評価実験動作を行ったと

きの $RMSE_{control}$ を Fig. 7.10 の中図に示す. ここで, 本節の制御手法における, 筋の破断情報 \mathbf{r} が無い場合とある場合の筋長制御について比較を行う. グラフから, どの状態においても情報 \mathbf{r} があった場合の方が, 無かった場合に比べて誤差が小さいことがわかる. $RMSE_{control}^{ave}$ は, \mathbf{r} がない場合は 0.058 rad, \mathbf{r} がある場合は 0.025 rad であった. よって, 筋が切れた際, \mathbf{r} の情報の有無に関わらず, 学習後よりも誤差が大きくなっていることもわかる.

最後に, その筋が切れた状態でオンライン学習を行った後に, 評価実験動作を行ったときの $RMSE_{control}$ を Fig. 7.10 の右図に示す. ここで, オンライン学習における, 筋の破断情報 \mathbf{r} がない場合とある場合について比較を行う. また, それぞれについて, 本節の制御手法における筋の破断情報 \mathbf{r} がない場合とある場合の筋長制御に関する比較も行う. \mathbf{r} がない場合とある場合のオンライン学習についてそれぞれ, 筋長制御には \mathbf{r} の情報があった方が誤差が少ないことがわかる. また, オンライン学習についても同様に, \mathbf{r} の情報があった方が誤差が少ないことがわかる.

Table 7.2: Results of evaluation experiments for controller in 1 DOF simulation: the averages of $RMSE_{control}$ for the five joint angles of the evaluation experiment, $RMSE_{control}^{ave}$.

Muscle state	w/o rupture		w/ #2 ruptured		
Online learning	Before	After	Before	After, w/o \mathbf{r}	After, w/ \mathbf{r}
Controller w/o \mathbf{r} [rad]	0.070	0.014	0.058	0.081	0.020
Controller w/ \mathbf{r} [rad]	-	-	0.025	0.025	0.0089

実機における状態推定実験

Musashi 実機の左腕を用いて, 状態推定・オンライン学習について実験を行う. 本節では, θ は 5 自由度, $\{\mathbf{l}, \mathbf{f}\}$ は 10 自由度である.

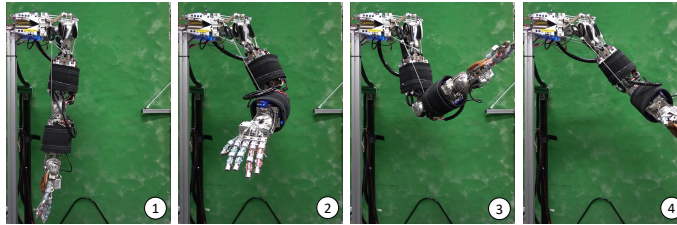


Fig. 7.11: Actual robot experiment of online learning: these pictures show the random movements after online learning of RMAE without any muscle ruptured.

1 自由度シミュレーション実験と同様のオンライン学習を行い (Fig. 7.11), このオンライン学習の前後の RMAE を使った際の, 実機における状態推定の誤差 $RMSE_{est}$ の遷移を Fig. 7.12 に示す. また,

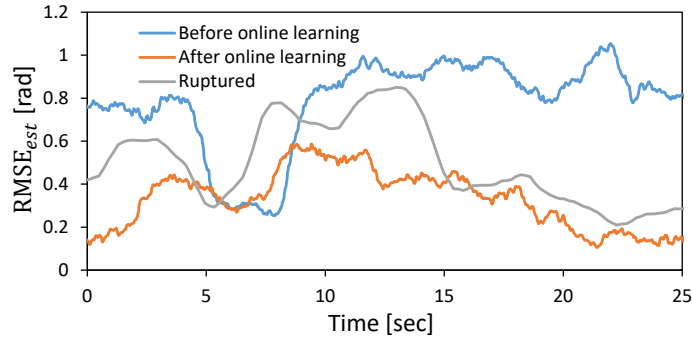


Fig. 7.12: Actual robot experiment of state estimation: comparison of transitions of RMSE_{est} among before online learning, after online learning, and when muscle #9 is ruptured.

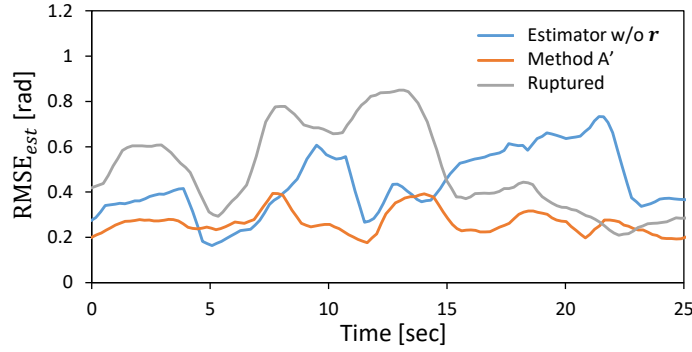


Fig. 7.13: Actual robot experiment of state estimation with muscle #9 ruptured: comparison of transitions of RMSE_{est} between after online learning with and without \mathbf{r} .

オンライン学習後の RMAE について、筋#9 が切れた状態で手法 A' を使った場合についても、 RMSE_{est} を Fig. 7.12 に示す。なお、以降の評価実験動作については 1 自由度シミュレーション実験と基本的には同様であるが、指令するランダムな関節角度数を 7、指令する秒数を 3 秒に増やしている。グラフから、学習後は学習前に比べて関節角度推定値の誤差が大きく減っていることがわかる。また、筋が切れた際には、学習後よりも若干誤差が増えてしまっていることがわかる。以降の実験を含めた実機における状態推定の結果を Table 7.3 に示す。25 秒間の本動作における RMSE_{est} の平均 RMSE_{est}^{ave} は、学習前は 0.78 rad、学習後は 0.33 rad、筋破断時は 0.50 rad であった。

次に、その筋が切れた状態でオンライン学習を行った後に、評価実験動作を行ったときの RMSE_{est} の遷移を Fig. 7.13 に示す。ここで、オンライン学習における、筋の破断情報 \mathbf{r} が無かった場合とあった場合のオンライン学習後について、本節の制御手法で最も精度の高かった手法 A' を用いた時の比較を行う。 \mathbf{r} の情報があつた場合は、無かつた場合に比べて誤差が少ないことがわかる。 \mathbf{r} の情報がなかつ

た場合は、学習前より誤差が少ない場合や誤差が大きい場合があり、大きくは変わらない。RMSE_{est}^{ave}は、 \mathbf{r} の情報がない場合は0.44 rad、 \mathbf{r} の情報がある場合は0.27 radであった。よって、 \mathbf{r} の情報を含めてオンライン学習を継続した結果、筋破断後も筋破断前と同程度かそれ以下の誤差を保つことができる。

Table 7.3: Results of evaluation experiments for state estimator in the actual robot: the averages of RMSE_{est}^{ave} during 25 seconds of the evaluation experiment, RMSE_{est}^{ave}.

Muscle state	w/o rupture		w/ #9 ruptured		
Online learning	Before	After	Before	After, w/o \mathbf{r}	After, w/ \mathbf{r}
Estimator w/o \mathbf{r} [rad]	0.78	0.33	-	-	-
Method A' [rad]	-	-	0.50	0.44	0.27

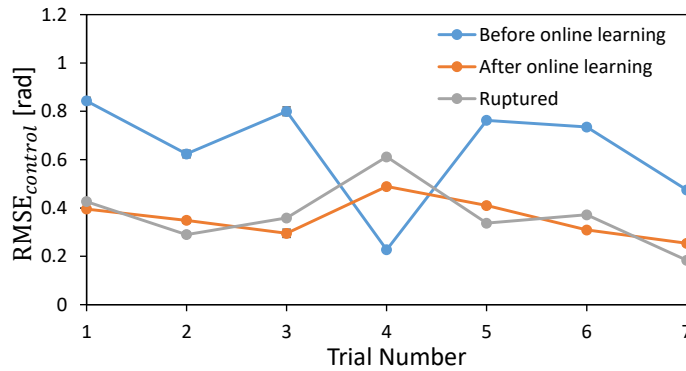


Fig. 7.14: Actual robot experiment of controller: comparison of transitions of RMSE_{control} among before online learning, after online learning, and when muscle #9 is ruptured.

実機における筋長制御実験

Musashi 実機の左腕を用いて、筋長制御・オンライン学習について実験を行う。

前述の評価実験動作と同様にランダムな7つの関節角度 θ^{rand} を θ^{ref} に指定し、これを順に3秒で指令する動作を行う。このとき、前述の状態推定実験のオンライン学習前と学習後の RMAE を使って本節の制御手法を実行し、それぞれの動作後の静止時における RMSE_{control} を計算する。本実験では、同じ動作を5回行い、その平均と分散を計算し、その結果を Fig. 7.14 に示す。また、オンライン学習後の RMAE について、筋#9 が切れた状態で、 \mathbf{r} の情報を使って本節の制御手法を実行した場合についても結果を示す。4つ目の関節角度を除き、基本的に学習後は学習前よりも誤差が小さくなっていることがわかる。また、筋が切れた際は状態推定とは異なりあまり大きく誤差に変化はなかった。これは、筋#9 を使わない状態についても学習が上手くいっているためだと考えられる。また、誤差の分散

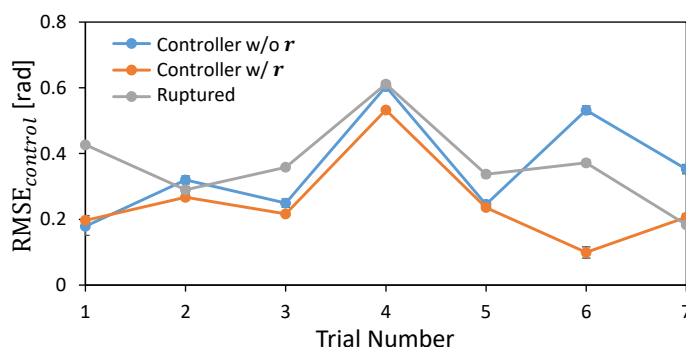


Fig. 7.15: Actual robot experiment of controller with muscle #9 ruptured: comparison of transitions of $RMSE_{control}$ between after online learning with and without r .

は非常に小さい。以降の実験を含めた実機における制御の結果を Table 7.4 に示す。7 回の本動作における $RMSE_{control}$ の平均 $RMSE_{control}^{ave}$ は、学習前は 0.64 rad、学習後は 0.36 rad、筋破断時は 0.37 rad であった。

次に、その筋が切れた状態でオンライン学習を行った後に、評価実験動作を行ったときの $RMSE_{control}$ を Fig. 7.15 に示す。ここで、オンライン学習における、筋の破断情報 r があった場合となかった場合について比較を行う。なお、本節の制御手法では破断情報 r を含めて筋長制御を行う。姿勢によって大きく異なるが、 r の情報がある場合はない場合よりも誤差が小さいことがわかり、シミュレーションと同等の結果であることがわかる。 $RMSE_{control}^{ave}$ は、 r がない場合は 0.35 rad、 r がある場合は 0.25 rad であった。

Table 7.4: Results of evaluation experiments for controller in the actual robot: the averages of $RMSE_{control}$ for the seven joint angles of the evaluation experiment.

Muscle state	w/o rupture		w/ #2 ruptured		
	Before	After	Before	After, w/o r	After, w/ r
Controller w/o r [rad]	0.64	0.36	-	-	-
Controller w/ r [rad]	-	-	0.37	0.35	0.25

実機における異常検知・筋破断評価実験

実機の筋を実際に切り、異常検知と筋破断確認について実験を行う。

まず、筋#9 をハサミで切ったときに、どのように異常検知・筋破断確認が働くかを調べる。通常動作を行い、その後筋を切り、また通常動作を行うという一連の動きをしたときの、筋#9 の異常度 d_j と筋長、筋#8 と#9 の筋張力を Fig. 7.16 に示す。なお、#8 と#9 は互いに似通ったモーメントアームを持つ

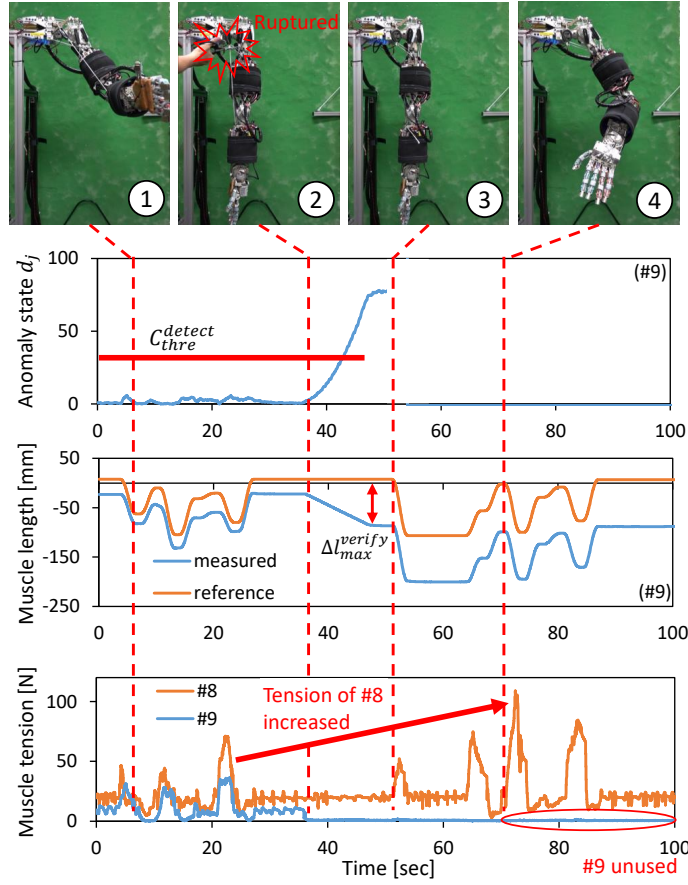


Fig. 7.16: Actual robot experiment of anomaly detection and muscle rupture verification when the muscle wire of muscle #9 is ruptured: the transitions of anomaly state d_j and muscle length of muscle #9, and transition of muscle tension of muscle #8 and #9.

共同筋である。通常動作時には#8の筋張力が最大で約70N、#9の筋張力が最大で約35N程度まで上昇している。#9を切ると、筋張力が0になり、筋長が f^{bias} によって徐々に短くなり、異常度は上がり、途中で C_{thre}^{detect} を超えた（なお、他の筋の異常度はほとんど変化しなかった）。指令筋長と測定された筋長の差は最大値として設定した Δl_{max}^{verify} のところで止まっている。その後、筋長指令が $l_i^{cur} - \Delta l_{pull}^{verify}$ の位置まで変化するが、筋が切れているため筋張力は一切変化しない。そのため#9は破断した（状態(a)）と判断され、 r が全コンポーネントに到達された。その後通常の動作を行うと、ロボットは通常通り関節を動かせるものの、#9には一切の筋張力がかからず、#8は最大で約100Nの力がかかっていた。

次に、筋#8の末端を外したときに、どのように異常検知・筋破断確認が働くかを調べる。肘を曲げ、その状態で筋を切り、その後通常の動作を行うという一連の動きをしたときの、筋#8の異常度 d_j と筋長、筋#8と#9の筋張力をFig. 7.17に示す。#8の筋破断後、筋長が f^{bias} によって徐々に短くなり、異

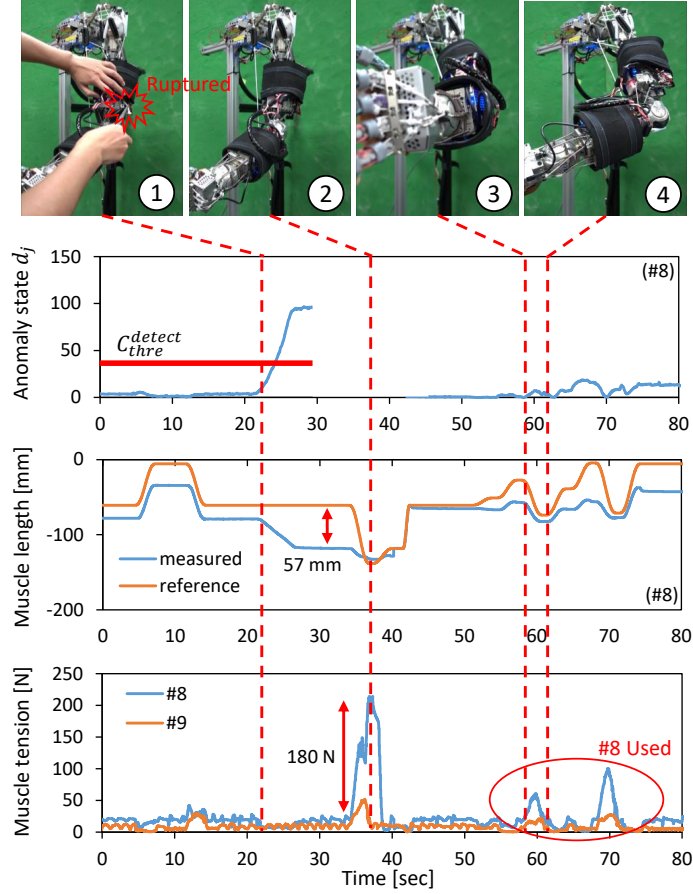


Fig. 7.17: Actual robot experiment of anomaly detection and muscle rupture verification when the endpoint of muscle #8 is ruptured: the transitions of anomaly state d_j and muscle length of muscle #8, and transition of muscle tension of muscle #8 and #9.

常度は上がり、途中で C_{thre}^{detect} を超えた (なお、他の筋の異常度はほとんど変化しなかった)。指令筋長と測定された筋長の差は最大値として設定した 100 mm よりも短い 57 mm のところで止まっている。その後、筋長指令が $l_i^{cur} - \Delta l_{pull}^{verify}$ の位置まで変化し、 l_i^{cur} はそれに追従し、筋張力が 180 N 程度増大した。 $\Delta f_i > \Delta f_{thre}^{verify}$ かつ $\text{abs}(l_i^{cur} - l_i^{ref}) > \Delta l_{thre}^{verify}$ であるため、#8 は状態 (b) と判断され、筋長の初期化が行われた。その後通常の動作を行うと、ロボットは通常通り関節を動かすことができ、#9 だけでなく、#8 にも 120 N 程度の力がかかっていることがわかる。

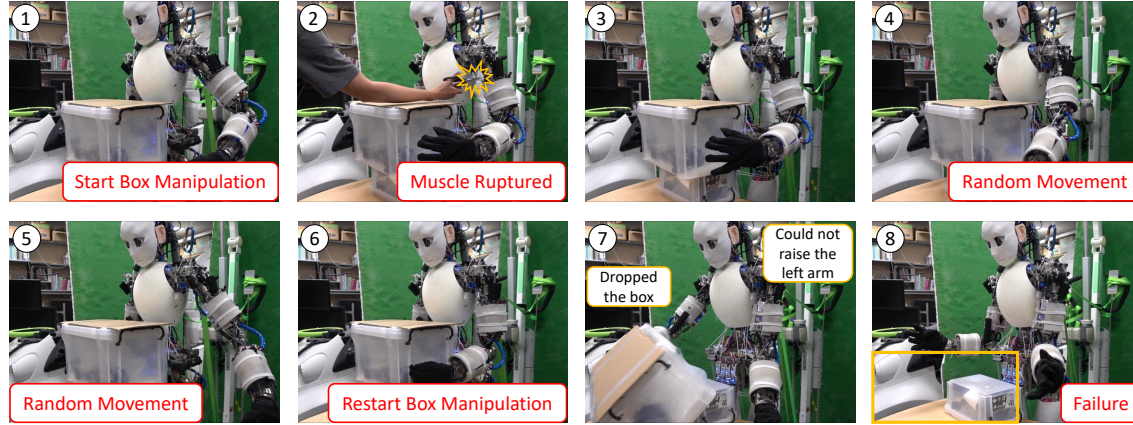


Fig. 7.18: Continuous motion experiment by Musashi without muscle rupture verification: Musashi starts box manipulation, muscle rupture occurs, RMAE is updated online during random movements, Musashi restarts box manipulation, but Musashi dropped the box because its left arm could not be raised as expected.

筋骨格ヒューマノイド Musashi による統合動作実験

筋骨格ヒューマノイド Musashi を用いて箱のマニピュレーション実験を行い、その際に筋破断・異常検知・筋破断確認・オンライン学習という一連の流れを行った。大きな箱を両手で持ち、移動させる動作を行う。これまで開発したコンポーネントを全て実行しておき、箱を持つ手前で左腕の筋#9をハサミで切断する。このとき、前述の筋破断確認を行う場合と行わない場合で比較する。行った場合は r が全コンポーネントに通達されるのに対して、行わない場合は筋が切れても r は考慮されない。その後、前述の実験と同様にランダムな関節角度・筋張力によって左腕を動かし、オンライン学習させ、元通りに実験を再開する。このとき、すぐにオンライン学習が再開されるよう、 $N_{thre}^{online} = 2$ とした。

その全体の動作シーケンスについて、筋破断確認を行わない場合を Fig. 7.18 に、行う場合を Fig. 7.19 に示す。筋破断確認を行わない場合は、一つの筋長が異常な値のまま学習が進むため、ランダム動作による学習後に動きを再開する過程で、意図したように手が上がらなくなり、箱を落としてしまっていることがわかる。これに対して筋破断確認を行う場合は、筋破断情報 r が通達され、正しく学習が進むため、動作の再開後、右手と左手が正しく協調して動き、箱を正しくマニピュレーションできていることがわかる。

7.1.4 議論

本研究の実験結果から、RMAE のオンライン学習・状態推定・制御について、筋が破断した際におけるそれらコンポーネントへの影響について考察する。

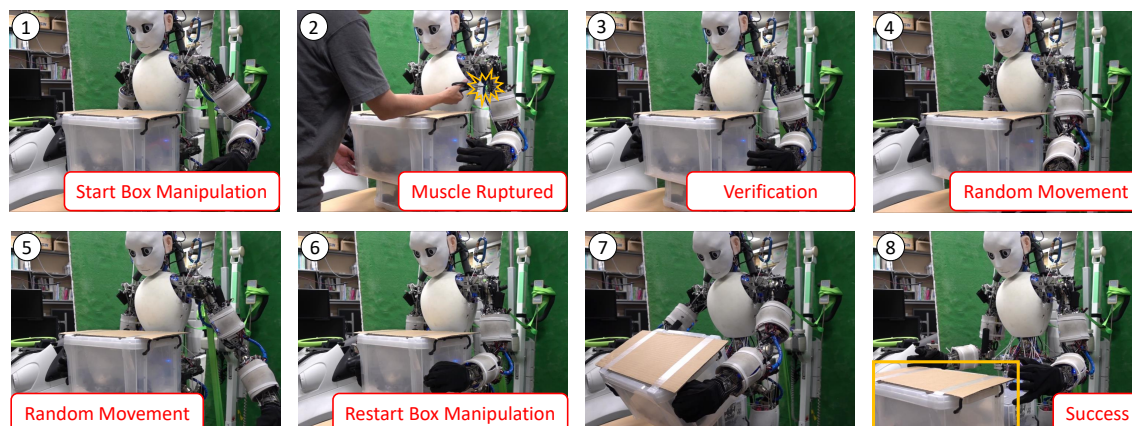


Fig. 7.19: Continuous motion experiment by Musashi with muscle rupture verification: Musashi starts box manipulation, muscle rupture occurs, muscle rupture verification is executed to notify muscle rupture state r to all components, RMAE is updated online during random movements, Musashi restarts box manipulation, and Musashi succeeded in putting the box down with both arms.

まず, 1 自由度のシミュレーション実験では, 理想的にどのようなパフォーマンスの違いがあるのかを知ることができる. オンライン学習では RMAE をオンラインで学習させることで, 4 分間で関節角度推定の誤差が 1/3 程度まで減少した. 状態推定においては, 筋が切れた際は, r の情報を用いることで大幅に誤差を軽減できる. その中でも, 繰り返し計算を行う手法 A' は計算コストは高いものの, 手法 A よりも高い精度を出すことが可能である. また, 筋破断後, r を用いずにオンライン学習させると, 推定誤差が非常に大きくなるのに対して, r を利用して学習方法を変化させることで, 筋破断前よりもさらに推定誤差を減らすことが可能なことがわかった. この傾向は RMAE を使った制御についても同様であり, RMAE のオンライン学習により大幅に制御誤差を減らすことができ, r の情報を用いることで筋破断後のオンライン学習によってもさらに誤差を減らすことが可能である.

次に, 5 自由度の Musashi の左肩・肘における実験では, 実機においてその性能がどの程度変化するかを知ることが可能である. 状態推定についてはシミュレーションとほぼ同じ傾向にあり, RMAE の学習によって大幅に誤差を改善することができ, r の情報を含めたオンライン学習が有効であった. しかし, 誤差のピークは抑えられるものの, 手法 A と A' についてはシミュレーションほど大きな差を見ることは出来なかった. これは, シミュレーションほど正しく RMAE が学習されるわけではないため, その誤差に手法 A と A' の違いが紛れてしまったと考えられる. また, 制御についてもオンライン学習により誤差を低減することができ, 同様に r の情報を含めたオンライン学習は有効であった. しかし, 全体的に r を含めた場合と含めない場合における誤差の変化はシミュレーションよりも小さく,

自由度が多いため誤差は姿勢に大きく依存していた。また、異常検知・筋破断確認は正確に現在の筋状態を判断することができ、使えない筋を認識して r を通達する、使えるが初期筋長がオフセットしてしまった筋を初期化して再度利用することができることがわかった。

最後に、全体の流れを記述した統合動作実験では、筋破断状態 r の有無によってタスク遂行に大きな差が出ることがわかった。 r の情報がない状態でオンライン学習を実行し続けると、破断した筋の異常なセンサ値がネットワーク全体に悪影響を及ぼし、ネットワーク構造が崩れ、タスクが失敗してしまうことがわかった。

本研究から、冗長な筋配置の性質を最大限に利用するためには、筋破断に気づき、それを適切に処理して、その柔軟な身体構造に関するオンライン学習・状態推定・制御等を行っていく必要があることがわかった。また、交換可能なハードウェア・異常検知や筋破断確認等の全体システムも重要である。本手法の考え方は、今後一部のセンサや駆動系が壊れる等のロボットの異常への対処法を与える一つの手段になると考えている。

7.2 身体図式ネットワーク入出力の増加 - 筋追加を考慮した適応的身体図式学習

7.2.1 概要と先行研究

筋骨格ヒューマノイドは人体模倣型の様々な利点を有する。その中でも冗長な筋肉は最も重要な特徴の一つである。この利点として、可変剛性制御 [88, 125, 108] や冗長駆動戦略 [258, 123, 165] が開発されている。一方欠点として、主動筋と拮抗筋の存在とモデル誤差による筋内力の高まりが挙げられ、これらを解決する拮抗筋抑制制御 [167], 筋弛緩制御 [189], 拮抗筋修正制御 [168] 等が提案されている。

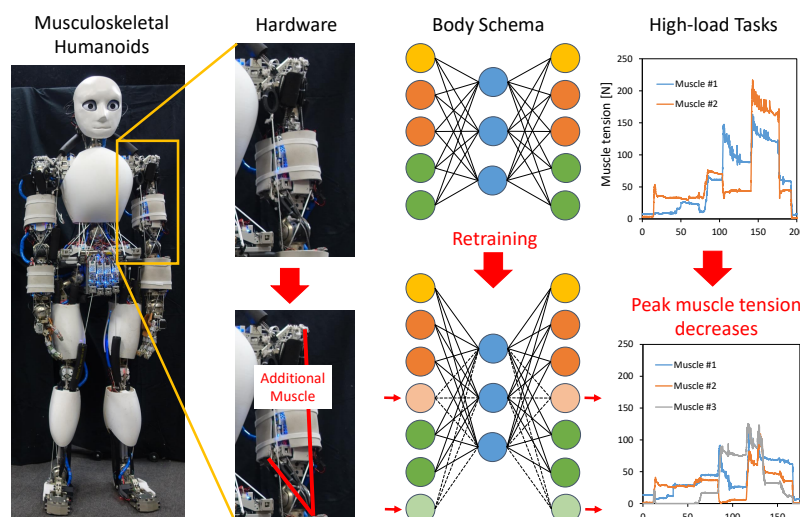


Fig. 7.20: The concept of the whole system of adaptive body schema learning considering additional muscles.

本研究はこれらの中でも、タスクに応じた筋の増加、つまり身体のスensa・アクチュエータの増加について扱う。タスクに応じた筋の増加は軸駆動型ロボットにはない非常に魅力的な特徴である。これまでの研究では、筋ワイヤの中間に筋が宙吊りになる構造の筋モジュールをタスクに応じて追加していた [258]。しかし、筋と同時にその配線も中に浮くため、信頼性に問題があり、骨格に直接筋モジュールを貼り付ける方式が主流になりつつある [107, 87]。そのため、本研究では筋モジュールと筋モジュール間を繋ぐアタッチメントにより、筋を増加させていく。筋張力測定ユニットの取り付け方向次第で筋の方向を自在に変化させ、筋経由点ユニットにより自由な筋配置を実現する。これらの筋追加の観点から、開発した筋骨格ヒューマノイド Musashi [87] の身体構造を捉え直す。

また、これまで筋を追加した後は人間がそのモーメントアームや筋配置をモデル化し、手作業で動作を生成していた。本研究では、関節角度-筋張力-筋長の相互関係を表現する身体図式 [178] を筋骨格

ヒューマノイドの制御に用い、筋の増加により変化した身体図式ネットワークを少数のデータから学習し直す手法について考察する (Fig. 7.20). つまり、筋の追加後、自動でデータを取得、身体図式を再学習し、動作を再開することが可能になる。

身体図式学習の観点で、いくつかの先行研究を紹介する。これまで、関節角度-筋長マッピング学習 [128, 100], 作業空間-筋長マッピング学習 [127], 関節角度-筋張力-筋長マッピング [124, 125] 等、様々な身体図式学習が提案されてきた。これらは、複雑で柔軟な筋骨格構造をニューラルネットワークによりモデル化し、制御や状態推定を行う。その中でも本研究では、これまで個別に開発されてきた制御やシミュレーション、状態推定、異常検知等を一つのネットワークで可能にした **Musculoskeletal AutoEncoder** を用いる [178]。これにより、たった一つのネットワークについてのみ、筋の増加を考えれば良い。ここで重要なのは、**Musculoskeletal AutoEncoder** に限らず、ほとんどのネットワーク [124, 125] において、ネットワークの入力にも出力にも筋長や筋張力等の筋センサ情報が入るため、入力次元の増加と出力次元の増加を扱う必要があるという点である。一方、筋骨格だけではない学習システムについて考えると、出力次元の増加を扱った研究は **Incremental Learning** の文脈で見られる。破壊的忘却を防ぎながらも、ネットワークの出力を増やし、新しいデータにより継続的に学習を行う [259, 260]。一方、入力次元の変化についてはほとんど扱われてきていない。また、大抵は画像認識タスクについてその分類するラベルが増えるというタスクであり、ロボットのセンサ・アクチュエータに関する回帰問題への応用はない。これは、ほとんどの場合ロボットはそのセンサ・アクチュエータが変化しない、成長しないシステムであることを前提としているからであると考えられる。

これらの観点から、本研究では筋を容易に追加可能な筋骨格ヒューマノイドにおける、筋増加を考慮可能な身体図式学習システムの開発について述べる。本研究の貢献は以下である。

- 筋骨格ヒューマノイドにおける筋増加に対応したハードウェアの要件と設計
- 筋増加に伴う身体図式変化時における少数のデータによる身体図式の再学習
- 筋増加を考慮した身体図式学習システムによるタスク実現

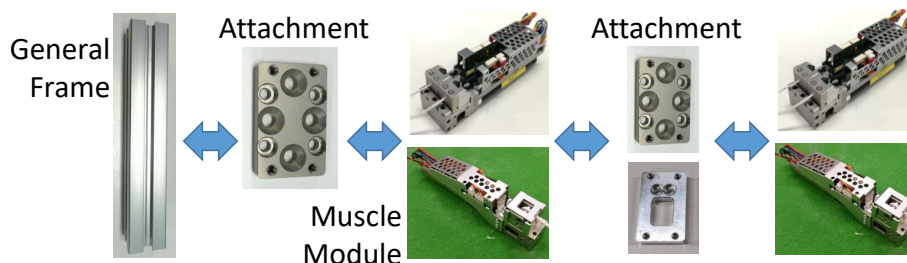


Fig. 7.21: The hardware design considering additional muscles.

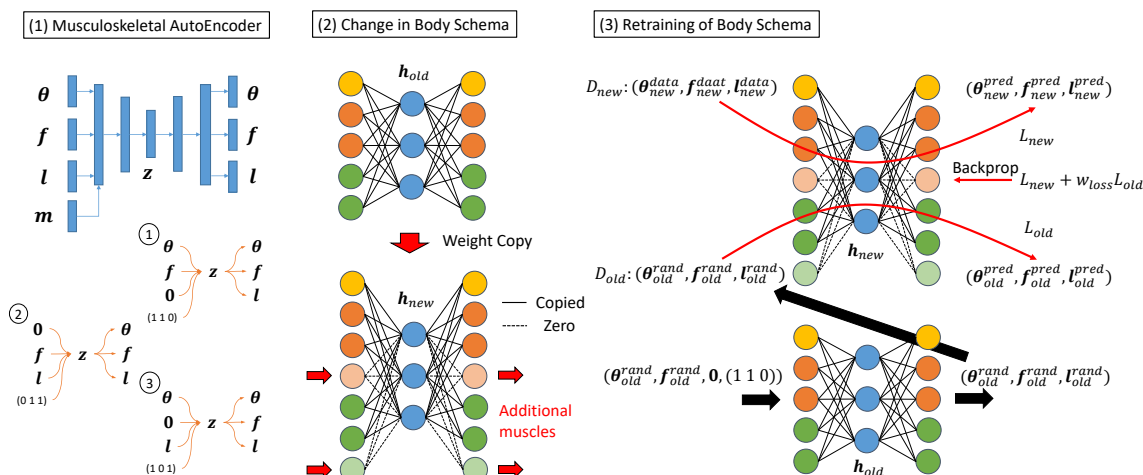


Fig. 7.22: The adaptive body schema learning system considering additional muscles.

7.2.2 筋追加を考慮したハードウェア設計

筋骨格ヒューマノイド Musashi の筋骨格構造について Fig. 7.21 に示す. 筋は, Fig. 4.6 と同様のアクチュエータ・プーリー・モータドライバ・筋張力測定ユニット等を一体化した筋モジュールにより構成されている [152, 115]. これにより, 信頼性やモジュール性が上がり, 筋の交換や筋の追加が容易になる. ここで, 筋追加を可能な身体構造に必要な点は以下の3つである.

- (1) 身体の様々な部位に筋モジュールを取り付けることができる.
- (2) 筋ワイヤを筋モジュールから様々な方向に出すことができる.
- (3) 筋の経路点を様々な指定し, 任意の筋経路を実現できる.

これらにより, 任意の筋経路で, 任意の場所に筋を追加することができるようになる. (1) は, Fig. 7.21 にあるように, Muscle Attachment により可能になる. 全身の骨格を汎用フレームとし, これと筋モジュールを Muscle Attachment により接続する. また, 筋モジュールと筋モジュール同士も Muscle Attachment により接続することができる. (2) は, Fig. 4.7 にあるように, 様々な筋ワイヤ方向を実現可能な筋張力測定ユニットにより可能になる. 筋張力測定ユニットは筋ワイヤの方向転換におけるモーメントによりロードセルを押しこむことで筋張力を測定できる. このユニットが筋モジュールの四面に対して接続可能であり, かつ, そこから四方向全てに出て行くことができる. (3) は, Fig. 4.11 にあるように, 様々な筋経路を実現可能な筋経路ユニットにより可能になる. 筋ワイヤが出る方向と筋経路ユニットを骨格に取り付ける方向の組み合わせからこのユニットを標準化し, これらの組み合わせにより任意の筋

経路を実現することができる。

回路全体は USB 通信によって通信している。各部位に配置された USB ハブ基板からデジチチェーンで筋モジュール内のモータドライバ同士が接続されている。新しい筋を追加する際は、近くに配置されている筋からケーブルを伸ばすのみで良い。

なお、Muscle Attachment, 筋モジュール, 筋ワイヤ, ケーブル類を新しく取り付けるのを、熟練した研究者は約 3 分で終わることができた。

7.2.3 筋追加を考慮した適応的身体図式学習

本システムの全体構成を Fig. 7.22 に示す。Musculoskeletal AutoEncoder の詳細構造, 筋増加の際のネットワーク構造変化, ネットワークの再学習手法が順に示されている。

身体図式学習: Musculoskeletal AutoEncoder

まず、本研究で扱う身体図式モデル, Musculoskeletal AutoEncoder (MAE) [178] について再掲する。MAE は、 (θ, f, l) の間の関係である、 $(\theta, f) \rightarrow l, (f, l) \rightarrow \theta, (l, \theta) \rightarrow f$ という 3 つの関係を一つのネットワークで表したものである。入力を (θ, f, l) とマスク値 m として、潜在空間 z を通して、出力を (θ, f, l) とした AutoEncoder 型のネットワーク h を実機センサデータから更新する。マスク m は、 $\begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T$, $\begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^T$, $\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T$ の 3 つの値を取り、例えば $m = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T$ のとき、 $(\theta, f, 0, \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T)$ を入力として、 (θ, f, l) を出力する。ここで、例えば現在の推定関節角度 θ^{est} を (f, l) の情報から計算するためには、マスク $m = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^T$ を用いれば良い。また、ある指令値 θ^{ref} を実現するための筋長 l^{ref} を計算するためには、 θ^{ref} と適当な f から z を計算する。そして、 z から (θ, f, l) を出力し、この値に対して、 θ が θ^{ref} に近づく制約、 f を最小化する制約、必要な関節トルクを発揮する制約を考慮した損失を計算する。この損失をもとに、 z を誤差逆伝播と勾配法により繰り返し更新することで、最終的に指令筋長 l^{ref} を計算することができる。なお、MAE は静的なセンサ関係のみ表しているため、摩擦やヒステリシス等によるモデル誤差までは吸収できない。

筋追加による身体図式の変容

筋増加により、MAE のモデルである h は変化する。筋増加前のモデルを h_{old} 、筋増加後のモデルを h_{new} とする。 h に用いられる関節数を N 、筋数を M とし、筋増加前と後の筋数を $M_{\{old, new\}}$ とする ($M_{new} > M_{old}$)。MAE の入出力の次元は (N, M_{old}, M_{old}) から (N, M_{new}, M_{new}) へと変化する。ここで、

h_{new} のモデルを完全に最初から学習し直すのは非常に効率が悪いので、 h_{old} の重みをコピーして用いる。コピーされる以外の部分については、ネットワークの重み・バイアスをともに0とする。これにより、 h_{new} の学習前については、入力において追加された筋の f, l にどんな値を入れても、元々ある筋については h_{old} と同様の挙動を示す。なお、実際には MAE は5層のネットワークで記述されているが、Fig. 7.22 の (2), (3) においては3層のネットワークに省略して記述している。

身体図式学習に向けたデータ収集

それぞれの筋は Eq. 2.4 の筋剛性制御 [121] によって動作する。筋増加の際、まずは新しくデータを取得する必要がある。この際、新しく追加された筋については (θ, f, l) の関係が分かっていないため、この筋のみ他の筋とは別の駆動を行いデータを取る。具体的には、Eq. 2.4 における k_{stiff} を新しい筋のみ0とし、指令筋長への追従を行わない。その代わりに、 f^{bias} をランダムに指定する。元からある筋に対しては、 $(\theta_{new}^{rand}, f_{new}^{rand}, \mathbf{0}, (1 \ 1 \ 0)^T)$ を h_{new} に入力し、得られた l^{ref} を実機に指令する ($\{\theta, f\}_{new}^{rand}$ は筋増加後のランダムな $\{\theta, f\}$ を表す)。このときに得られた $(\theta_{new}^{data}, f_{new}^{data}, l_{new}^{data})$ のデータを D_{new} 、そのデータ数を N_{new} とする。

身体図式の再学習

最後に、 h_{old} と得られたデータ D_{new} を同時に使うことで、少数のデータでも効率的に h_{new} を再学習する。 D_{new} が比較的少数のデータの場合、これらのデータだけで h_{new} を学習してしまうとそのデータだけに過学習してしまい、 h_{old} の情報を忘却してしまう。一方、筋が追加されることによってネットワーク全体の構造も変化するため、 h_{old} の情報は参考にはなるものの h_{new} として正しくはない。そこで、本研究では以下のように損失関数を定義して h_{new} の学習を行う。

$$L = L_{new} + w_{loss} L_{old} \quad (7.19)$$

$$L_{new} = \|\theta_{new}^{pred} - \theta_{new}^{data}\|_2 + \|f_{new}^{pred} - f_{new}^{data}\|_2 + \|l_{new}^{pred} - l_{new}^{data}\|_2 \quad (7.20)$$

$$L_{old} = \|\theta_{old'}^{pred} - \theta_{old'}^{rand}\|_2 + \|r \otimes (f_{old'}^{pred} - f_{old'}^{rand})\|_2 \\ + \|r \otimes (l_{old'}^{pred} - l_{old'}^{rand})\|_2 \quad (7.21)$$

ここで、 L_{new} は D_{new} に関する損失、 L_{old} は h_{old} に関する損失、 w_{loss} は損失に関する重みの係数を表す。 $\{\theta, f, l\}_{new}^{pred}$ は h_{new} に $\{\theta, f, l\}_{new}^{data}$ を入力した際に予測された値、 $\{\theta, f, l\}_{old'}^{pred}$ は h_{new} に $\{\theta, f, l\}_{old'}^{rand}$ を入力した際に予測された値を表す。 r は元からある筋に対しては1、新しく追加された筋に対しては

0を示すマスク値であり, \otimes は要素ごとの積を表す. ここで, L_{old} について, $\{\theta, f, l\}_{old'}^{rand}$ を求める必要がある. まず h_{old} への入力としてランダムな $\{\theta, f\}_{old}^{rand}$ を用意し, $h_{old}(\theta_{old}^{rand}, f_{old}^{rand}, 0, (1 \ 1 \ 0)^T)$ から l_{old}^{rand} を計算する. 次に, このデータを h_{new} への入力とするため, 新しく追加された筋については0を付け加えた $\{\theta, f, l\}_{old'}^{rand}$ を作成する. 新しく追加された筋の情報は正確ではないため, マスク r をかけ, これを損失とする. なお, $\theta_{old'}^{rand} = \theta_{old}^{rand}$ である. 本研究は Distillation [260] と似た考え方である一方, ネットワークの入出力に対して次元が追加される点・ h_{old} が h_{new} に対して正しいとは限らない点が異なる.

本研究では以下の3つのケースについて比較・考察も行う.

(i) $w_{loss} = 0$

(ii) $w_{loss} = 1.0$

(iii) $w_{loss} = 1.0 - e/N_{epoch}$

ここで, e は現在のエポック数, N_{epoch} は学習全体のエポック数を表す. 仮説として, (i) は過学習し h_{old} の情報を忘却してしまい, (ii) は h_{new} としては正しくない h_{old} の情報が常に入り続けてしまい, (iii) が最も良く少数のデータから h_{new} を再学習することができると考えられる.

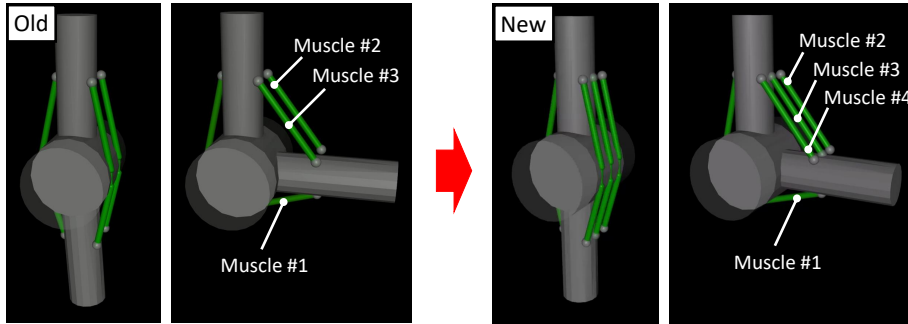


Fig. 7.23: The old and new designs of muscle arrangement for the experiment of 1-DOF tendon robot simulation.

7.2.4 実験

1 自由度腱駆動シミュレーション実験

1 自由度腱駆動シミュレーション実験について説明する. Fig. 7.23 の左図に示すような人間の肘を模した 1 自由度 3 筋のロボットを MuJoCo [257] 上に作成する. 実機に近づくよう Musashi [87] の筋

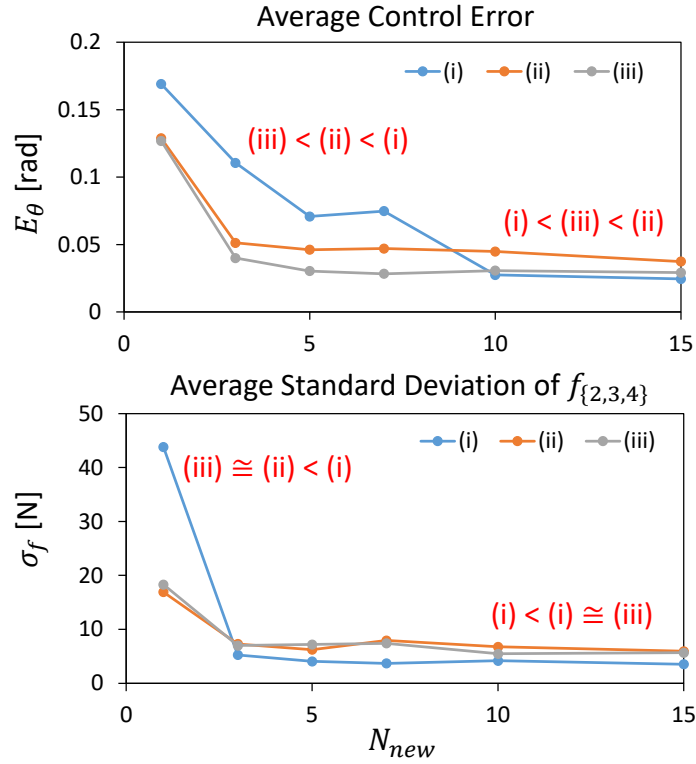


Fig. 7.24: The evaluation experiment of 1-DOF tendon robot simulation. The graphs show E_θ (upper) and σ_f (lower) when changing the number of collected data N_{new} , regarding the methods of (i)–(iii).

と同様の特性を持つ非線形弾性要素を配置し、筋に摩擦損失を持たせ、筋のモーメントアームも関節角度に応じて変化するような構成としている。筋配置としては、屈曲側に 2 本の筋 (#2, #3), 伸展側に 1 本の筋 (#1) を配置している。この元々の筋配置を Old, この屈曲側の筋を 3 本 (#2, #3, #4) に増やした筋配置を New と呼び、筋増加に伴うモデル学習実験を行う。

まず、十分に学習された \mathbf{h}_{old} から、重みのコピーにより \mathbf{h}_{new} を作成する。次に、7.2.3 節の方法で D_{new} を収集し、(i)–(iii) の手法で \mathbf{h}_{new} を更新する。得られた \mathbf{h}_{new} に対して評価実験を行う。指令関節角度 θ^{ref} を 16 個定め (ヒステリシスを考慮して 0–120 deg の間を 15 deg ずつ進んで戻る θ^{ref} を生成した), MAE を使った制御を行い、この際の制御誤差の平均 $E_\theta = \|\theta^{ref} - \theta\|_2$ と屈曲側の 3 つの筋 (#2–#4) の筋張力の標準偏差の平均 σ_f を評価する。 E_θ , σ_f ともに小さくなることが望ましい。また、データ数 N_{new} を $\{1, 3, 5, 7, 10, 15\}$ に変化させ、このときの評価値の変化についても考察する。その結果を Fig. 7.24 に示す。 E_θ は、 N_{new} が小さいときは (iii)<(ii)<(i) の順で小さい。一方、 N_{new} が 10 を超えたところで (i)<(iii)<(ii) となった。 σ_f は、 N_{new} が極端に少ないときは (i)≅(iii)<(ii) であるが、それ以降は (i)<(iii)≅(ii) となった。なお、再学習前については、 $E_\theta = 0.0214$, $\sigma_f = 7.74$ であった。 $N_{new} = 15$

において (i) は $E_\theta = 0.0245$, $\sigma_f = 3.54$ であり, E_θ は同程度, σ_f は半分以下まで下げることができていた. また, 7.2.3 節の重みコピーを行わない (i) の場合は $N_{new} = 15$ でも $E_\theta = 0.158$ であり, 重みコピーは必須であった.

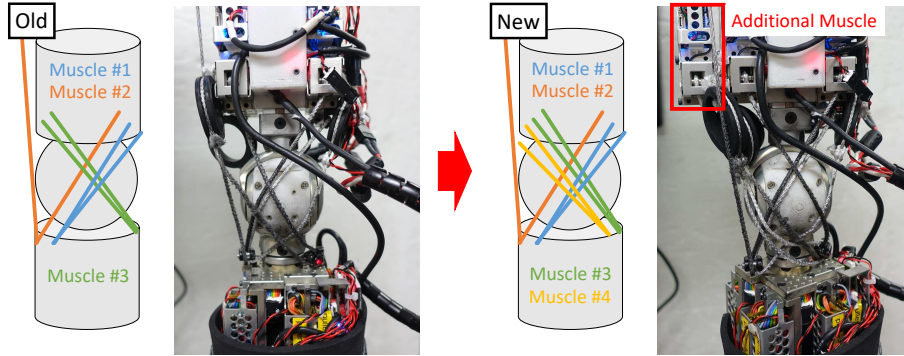


Fig. 7.25: The old and new designs of muscle arrangement for the experiment of the left arm of the musculoskeletal humanoid Musashi.

筋骨格ヒューマノイド Musashi による実機実験

次に, 筋骨格ヒューマノイド Musashi [87] の左腕を用いた実験について説明する. 肩・肘・手首を有する左腕の中でも, 肩・肘の5自由度10筋についてMAEを作成し実験を行う. Fig. 7.25 に示すように肘の屈曲側に備わる3本の筋(#1, #2, #3)に4つめ(#4)を加えて配置した. 元々の10本の筋配置を Old, 新しい11本の筋配置を New と呼び, 筋増加に伴うモデル学習実験を行う.

シミュレーション実験と同様に D_{new} を取得し, (i)–(iii) の方法で \mathbf{h}_{new} を更新する. 評価実験については, ランダムな指令関節角度 θ^{ref} を10個定め, これらに対して制御誤差の平均 E_θ を評価する. 筋張力については, モーメントアームがそれぞれ異なるため同じタイミングでは全く異なる筋張力が出るが多いため, シミュレーション実験と同様に評価することは難しい. そのため, 本研究では $E_f = |f_3^{ave} - f_4^{ave}| / (f_3^{ave} + f_4^{ave})$ と, f_4^{max} について評価を行う. なお, $f_{\{3,4\}}^{ave}$ は評価実験における筋#3と#4の平均の筋張力, f_4^{max} は評価実験における筋#4の最大の筋張力を表す. これにより, 評価実験中における, 役割の近い f_3 と f_4 が全体として近いかどうか, 新しく追加した筋#4に無理な力がかかっていないかを知ることができる. モーメントアームが異なり f_3^{ave} と f_4^{ave} の正しい比率は分からないため, E_f については (i)–(iii) に関する E_f の大きさの比率 R_f を評価値とする. また, データ数 N_{new} を {5, 10, 15, 20, 30} に変化させ, このときの評価値の変化についても考察する. その結果を Fig. 7.26 に示す. E_θ は, 一部逆になることがあるものの, 全体的に (iii)<(ii)<(i) の順で小さい. R_f は, 実験の

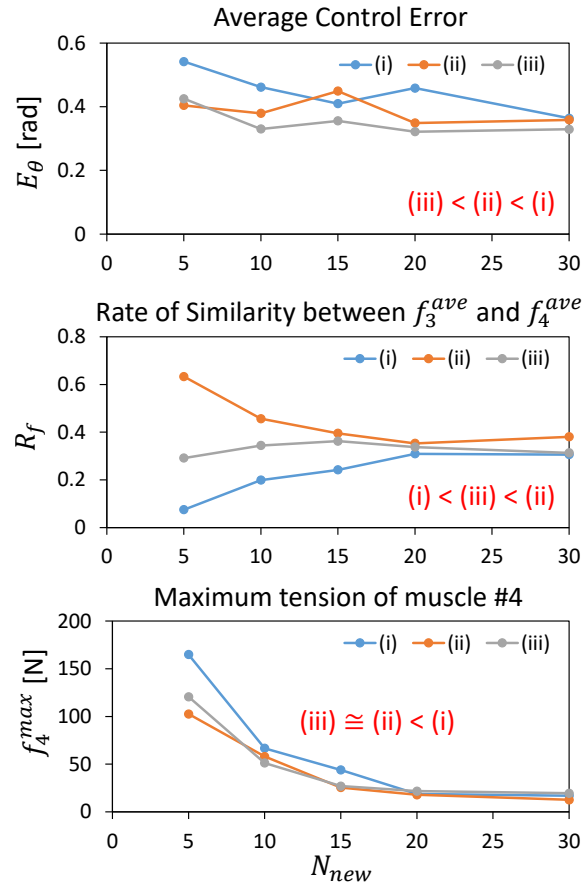


Fig. 7.26: The evaluation experiment of the left arm of the musculoskeletal humanoid Musashi. The graphs show E_θ (upper) R_f (middle), and f_4^{max} (lower) when changing the number of collected data N_{new} , regarding the methods of (i)–(iii).

範囲内では常に (i)<(iii)<(ii) であり、その差は N_{new} が増えるにつれて小さくなっていった。 f_4^{max} は、(iii)≈(ii)<(i) であり、その差は N_{new} が増えるにつれて小さくなっていった。なお、再学習前については、 $E_\theta = 0.526$ であった。 $N_{new} = 30$ において (iii) は $E_\theta = 0.329$ であり、筋の追加と再学習により誤差は大幅に小さくなった。

高負荷動作実験

最後に、高負荷のかかるタスクを行い本研究の筋追加の有効性を示す。本実験では、筋骨格ヒューマノイド Musashi [87] と同じ双腕にメカナム台車と昇降自由度が追加された Musashi-W を用いて実験を行う。Fig. 7.27 の左図に肩のピッチ自由度に直接関与する筋 (#1, #2) を抜き出し示す。これに筋#3を追加し、これまでと同様に h_{new} の再学習を行う。この際、データ数は $N_{new} = 17$ であり、前節まで

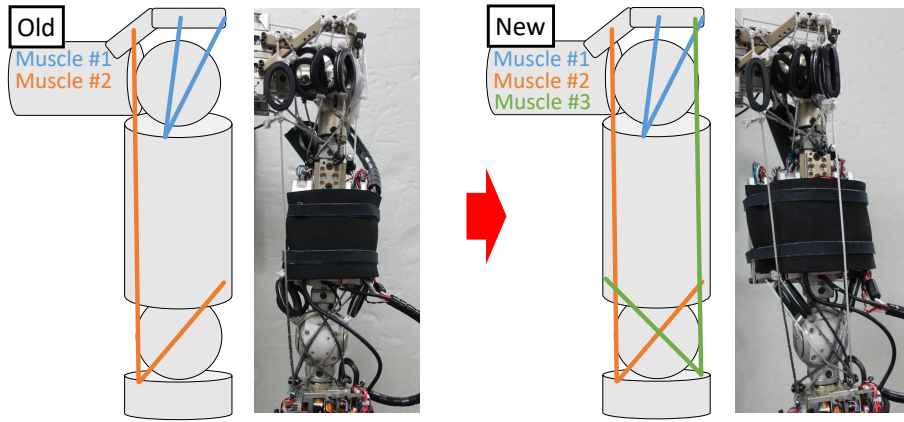


Fig. 7.27: The old and new designs of muscle arrangement for the high-load task experiment using Muashi-W.

の実験から制御誤差の小ささと追加した筋の積極的利用を両立できる (iii) の方法を使って再学習を行った。筋を追加する前と追加し再学習を行った後について、6.8 kg の重りを持ち上げて低い台から高い台の方へ片付ける実験を行った (Fig. 7.28)。この際に最も力のかかる肩のピッチ自由度に直接関与する筋の筋張力遷移を Fig. 7.29 に示す。指令している関節角度等は同一のものだが、人間がそのタイミングや台車の操作を行っているため実行時間は異なることに注意されたい。筋追加前は筋張力に最大で 215 N の力がかかっているのに対して、筋追加後は最大で 118 N と、大幅に筋張力を削減できていることがわかる。また、筋追加後はピーク時には筋#1-#3 の間で筋張力が一様に分散されており、再学習によって追加した筋を正しく利用できていることがわかる。

7.2.5 議論

本研究の実験結果について考察する。

シミュレーション実験では、本手法の全体的な特性を理解することができた。制御誤差について、得られるデータ数が少ない状況においては、 h_{old} に関する情報を徐々に減衰させる (iii) の手法が最も正確である。 h_{old} の情報を常に利用する (ii) は (iii) には劣るもの、新しく得られたデータのみを用いる (i) に比べると精度は高かった。一方、得られるデータ数が多い場合は、わざわざ h_{old} の情報を利用する必要はなく、得られたデータのみから学習を行う (i) の精度が最も高かった。これに対して共同筋の筋張力の差は、得られるデータ数が極端に少ない場合は (i) の精度は著しく低い一方で、ある程度のデータ数があれば (ii), (iii) を抜き (i) の精度が最も高かった。 h_{old} の情報は追加した筋を積極的に利用することを妨げる効果があると考えられる。全体として、どの方法も再学習前よりも制御誤差を減

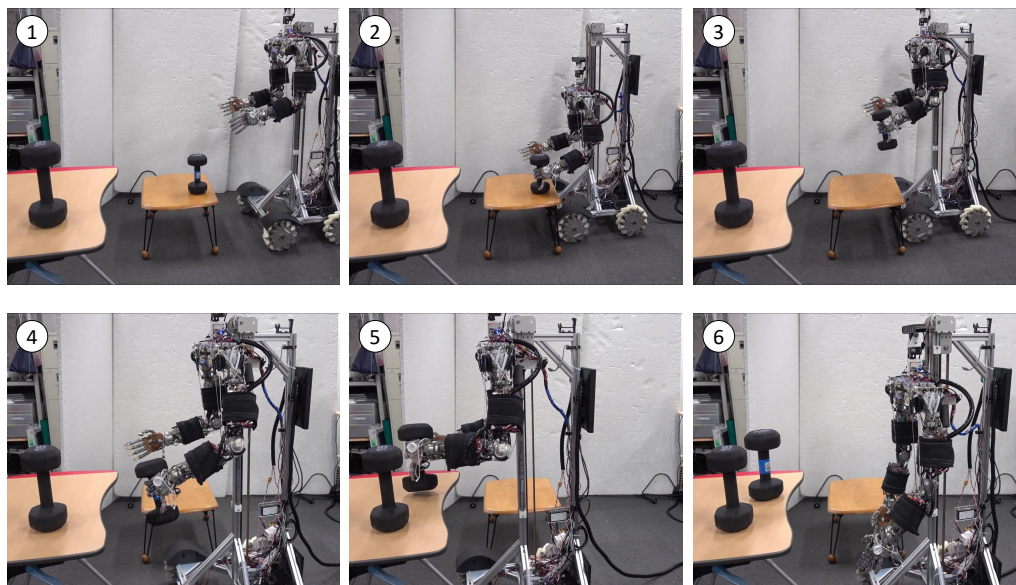


Fig. 7.28: The high-load task experiment using Muashi-W.

らし、追加した筋を利用して他の筋の負荷を下げるように働いていることがわかった。また、重みのコピーを行わない場合は極端に制御誤差が大きく、重みのコピーは必須であることがわかった。

次に、実機実験では本シミュレーションの結果と同様の結果を得ることができた。実機実験では1自由度のシミュレーションと比べて5自由度の関節を扱っているため、同様のデータ数ではシミュレーションにおけるデータ数が非常に少ない状態と等価であると考えられる。制御誤差は全体を通して (iii)<(ii)<(i) であり、シミュレーションにおける $N_{new} < 10$ の状態と特性は一致している。新しく追加した筋と、役割の近い筋の筋張力は再学習により近くなり、その精度は (i)<(iii)<(ii) であることから、シミュレーションにおける $N_{new} > 1$ の状態と特性は一致していると考えられる。また、(i)–(iii) の制御誤差と筋張力差の違いは N_{new} が大きくなるにつれて小さくなっていくことも分かった。加えて、 N_{new} が小さい時は (i) が (ii), (iii) に比べて高い筋張力を発揮してしまうため、気をつける必要がある。これらの知見から、少数の N_{new} で再学習を行いたい場合は (iii) が良く、たくさんの学習データを集めることが可能な場合は (i) の方法が良いことが分かった。

最後に、実際にロボットで高負荷のかかるタスクを実行し、この際の筋追加の効果を検証した。筋を追加し、 $N_{new} = 17$ という比較的少数のデータで (iii) の再学習を実行することで、筋を追加しない場合に比べて大幅に筋張力のピークを下げる事が可能であることが分かった。たった17のデータでもモーメントアームの近い複数の共同筋に筋張力を分配し、安全にタスクを実行可能になることがわかった。

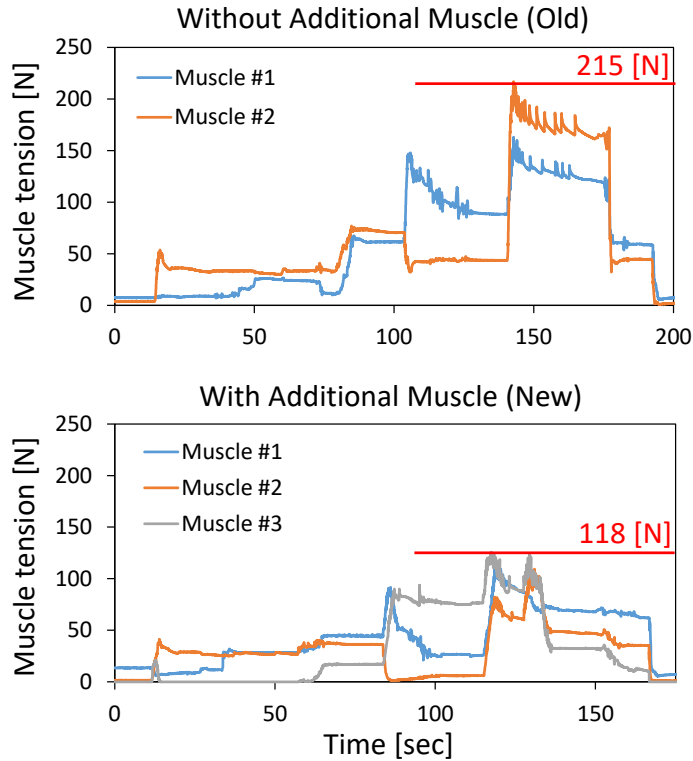


Fig. 7.29: The evaluation experiment of the high-load task. The graphs show the transition of muscle tensions relating with the movement of the shoulder pitch, when using the hardware without an additional muscle (upper) and with an additional muscle #3 (lower).

本研究は Musculoskeletal AutoEncoder に対して手法を適用したが、実際には損失関数の定義を変えるのみで、様々なネットワーク構造に対して利用することができる。本研究はネットワーク構造を限定するものではなく、アクチュエータの容易な追加を可能とするハードウェア・これをもとに少数のデータから身体図式を再学習して動いていくロボットシステムの提案である。今後、アクチュエータを増やしたり、減らしたりするロボットにおいて、ネットワーク入出力次元の減少と増加を考慮し、効率的に身体図式を再構築・成長していくロボット開発に役立つと考えられる。

7.3 確率を含む静的身体図式学習 - 危険回避学習

7.3.1 概要と先行研究

筋骨格ヒューマノイドは様々な生物模倣型の利点を有すると同時に、その複雑な筋骨格構造はモデル化が難しく、これまで様々な学習型制御手法が開発されてきている [122, 100, 125, 178]. しかしこれらの手法は主に静的なセンサ間関係のみの獲得に着目しており、筋骨格構造に含まれる筋の摩擦やヒステリシスの影響から、必ず多少の誤差が残る. また、これらの手法は関節-筋の関係を学習するのみであり、骨格間の位置関係や干渉等は幾何モデルを用いる他ない. 一方で、筋骨格構造においては、さらに弾性バネや筋、骨格らの間の干渉等の影響も有り、通常の自己干渉回避では解決することができない. そのため、間違った拮抗関係から意図せず大きな筋内力が発生してしまったり、幾何モデルと実機のずれから身体同士が干渉して大きな筋張力が発生したりすることがある.

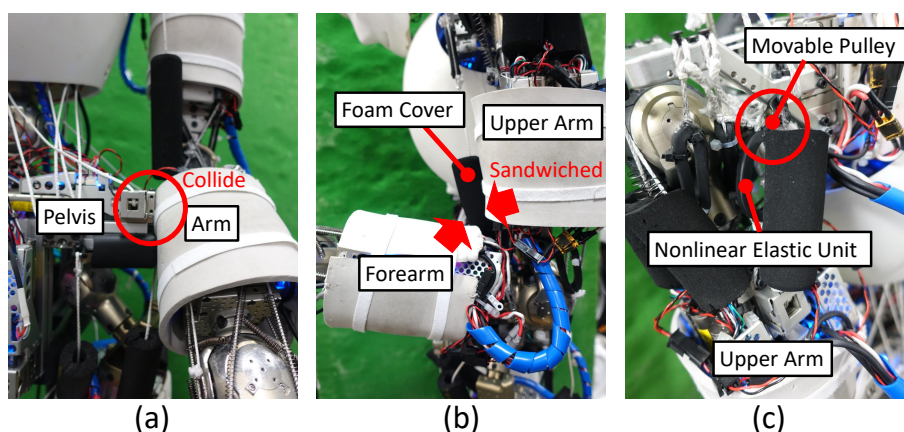


Fig. 7.30: Various danger situations of musculoskeletal humanoids [261].

これまで、この問題を解決するために、様々な安全機構が開発されてきた. [125] では温度と筋張力を考慮して筋長を緩める安全機構が採用されている. [189] では姿勢を崩さないように必要のない筋から順番に筋を緩めていくことで、筋内力を削減している. [255] は異なるアプローチであるが、筋に装着した修復モジュールが大きな力が加わると断裂し、時間が経つと元に戻るような機構を採用している. これらのアプローチは、大きな筋張力が発生、つまり危険が起きたときに始めて作動し、それを未然に防ぐものではない. 筋張力に比例して筋を伸ばすような動作をしたとしても、干渉や大きな筋内力は突発的に起こるため、それに素早く対応することはできない.

そこで本研究では、制御指令に対応した危険度をオンラインで学習し、ロボットが徐々に未然に危険を防ぐことができるようになる手法を提案する [261]. 制御指令である筋長に対応した危険度をニュー

ラルネットワークにより表現する．これを用いて指令筋長を安全なものに修正したり，優先度付き逆運動学に使用することで危険を回避しながら動作させたりすることが可能になる．本研究のネットワークは，静的な一般化多感覚相関モデルにおいて， m と p を消し，出力を確率とした形を用いる．

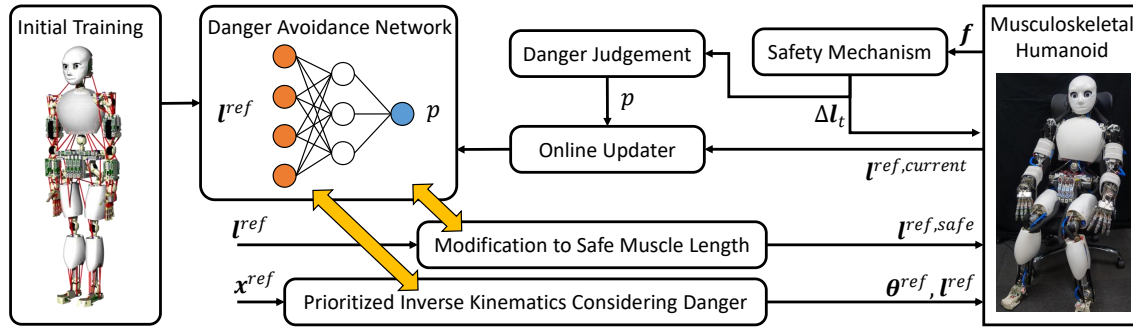


Fig. 7.31: The whole system of danger avoidance for musculoskeletal humanoids [261].

7.3.2 危険回避のオンライン学習

複雑身体構造の問題点

複雑でモデル化困難な筋骨格構造では主に2つの危険状態が起こりえる．1つ目に，拮抗関係は閉リンク構造を作るため，モデル誤差により大きな筋内力が発生する可能性がある．2つ目に，骨格間の干渉により，大きな筋張力が発生する可能性がある．その例を Fig. 7.30 に示す．通常の軸駆動ヒューマノイドであれば幾何モデルから事前に干渉を計算することが可能であるが，筋骨格構造においてはそう簡単な話ではない．まず，制御指令が関節角度ではなく筋長であり，その指令筋長を送ったとしても，学習誤差等によって完全にその関節角度を実現できるとは限らない．また，複雑な構造ゆえに，Fig. 7.30 の (a) のように単純に腰リンクと前腕リンクが干渉する場合だけでなく，(b) のように上腕リンクと前腕リンクの間に発泡性の筋外装が挟まれて干渉する場合もある．その他，(c) のように筋の折り返し部品と非線形弾性要素が干渉し，それ以上筋を引っ張ることができなる状態も存在する．そしてこの危険状態は経年劣化等によって常に変化していくものであり，逐次的に更新していくことが望ましい．

ネットワーク構造

危険回避システムの全体像を Fig. 7.31 に示す．本研究で提案する Danger Avoidance Network (DAN) の構造について説明する．軸駆動型ロボットは制御入力関節角度 θ^{ref} なのに対し，筋骨格型ロボッ

トは制御入力筋長 l^{ref} となる．そのため、軸駆動型では θ^{ref} に対応したリンク間干渉が得られるのに対して、筋骨格型では l^{ref} に対応した危険度が得られる．ここで言う危険には、前述した拮抗関係の誤差による高い筋内力と干渉による高い筋張力が含まれる．もし筋骨格型において θ^{ref} に対応する危険度を表すとする、筋内力や、Fig. 7.30 にあるような筋の状態によって変化する干渉を考慮することができない．よって、本研究では以下の関数 h_{dan} を学習していく．

$$p = h_{dan}(l^{ref}) \quad (7.22)$$

ここで、 p は危険度 ($0 \leq p \leq 1$) を表し、その定義については後に説明する．また、 l^{ref} は M 次元のベクトルである (M は関係する筋の数を表す)．

この関数 h_{dan} をニューラルネットワークで表現する．本研究では4層のニューラルネットワークとし、それぞれのユニット数は $M, 64, 64, 1$ とした．最終層以外の各層の後には Batch Normalization [45] を適用した．中間層の活性化関数は ReLU [138] であり、最終層は 0 から 1 の確率を出すため Sigmoid とした．

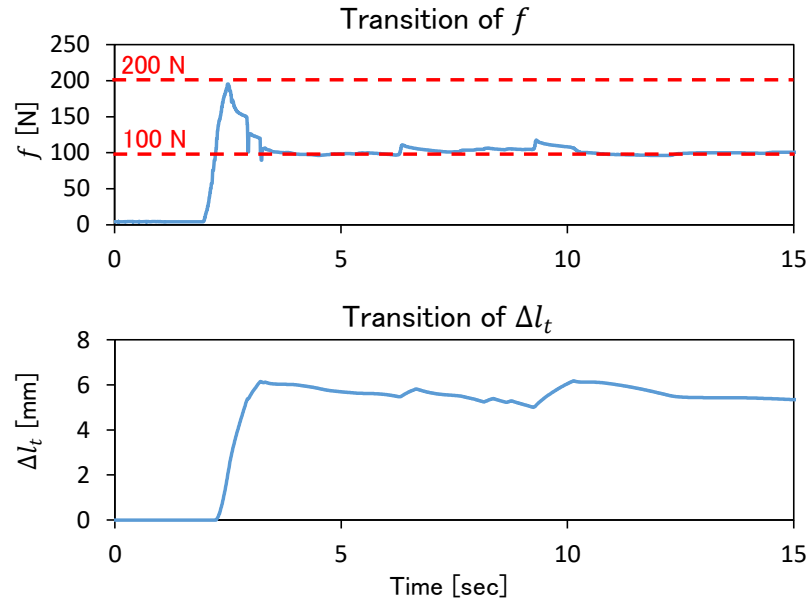


Fig. 7.32: The transition of f and Δl_t when contracting one muscle by 40 mm over 0.5 seconds with a safety mechanism [261].

安全機構

本研究では危険回避だけでなく、危険時にそれを緩和する安全機構も同時に備える。そして、この安全機構が作動したときを危険と見なし、危険度 p を定義する。それぞれの筋について、安全機構は以下のように筋張力に応じて筋長弛緩度 Δl_t を計算し、実機には $l^{ref} + \Delta l_t$ を指令する。

$$\begin{aligned}
 & \text{if } f > f_{thre} \\
 & \quad \Delta l_t = \Delta l_{t-1} + \min(C_{gain}d - \Delta l_{t-1}, C_{plus}d) \\
 & \text{else} \\
 & \quad \Delta l_t = \Delta l_{t-1} + \max(0 - \Delta l_{t-1}, -C_{minus}d) \\
 & \quad d = |f - f_{thre}|
 \end{aligned} \tag{7.23}$$

ここで、 f_{thre} は筋長を弛緩させ始める筋張力 f の閾値、 $|\bullet|$ は絶対値、 Δl_t はタイムステップ t における弛緩度、 $C_{\{minus, plus\}}$ はマイナス方向またはプラス方向に対する一ステップの筋長変化量を決める係数、 C_{gain} は最大弛緩量を決める係数である。つまり、 $C_{minus}d$ 、 $C_{plus}d$ で制限をかけながら、筋張力が最大値を越えないように筋を弛緩・緊張させている。本研究では、 $f_{thre} = 200$ [N]、 $C_{minus} = 0.001$ [mm/N]、 $C_{plus} = 0.003$ [mm/N]、 $C_{gain} = 2.0$ [mm/N] とし、本制御は 8 msec 周期で行う。

一本の筋を張った状態で末端を固定し、0.5 秒で -40 mm 緊張させたときの、安全機構の挙動を Fig. 7.32 に示す。また本実験のみ、安全のため $f_{thre} = 100$ [N] としている。筋張力は一瞬だけ 200 N まで上がり、その後 100 N まで抑えられている。このように、安全機構は高まった筋張力を瞬時に f_{thre} まで下げることができる一方で、そのピークは抑えられない。また、学習等のシステムがなければ、再び同じように危険な動作を何度も行ってしまう。

最後に、以下のように危険度 p を定義する。

$$p = \begin{cases} 1.0 & \Delta l_t > 0.0 \\ 0.0 & \text{otherwise} \end{cases} \tag{7.24}$$

つまり、安全機構が作動したとき、それを危険と見なし、それ以外は危険がないと見なす。これはあくまで一例に過ぎず、ユーザが危険と感じた動作を全く異なった形で定義することも可能である。

初期学習

DAN の初期学習を行う。DAN の重みが全くランダムな状態から始めることもできるが、オンライン学習におけるより速い収束を目指し、関節-筋空間マッピングを用いてネットワークを初期化する。

まず幾何モデル上でそれぞれの関節角度の下限と上限を決め、その値よりも少し広い範囲 (本研究では ± 10 deg) でランダムに関節角度の値を一様サンプリングする。このとき、一つでもサンプリングされた値が下限上限を超えた関節があれば、それは危険、つまり $p = 1.0$ とし、そうでなければ $p = 0.0$ とする。[125] で得られた関節と筋空間のマッピングを用いてその関節角度指令 θ^{ref} を筋長 l^{ref} に変換し (なお、筋張力は一律 10 N としている)、 (l^{ref}, p) のデータを蓄積する。このデータセットを用いて DAN を学習させる。なお、ノイズに強くするため、 l^{ref} には常に平均 0、標準偏差 C_{st} のガウシアンノイズを加える。本研究では $C_{st} = 3$ [mm] とし、データ数を 12000、バッチ数を 100、エポック数を 100、更新方法を Adam [46] として学習させた。

オンライン学習

DAN を実機データを使ってオンラインに更新していく。まず、実機に現在送られている指令筋長 $l^{ref, current}$ を取得する。これを現在のネットワークに入れたときに推測された危険度 p^{pred} を取得する。また同時に、Eq. 7.24 から p を取得し、現在の実機状態が危険状態であるかどうかを判定する。ここで、 $(p = 1.0)$ and $(p^{pred} < 0.9)$ または $(p = 0.0)$ and $(p^{pred} > 0.1)$ の場合に $(l^{ref, current}, p)$ をデータとして蓄積する。これは、予測と実際の危険度が大きく違う場合 ($(p = 1.0)$ and $(p^{pred} < 0.1)$ または $(p = 0.0)$ and $(p^{pred} > 0.9)$)、また、確率が曖昧である場合 ($0.1 < p < 0.9$) にデータを蓄積することに相当する。また、このデータ蓄積は、一つ前に蓄積された $l^{ref, pre}$ に対して、 $\|l^{ref, current} - l^{ref, pre}\|_2 > C_{diff}$ の場合のみ実行される ($\|\bullet\|_2$ は L2 ノルム、 C_{diff} は指令筋長差分の閾値とする)。つまり、ロボットがある程度動かない限りはデータは蓄積されない。データの最大数 N_{max} を決め、それより多いデータが蓄積された場合は最も古いデータを削除する。データ数が閾値 N_{thre} を超えたところから、新しいデータが得られる度に、蓄積された全データを使ってオンラインで DAN を更新する。

本研究では、 $C_{diff} = 20.0$ [mm]、 $N_{max} = 100$ 、 $N_{thre} = 30$ とする。また、オンライン学習の際のバッチ数は 10、エポック数は 3 とし、更新則は Momentum SGD を用いる。

アプリケーション

得られた DAN を用いて、実機に l^{ref} を指令する前に危険度 p^{pred} を予測し、動作を行うかどうかを決定することができる。これだけでなく、本章では DAN を用いた 2 つの応用について述べる。

まず、ある指令筋長 l^{ref} を送る際に、これをなるべく危険ではない筋長指令へと修正する方法について述べる。 l^{ref} から DAN を用いて p^{pred} を求めた時、これが $p^{pred} > 0.1$ であれば危険な可能性が

あるため、そのまま \mathbf{l}^{ref} を指令すべきではない。そこで、以下のように指令筋長 $\mathbf{l}^{ref, safe}$ を更新することで、安全な筋長指令 $\mathbf{l}^{ref, safe}$ を計算し、これを実機に送ることを考える。

$$L = \mathbf{h}_{loss}(\mathbf{l}^{ref, safe}) = |\mathbf{h}_{dan}(\mathbf{l}^{ref, safe})| + C_{loss} \|\mathbf{l}^{ref} - \mathbf{l}^{ref, safe}\|_2 \quad (7.25)$$

$$\mathbf{l}^{ref, safe} \leftarrow \mathbf{l}^{ref, safe} - \gamma \partial L / \partial \mathbf{l}^{ref, safe} \quad (7.26)$$

ここで、 C_{loss} は重み付けの定数、 L は損失、 γ は学習率とする。ここで、 $\mathbf{l}^{ref, safe}$ の初期値は現在の指令筋長である $\mathbf{l}^{ref, current}$ を用いる。つまり、 $\mathbf{l}^{ref, safe}$ は現在の指令筋長 $\mathbf{l}^{ref, current}$ から送るべき指令筋長 \mathbf{l}^{ref} に徐々に近づいていくが、危険度 p^{pred} が高くなると途中で更新が止まり、危険になる手前の指令筋長 $\mathbf{l}^{ref, safe}$ が得られることになる。 γ はある固定の値でも良いが、本研究では γ の最大値 γ^{max} とバッチ数 N_{batch} を決め、 γ^{max} 以下の N_{batch} 個の学習率を用いる。それぞれの学習率で $\mathbf{l}^{ref, safe}$ を更新したあともう一度 L を計算し、最も L が小さかったものを採用する。この処理を N_{iter} 回繰り返すことで、 $\mathbf{l}^{ref, safe}$ を計算する。本研究では $C_{loss} = 0.01$, $\gamma^{max} = 0.1$, $N_{batch} = 10$, $N_{iter} = 30$ とする。

次に、優先度付き逆運動学 [262] に DAN を利用することを考える。ある手先の姿勢 \mathbf{x}^{ref} に対して逆運動学を解き、指令関節角度 $\boldsymbol{\theta}^{ref}$ 、そのときの指令筋長 \mathbf{l}^{ref} を計算する。第一タスクとして $\mathbf{x}^{ref} = \mathbf{h}_{kinematics}(\boldsymbol{\theta}^{ref})$ を設定する ($\mathbf{h}_{kinematics}$ は関節角度を手先位置に変換する関数を表す)。このとき、得られた $\boldsymbol{\theta}^{ref}$ を [125] によって筋長に変換し、DAN によって危険度 p^{pred} を予測する。 $p^{pred} > 0.1$ のとき、その姿勢 $\boldsymbol{\theta}^{ref}$ は危険な可能性があるため、回避姿勢 $\boldsymbol{\theta}^{avoid}$ として設定する。次に、第二タスクとして、 $|\boldsymbol{\theta}^{ref} - \boldsymbol{\theta}^{avoid}| > d$ を設定して、優先度付き逆運動学 [262] を解く (d は閾値の定数を表す、本研究では 0.2 rad とする)。同様に p^{pred} を予測し、危険度が下がらなければこの $\boldsymbol{\theta}^{avoid}$ は増えていき、第二タスクの条件は増加していくことになる。また、実際には第一タスク、第二タスクはそれぞれ現在の関節角度の周りに線形近似され、優先度付き逆運動学 [262] において用いられる。

7.3.3 実験

本研究では筋骨格ヒューマノイド Musashi を実験に用いる [87]。筋配置は Fig. 4.24 のようになり、人体の主要な筋が模倣されている。本研究では主に肩 3 自由度と肘 2 自由度の 5 自由度を用いる。これらに関係する筋は 10 本であり、内二関節筋が一本含まれる。

オンライン学習実験

DAN のオンライン学習に関する実験を行う。関節角度の下限と上限の範囲内のランダムな関節角度を [125] により指令筋長に変換し、Fig. 7.33 のようにロボットに送り続ける。これと同時に安全機

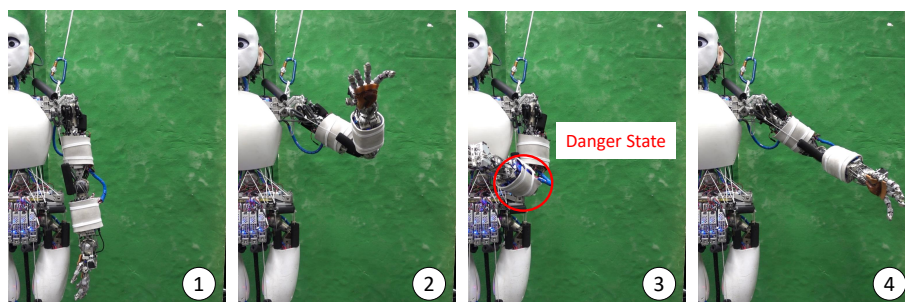


Fig. 7.33: Online learning experiment of Danger Avoidance Network [261].

構・DAN のオンライン学習を実行させたときの危険状態, データ蓄積の遷移を Fig. 7.34 に示す. 300 秒間で約 10 回程度危険が続く状態が起きている. データ蓄積は常に起きているが, 危険状態のときは比較的頻繁に起きている.

オンライン学習の効果を示すために, オンライン学習の 0, 100, 200, 300 秒後のモデルを使って, オンライン学習のときは異なる同様のランダムな動作を 300 sec 行う. このとき, 常に p^{pred} を計算して $p^{pred} > 0.1$ の場合を危険状態とし, これが現在の Eq. 7.24 から計算した危険状態と合致していた場合の頻度を Fig. 7.35 に示す. 学習するに従って徐々に合致の頻度が増えていくことが読み取れる. ここで, それぞれについて合致する確率を計算したものを Fig. 7.36 に示す. 学習が進むごとに確率は上がっていき, 300 秒後には 72% まで上がることがわかった. この後学習を続けても, 確率が大きく上がることはなく, 70% 付近を振動していた.

筋長修正制御実験

7.3.2 節で述べた, 安全な筋長指令への修正に関する実験を行う. 前述の学習で得られた 300 秒後のモデルを用いて, 先と同様にランダムに身体を動かす. このとき同じランダム動作において, 筋長修正手法を用いない場合 (l^{ref} を実機に指令する) と用いた場合 ($l^{ref, safe}$ を実機に指令する) を比較する. 筋#1 の l^{ref} と $l^{ref, safe}$ の遷移と, 危険状態の遷移を Fig. 7.37 に示す. 筋は 10 本あるため, 見やすいように一本だけ取り出している. 指令筋長の推移から, 筋長修正を用いることで, 筋長が増えている, つまり筋を伸ばしている箇所が複数見受けられる. それらの該当する箇所における筋長修正を用いない場合の危険状態の推移を見ると, ほとんどの箇所で危険状態が起きていることがわかる. これに対して, 筋長修正を用いた場合は, 該当する箇所の危険状態が全く起こっていない, または少しだけ起きていることがわかる. 危険状態の頻度は, 筋長修正を用いない場合は 15%, 筋長修正を用いる場合は 3% となった. つまり, 危険状態を正しく予測し, 危険状態が起らないように筋長を修正できて

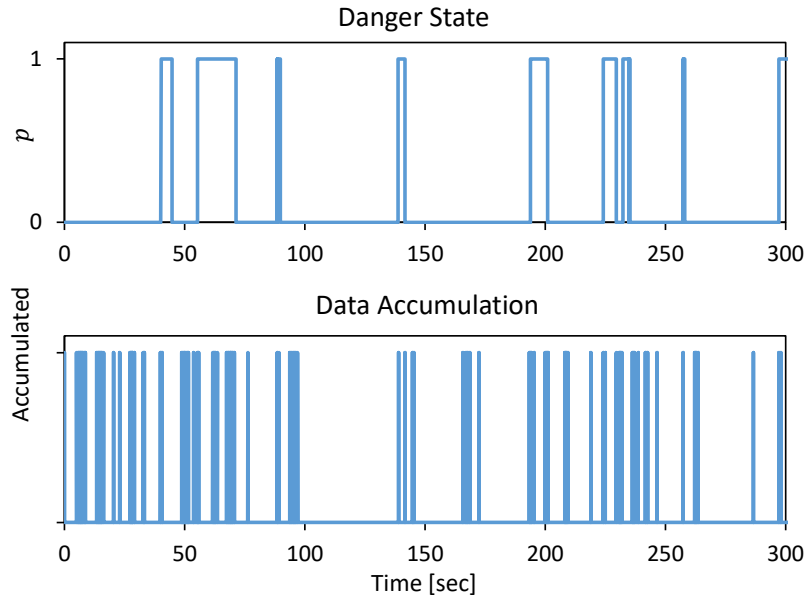


Fig. 7.34: Transition of danger state and data accumulation frequency when conducting online learning [261].

いることがわかる。一方、約 50 秒付近においては、筋長修正をしない場合とした場合とで、同様に危険状態が起きている。これは、オンライン学習実験において学習し切れなかった動作であると考えられる。

DAN の優先度付き逆運動学への応用実験

7.3.2 節で述べた、DAN を用いた優先度付き逆運動学に関する実験を行う。本研究では、 $f_{thre} = 150$ [N] としている。ある \mathbf{x}^{ref} を設定し、ある初期姿勢から逆運動学を解く。このときの危険度を予測し、危険な場合にはその姿勢を条件に入れ、再度逆運動学を解く。この一度目、二度目の姿勢を Fig. 7.38 の上図に示す。このときの予測された危険度と、実際の危険状態の遷移を Fig. 7.38 の下図に示す。一度目は 0.96 と危険が予測され、実際に動かした際にも危険状態が起きている。これに対して、二度目は危険度が 0.08 と予測され、実際に動かした際にも危険状態は起っていない。

7.3.4 議論

実験結果について議論を述べる。オンライン学習における実験では、学習時間が増えるごとに危険状態の予測精度が上がっていくことがわかった。同時に、その予測精度はある程度以上は上がらず、途

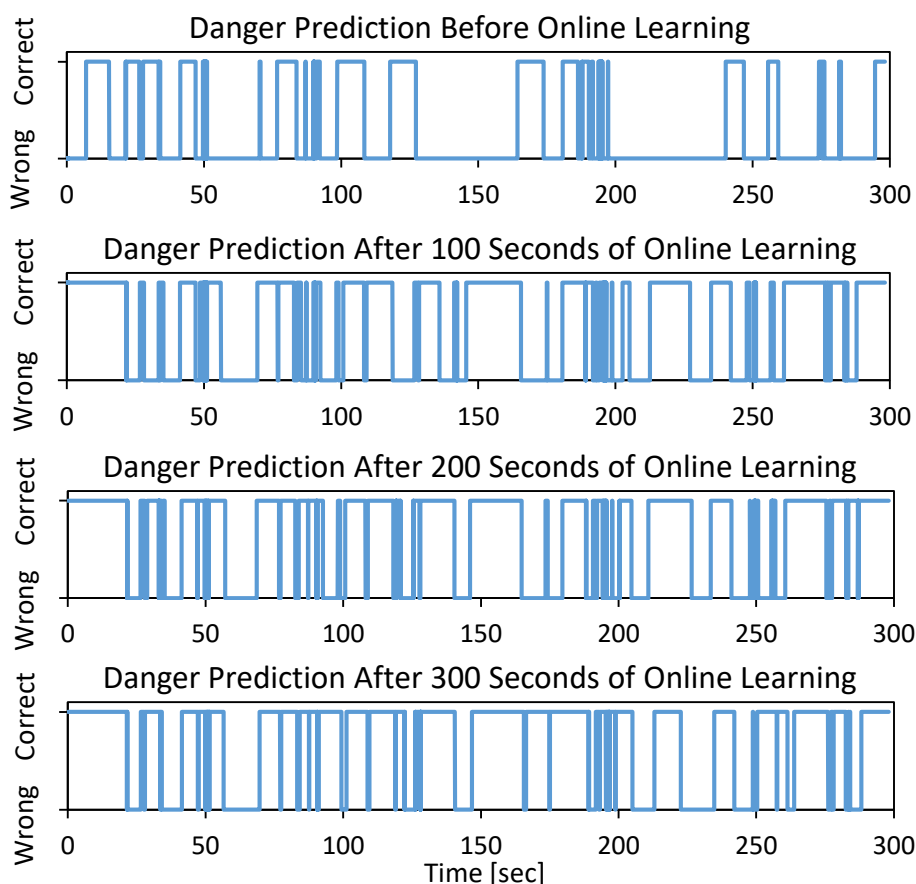


Fig. 7.35: Transition of danger prediction correctness after 0, 100, 200, and 300 seconds of online learning [261].

中で停滞することがわかった。これは、危険が起こるか起こらないかの境目における状態では予測が収束しないことが理由であると考えられる。また、安全な筋長修正においては、学習したネットワークを用いることで、正確に危険を回避できることがわかった。通常は指令筋長を変化させず、危険が予想される場合のみ筋長を少し緩め、危険状態を未然に防ぐことができる。最後に、優先度付き逆運動学を利用することで、危険でない姿勢でタスクを遂行できることがわかった。危険な姿勢を蓄積していき、それらから離れた姿勢の探索が可能となる。

本研究の問題点について述べる。まず、全身の干渉を行う場合は全身の筋の \mathbf{l}^{ref} をネットワークの入力としなければならない。しかし、入力の状態数が増えるほど学習は難しくなり、適切な筋のグルーピングが必要になる可能性がある。また、安全な筋長への修正はロボットの動きを変化させてしまうため、DAN を動作計画に組み込んでいくことが今後必要になると考える。

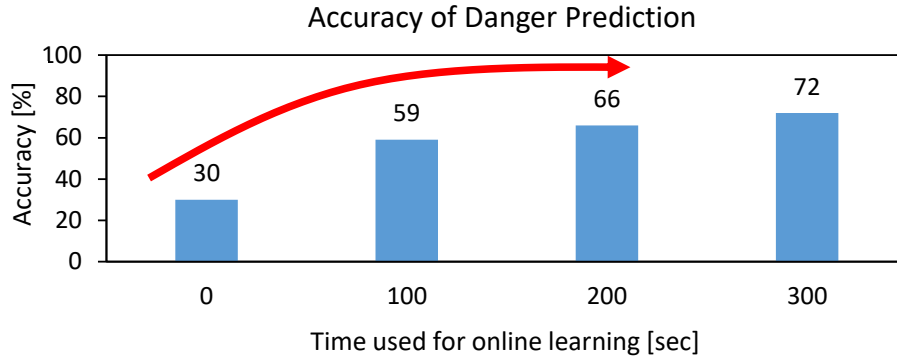


Fig. 7.36: The accuracy of danger prediction after 0, 100, 200, and 300 seconds of online learning [261].

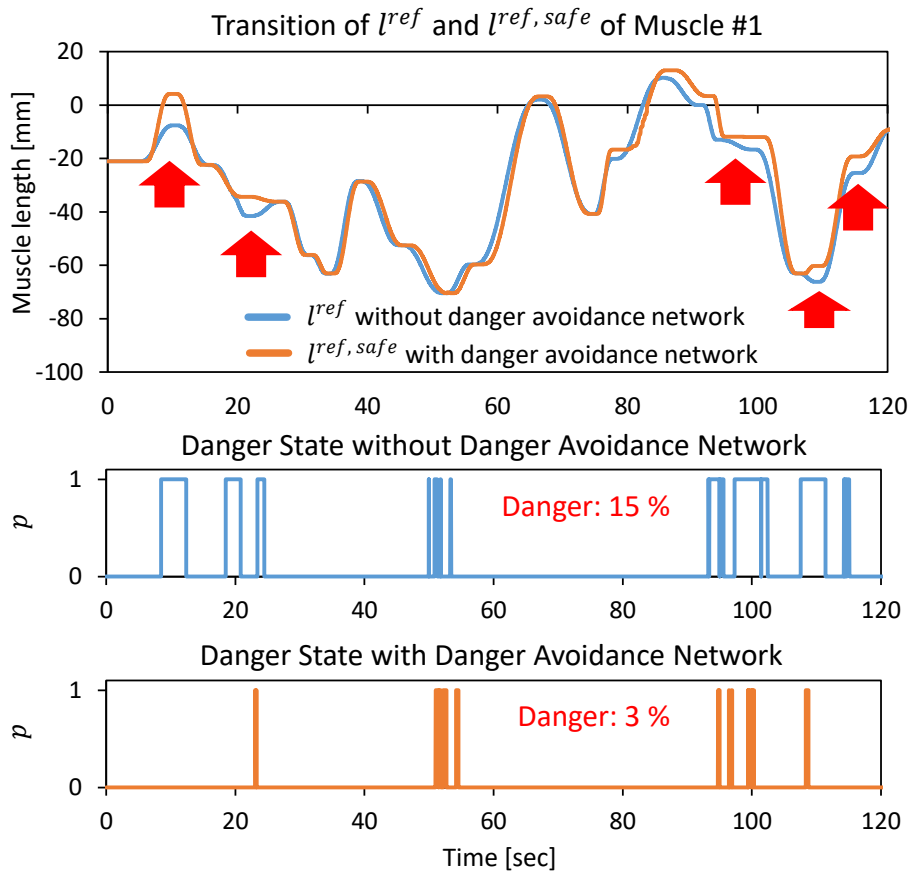


Fig. 7.37: The transition of target muscle length and danger state with and without muscle length modification using danger avoidance network [261].

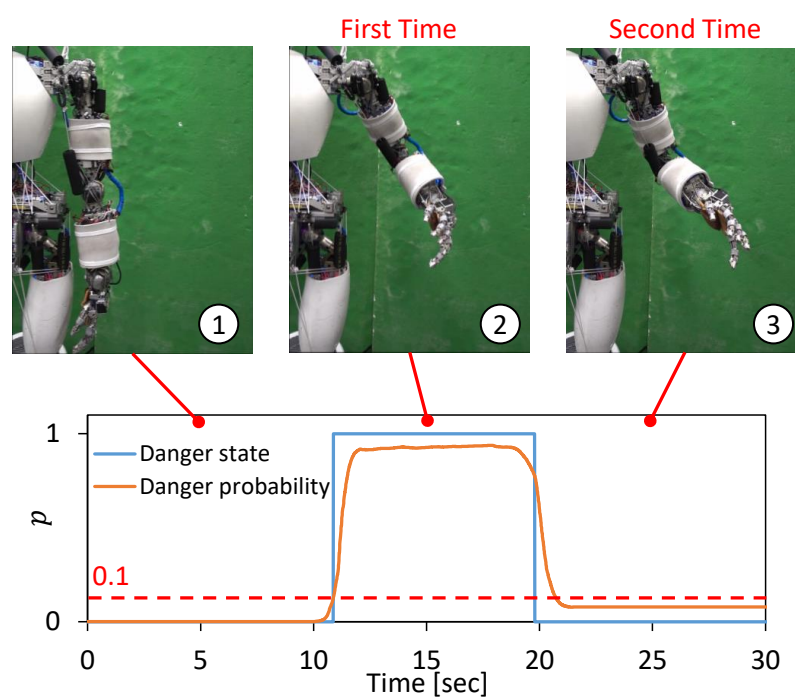


Fig. 7.38: The experiment of prioritized inverse kinematics using DAN [261]. The lower graph shows the transition of p and p^{pred} .

7.4 平均分散表現を含む動的な身体図式学習 - 環境適応型台車制御学習

7.4.1 概要と先行研究

これまで台車型 [151] や軸駆動型ヒューマノイド [179], 筋骨格ヒューマノイド [87] やソフトロボット [10] など, 様々なロボットが開発されてきている. その中でも, 柔軟な身体を持つロボットや, 部分的にしか状態を観測できずモデル化が困難なロボットにおいては, これまでの古典的な制御で扱うことが難しい.

これらの問題点は, (1) そもそもモデル化が困難であること, (2) そのモデルがロボットの身体状態や動作環境によって逐次的に変化してしまうこと, (3) モデルが確率的な挙動を示す場合があること, であるとする. (1) については現在広く周知されており, これまでの古典的なモデルベース制御ではなく, 深層学習型の様々なアプローチが開発されてきている. 学習型モデル予測制御 [24, 229] や強化学習 [50]・模倣学習 [231] 等がその例として挙げられる. (2) については, 一度学習した環境や身体の状態から外れたところでは意図したように制御が働かないという問題意識から, 様々なアプローチが成されている. 様々な環境で動作を行うことでどのような環境においても動作する制御則を得る方法 [71], オンライン学習により逐次的に身体や環境の変化をネットワークに取り込む方法 [263, 178], Parametric Bias を用いて暗黙的に身体や環境の情報をネットワークの一部に埋め込む方法 [146] 等がある. (3) については, 場合によって深刻な問題を引き起こすことがある. 台車型ロボット Fetch [151] を例に挙げてその問題を説明しよう. Fetch 台車部分の駆動輪と受動輪の関係を Fig. 7.39 に示す. この台車には2つの駆動輪と, 周りに4つのキャストが配置されており, このキャストが場合によって問題を引き起こす. キャスターの角度はその前の動きに依存し, その角度によっては後ろに進めなかったり, 確率的にキャストが回り, 進めたとしても意図していない方向に大きく動いてしまったりということが多い (古い型の Fetch に特徴的である). また, この挙動は床の材質によっても大きく変化する. よって, この問題を解決するには, 床等の環境によるモデルの変化を考慮しつつ, モデルに平均と分散の確率的な挙動を入れ, 動作の際にはその分散を最小化しつつ行動することが必要になる. 確率的挙動についてはこれまで, 強化学習の方策に確率的要素を入れた研究 [264], 人間のデモンストレーションのばらつきをネットワークに埋め込み模倣学習に利用する手法等が開発されている [265]. 一方, (1)(2)(3) 全ての要素を解決する予測モデル型の制御は開発されていない.

そこで本研究では, [146] や [265] の考えを取り込み, 確率的挙動を考慮可能かつ, 暗黙的に環境情報を変数として埋め込み環境適応が可能な予測モデル型のニューラルネットワークの構築, これを用いた分散最小化を含む環境適応型ロボット制御の開発を行う [98]. 本研究のネットワークは, 動的な一

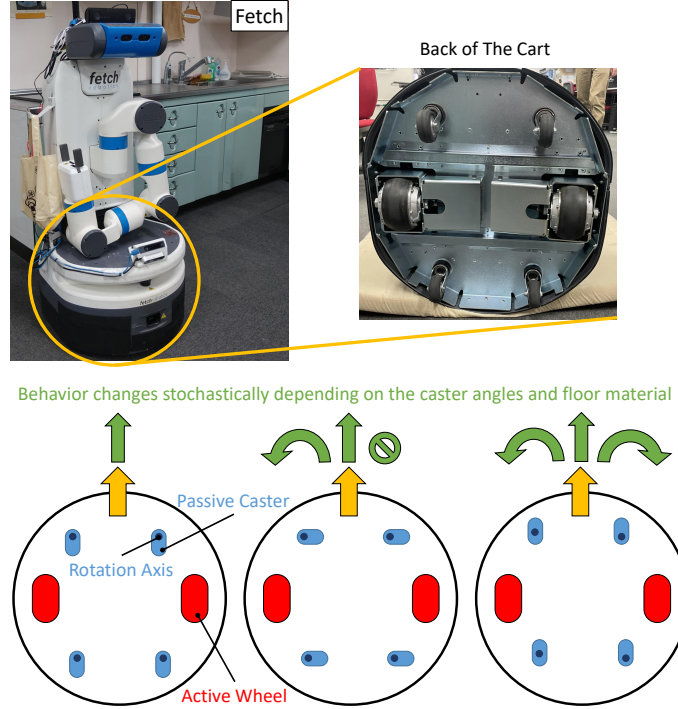


Fig. 7.39: The motion of the robot Fetch is stochastic depending on the angles of the casters and the friction of the floor [98].

般化多感覚相関モデルにおいて、出力にセンサ値の平均と分散を用いた形のネットワークを利用する。

7.4.2 SPNPB による分散最小化を含む環境適応型制御

本研究で提案するネットワークを Stochastic Predictive Network with Parametric Bias (SPNPB) と呼ぶ。SPNPB とそれを取り巻く全体システムを Fig. 7.40 に示す。

SPNPB のネットワーク構造

SPNPB は数式で以下のように表せる。

$$(s_{t+1}, v_{t+1}) = h(s_t, u_t, p) \quad (7.27)$$

ここで、 t は現在のタイムステップ、 s は画像や車輪速度等のロボット状態、 v はガウス分布を仮定した場合の s の分散、 u は制御入力、 p は Parametric Bias [72]、 h は SPNPB を表す。なお、 s, u の値は得られた全データを使って正規化されている。また、 v_{t+1} は分散を表し常に正の値を取るため、指数関数を通して出力される。

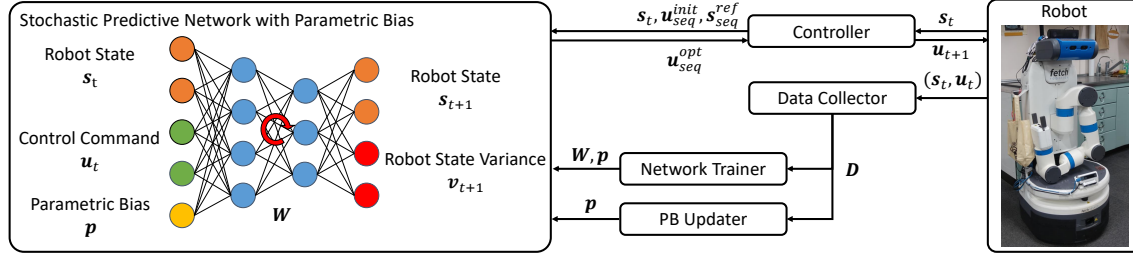


Fig. 7.40: The overall system of using stochastic predictive network with parametric bias (SPNPB) [98].

本研究において SPNPB は 10 層とし、順に 4 層の全結合層, 2 層の LSTM 層 [241], 4 層の全結合層からなる。ユニット数については, $\{N_u + N_s + N_p, 50, 20, 10, 10$ (LSTM のユニット数), 10 (LSTM のユニット数), 10, 20, 50, $2N_s\}$ とした (なお, $N_{\{u,s,p\}}$ は $\{u, s, p\}$ の次元数とする)。活性化関数は Tanh, 更新則は Adam [46] とした。また, Eq. 7.27 の実行周期は 5 Hz とする。

SPNPB の学習

ランダムな動作や人間の操縦等によって, s, u に関するデータを取得する。この際, 環境 (例えば床の材質やロボットの状態) を変化させながらデータを取得することで, それらを暗黙的な情報として Parametric Bias に埋め込むことができる。ある同一の環境において取得した一連の試行 k について, データ $D_k = \{(s_1^k, u_1^k), (s_2^k, u_2^k), \dots, (s_{T_k}^k, u_{T_k}^k)\}$ を得る ($1 \leq k \leq K$, K は全試行回数, T_k はその試行 k に関する動作ステップ数とする)。そして, 学習に用いるデータ $D_{train} = \{(D_1, p_1), (D_2, p_2), \dots, (D_K, p_K)\}$ を得る。 p_k はその試行 k に関する Parametric Bias であり, その試行中については共通の値で, 異なる試行については別の値となるような変数である。このデータ D_{train} と以下の損失関数を用いて SPNPB を学習させる。

$$P(s_{i,t}^k | D_{k,1:t-1}, W, p_k) = \frac{1}{\sqrt{2\pi\hat{v}_{i,t}}} \exp\left(-\frac{(\hat{s}_{i,t}^k - s_{i,t}^k)^2}{2\hat{v}_{i,t}}\right) \quad (7.28)$$

$$L_{likelihood}(W, p_{1:K} | D_{train}) = \prod_{k=1}^K \prod_{t=1}^{T_k} \prod_{i=1}^{N_s} P(s_{i,t}^k | D_{k,1:t-1}, W, p_k) \quad (7.29)$$

$$L_{train} = -\log(L_{likelihood}) \quad (7.30)$$

ここで, P は確率密度関数, $\{s, v\}_i$ は i 番目のセンサの $\{s, v\}$, $D_{k,1:t-1}$ は $[1:t-1]$ の間の D_k のデータ, W は SPNPB のネットワークの重み, $\{\hat{s}, \hat{v}\}$ はデータ $D_{k,1:t-1}$, 現在の重み W , 試行 k に関する現在の Parametric Bias p_k を使って SPNPB から予測された $\{s, v\}$ の値, $p_{1:K}$ は $1 \leq k \leq K$ の p_k をまとめたベクトルを表す。 $L_{likelihood}$ は D_{train} が与えられたときの W, p に関する尤度関数を表し, これを最

大化する問題を考える。このとき、Eq. 7.30 のように変換を行うことで $\log(P)$ の足し算へと計算を簡単にし、 L_{train} の最小化問題に落としこむことができる。通常であればネットワークの重み W のみを更新していくが、本研究では同時に p_k も更新していくことになる。なお、全 p_k は初期値を 0 として最適化される。

Parametric Bias のオンライン学習

Parametric Bias p をオンラインで更新し続けることで、環境の変化に適応することができる。ロボットが動いている間に、常に訓練時と同様な s, u のデータ D_{update} を取得しておく。この D_{update} のデータ数 N_{data}^{update} が閾値 N_{thre}^{update} を超えたところから p のオンライン学習を始める。 N_{max}^{update} を超えたデータは古いものから破棄する。損失関数は Eq. 7.28 – Eq. 7.30 と同じものを使用し、バッチ数とエポック数をそれぞれ $N_{batch}^{update}, N_{epoch}^{update}$ として学習を行う。この際、ネットワークの重み W は固定し、 p のみを更新する。

なお、本研究では $N_{batch}^{update} = N_{data}^{update}$, $N_{thre}^{update} = 10$, $N_{max}^{update} = 50$, $N_{epoch}^{update} = 1$, 更新則は Momentum SGD [139] として学習を行う。環境変化への適応速度は更新則の学習率により調節ができるが、大きいほど不安定になる。また、動作データ次第でその速度や正確性は変わりうる。

分散最小化制御

学習された SPNPB, 現在の環境に合わせて更新された p から、動きの分散を小さくしつつ指令したタスクを実行可能な制御を行うことができる。動きの分散が小さくなることで、不安定な動きを回避しながら目的を達成できるようになる。まず、最適化する制御入力 $u_{t:t+N_{seq}-1}^{opt}$ の初期値 $u_{t:t+N_{seq}-1}^{init}$ を決定する。ここで、 N_{seq} は考慮する制御入力の時系列の長さを表し、以降これらは $u_{seq}^{opt}, u_{seq}^{init}$ のように省略する。現在状態 s_t を取得し、以下の処理を繰り返すことで制御入力 u_{seq}^{opt} を更新していく。

$$L_{control} = \|s_{seq}^{ref} - \hat{s}_{seq}\|_2 + C_{variance} \|\hat{v}_{seq}\|_2 \quad (7.31)$$

$$u_{seq}^{opt} \leftarrow u_{seq}^{opt} - \gamma \partial L_{control} / \partial u_{seq}^{opt} \quad (7.32)$$

ここで、 s_{seq}^{ref} は $[t+1 : t+N_{seq}]$ の間の指令状態、 $\{\hat{s}, \hat{v}\}_{seq}$ は $[t+1 : t+N_{seq}]$ の間における、現在の SPNPB, s_t, u_{seq}^{opt} から予測された $\{s, v\}$ の値、 $C_{variance}$ は損失関数の重みである。Eq. 7.31 の右辺第一項は与えられた指令状態を満たすための損失、右辺第二項は動作の分散を小さくするための損失である。Eq. 7.32 は誤差逆伝播と最急降下法を用いて u_{seq}^{opt} を更新する。このとき、 u_{seq}^{init} は、前ステップ $t-1$ で最適化された値 $u_{t-1:t+N_{seq}-2}^{prev}$ について、 u_{t-1}^{prev} を削除し $u_{t+N_{seq}-2}^{prev}$ を複製したものとした。このよう

に前ステップで最適化された値を初期値とすることで、より速い収束を得ることができる。また、 γ は固定値でも良いが、本研究では $[0, \gamma^{max}]$ を対数的に等間隔に分けた $N_{batch}^{control}$ 個の γ を用いて \mathbf{u}_{seq}^{opt} をそれぞれ更新し、Eq. 7.31 を実行した際に最も損失の小さかった \mathbf{u}_{seq}^{opt} を利用することを繰り返すことでより速い収束を得る。Eq. 7.31 – Eq. 7.32 を現在のステップ t において $N_{epoch}^{control}$ 回行い、最終的に得られた \mathbf{u}_{seq}^{opt} を実機へ指令する。

なお、本研究では $N_{seq} = 10$, $N_{batch}^{control} = 10$, $N_{epoch}^{control} = 3$, $\gamma^{max} = 3.0$ とした。 $C_{variance}$ は経験から設定され、大きいほどより分散を考慮できる一方、動作は不安定になりやすい。

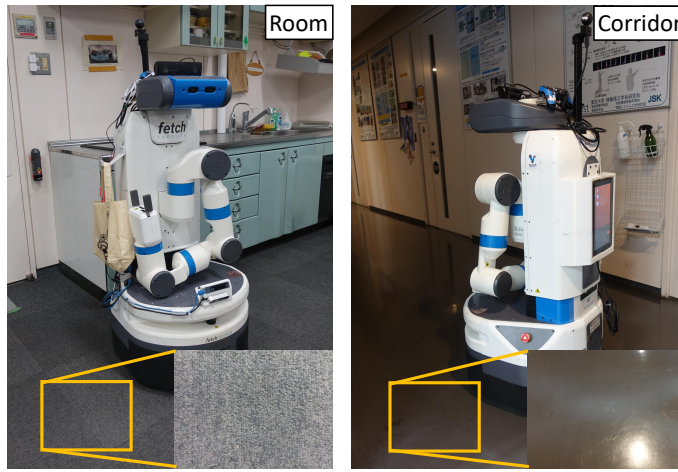


Fig. 7.41: Two types of floor materials, Room and Corridor, for Fetch experiment [98].

7.4.3 実験

実験セットアップ

台車のシミュレーション実験・Fetch を使った実機実験の2つを行う。どちらの実験においても、 $\mathbf{s}^T = (w_{trans} \ w_{rot})$, $\mathbf{u}^T = (w_{trans}^{ref} \ w_{rot}^{ref})$ とする (なお、 $w_{\{trans, rot\}}$ は並進・回転方向に測定された速度 (単位は $\{[m/s], [rad/s]\}$), $w_{\{trans, rot\}}^{ref}$ は並進・回転方向への指令速度を表す). $\mathbf{w} = (w_{trans} \ w_{rot})^T$, $\mathbf{w}^{ref} = (w_{trans}^{ref} \ w_{rot}^{ref})^T$ と表現する場合もある。主に並進・回転速度に関する最適化のみ行い、本研究では軌道については考えない。ここで、Eq. 7.27 には表していないが、実際には3.2節の実験で得られたマスク変数 \mathbf{m} が入力として存在している。また、全ての実験で \mathbf{p} は2次元とした。 \mathbf{p} はできるだけ小さな方が組織化されやすい。本研究の実機実験では2種類の床しか用いないため1次元で十分であるが、本研究では問題にならなかったため2次元とした。

シミュレーションでは, 簡易的に以下のように制御入力に対して動作出力に分散が加わった系を考える.

$$w_{trans} \leftarrow w_{trans} + \alpha(w_{trans}^{ref} - w_{trans}) + \beta \mathbf{N}(0, \frac{1}{|w_{trans}| + |w_{rot}| + 0.1}) \quad (7.33)$$

$$w_{rot} \leftarrow w_{rot} + \alpha(w_{rot}^{ref} - w_{rot}) + \beta \mathbf{N}(0, 0.1) \quad (7.34)$$

ここで, α はフィードバックの割合を表す係数, β はノイズの大きさを表す係数, $\mathbf{N}(\mu, \sigma)$ は平均 μ , 分散 σ の正規分布ノイズを表す. 回転に関するノイズは小さい一方で, 並進に関しては並進または回転の速度がある程度大きくないと, 動作に大きな分散がのってしまう系になっている. α, β の変化はモータの動きや床の摩擦等, ロボット身体や動作環境の状態の変化を簡易的に模擬している.

Fetch を使った実験では, Fig. 7.41 にある Corridor, Room の2つの環境で動作し, データを取得した. Corridor はツルツルと滑るような床なのに対して, Room はカーペットが敷かれているため非常に摩擦が強い床になっている. $w_{\{trans, rot\}}$ は Fetch の台車前方についた Intel RealSense T265 の Visual Odometry から得た.

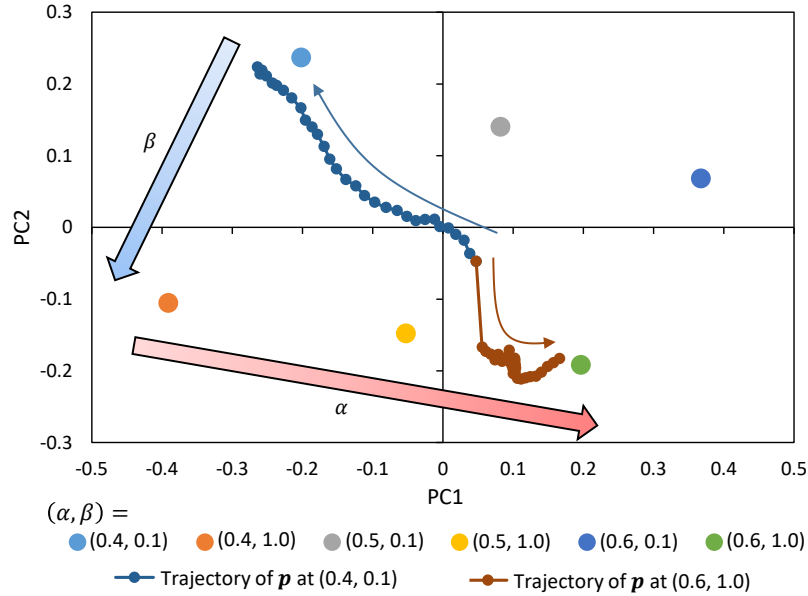


Fig. 7.42: Training experiment in simulation [98]: parametric biases p_k trained using the collected data and the trajectory of p updated by online learning.

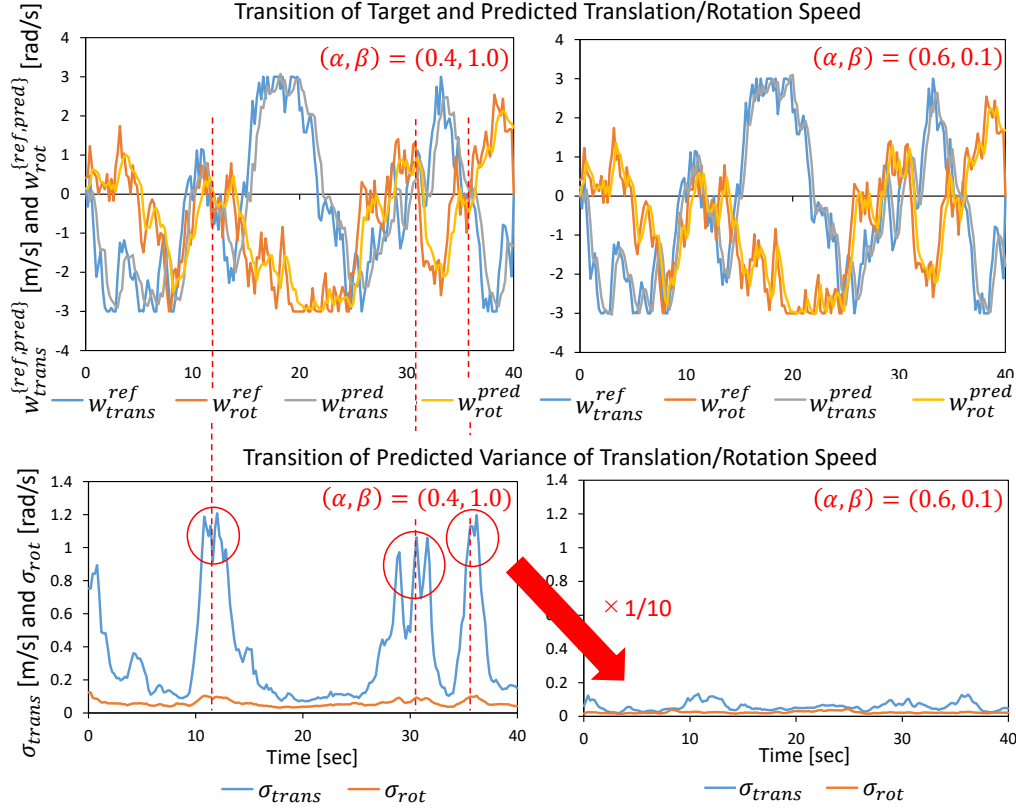


Fig. 7.43: State estimation experiment in simulation [98]: transition of the target velocity \mathbf{w}^{ref} , the predicted velocity \mathbf{w}^{pred} from SPNPB, and the predicted standard deviation σ of \mathbf{w}^{pred} .

シミュレーション実験

シミュレータを $\mathbf{w}^{ref} \leftarrow \max(-3, \min(\mathbf{w}^{ref} + \text{Random}(-1, 1), 3))$ のようなランダムな制御入力で動かす (なお, $\text{Random}(a, b)$ は $[a, b]$ の範囲内のランダムな値を返す関数とする). $\alpha = \{0.4, 0.5, 0.6\}$, $\beta = \{0.1, 1.0\}$ の 6 つの組み合わせについてそれぞれ約 200 ステップのデータを取得する. 得られたデータ D_{train} から SPNPB を学習した際の Parametric Bias \mathbf{p}_k を Principal Component Analysis (PCA) に通し 2 次元平面に表現した結果を Fig. 7.42 に示す. α, β の大きさに応じて \mathbf{p}_k が綺麗に配置されていることがわかる.

次に, Parametric Bias \mathbf{p} のオンライン学習について述べる. シミュレータを $(\alpha, \beta) = (0.4, 0.1)$ または $(\alpha, \beta) = (0.6, 1.0)$ の 2 種類に設定し, 上述のようにランダムに \mathbf{w}^{ref} を変化させた際のデータからオンラインで \mathbf{p} を更新したときの \mathbf{p} の挙動を Fig. 7.42 に示す. なお, 初期は $\mathbf{p} = \mathbf{0}$ としている (PCA により変換されるため, $\mathbf{p} = \mathbf{0}$ が原点にあるとは限らない). 学習が進むごとに \mathbf{p} は, 訓練時に得られた \mathbf{p}_k

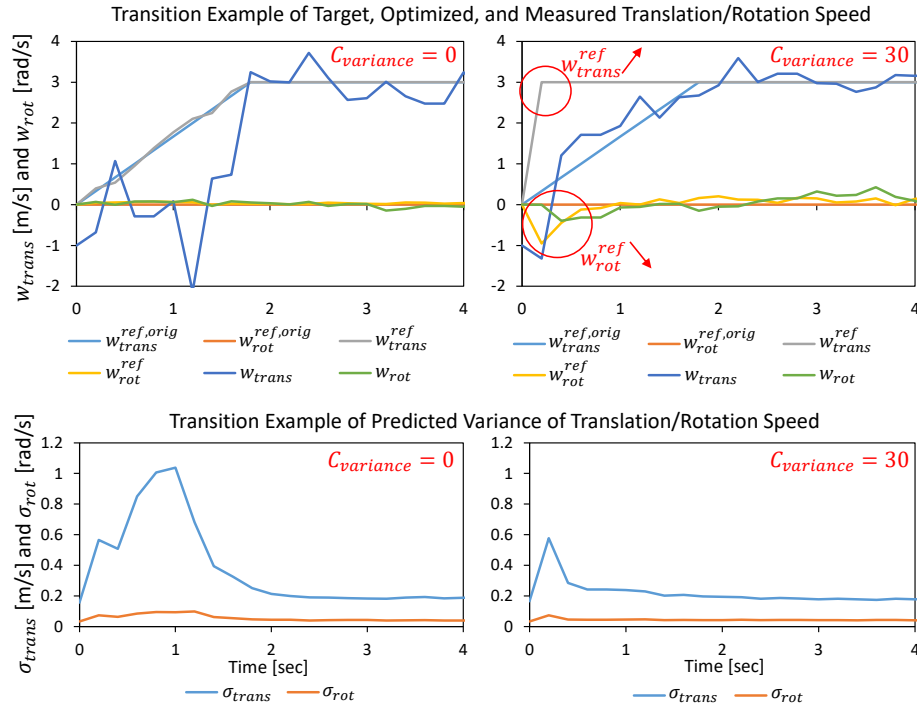


Fig. 7.44: Control experiment in simulation [98]: transition example of original target velocity $\mathbf{w}^{ref,orig}$, optimized velocity \mathbf{w}^{ref} , and measured velocity \mathbf{w} . transition of predicted standard deviation σ in the middle graphs, and ten transitions of w_{trans} and its average in the lower graphs, when setting $C_{variance} = \{0, 30\}$.

の中でも現在のシミュレーションのパラメータの \mathbf{p}_k に近づいていき、動きのダイナミクスから現在の状態を把握できていることがわかる。

次に、上述のようにランダムに動いた際の \mathbf{w}^{ref} から、SPNPB によってどのような \mathbf{w} の予測平均 \mathbf{w}^{pred} とその分散 \mathbf{v} が出力されるかを検証する。 \mathbf{p} を $(\alpha, \beta) = (0.4, 1.0)$ のデータから得られた \mathbf{p}_k , $(\alpha, \beta) = (0.6, 0.1)$ のデータから得られた \mathbf{p}_k の 2 種類に設定した際に、SPNPB から予測される $\mathbf{w}_{\{trans, rot\}}^{pred}$, $\sigma_{\{trans, rot\}}$ ($\sigma = \sqrt{\mathbf{v}}$) の推移を Fig. 7.43 に示す。 $\alpha = 0.6$ のときは $\alpha = 0.4$ のときに比べ、 \mathbf{w}^{pred} がより \mathbf{w}^{ref} に近く、速く収束していることが SPNPB により表現できている。 $(\alpha, \beta) = (0.4, 1.0)$ のときを見ると、 $|\mathbf{w}_{trans}^{pred}| + |\mathbf{w}_{rot}^{pred}|$ が小さいときに σ_{trans} は大きくなっており、Eq. 7.33 の特性と一致している。 また、 $\beta = 0.1$ のときは $\beta = 1.0$ のときに比べて σ の値が約 1/10 になっており、動作の分散についても SPNPB により正しく表現できている。

最後に、分散最小化を含む制御について検証する。シミュレータを $(\alpha, \beta) = (0.5, 1.0)$ に固定し、 \mathbf{p} も $(\alpha, \beta) = (0.5, 1.0)$ のデータから得られた \mathbf{p}_k に固定する。本実験では分散最小化制御における指令値 \mathbf{s}^{ref} を $\mathbf{w}^{ref,orig}$, 最適化されて最終的にロボットに送られる指令値 \mathbf{u}^{opt} を \mathbf{w}^{ref} とする。ここ

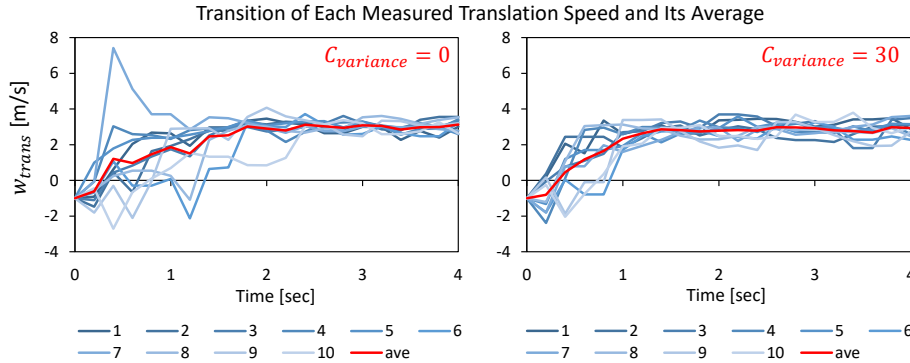


Fig. 7.45: Control experiment in simulation [98]: transition of predicted standard deviation σ in the middle graphs, and ten transitions of w_{trans} and its average in the lower graphs, when setting $C_{variance} = \{0, 30\}$.

で、ロボットが $\mathbf{w} = \begin{pmatrix} -1 & 0 \end{pmatrix}^T$ の状態から指令値として $\mathbf{w}^{ref,orig} = \begin{pmatrix} 3 & 0 \end{pmatrix}^T$ を送る場合 (なお、最初の 2 秒間は $\mathbf{w}^{ref,orig} = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$ から線形補間して指令値を送る) について、分散最小化制御を行い、ロボットの状態 \mathbf{w} の挙動について確認する。なお、分散を考慮しない $C_{variance} = 0$ 、分散を考慮する $C_{variance} = 30$ の 2 つの場合についてそれぞれ 10 回実験して検証している。元の指令値 $\mathbf{w}^{ref,orig}$ 、最適化された指令値 \mathbf{w}^{ref} 、得られたロボット状態 \mathbf{w} の遷移の一例を Fig. 7.44 の上図に示す。また、その際に SPNPB から予測される σ を Fig. 7.44 の下図に示す。 $C_{variance} = 0$ の際は、一切分散が考慮されないため、本研究の \mathbf{s}, \mathbf{u} の定義では、 $\mathbf{w}^{ref} = \mathbf{w}^{ref,orig}$ となっていることがわかる。この場合、 σ_{trans} の値が約 1.0 まで上がり、 w_{trans} がうまく w_{trans}^{ref} に追従せず、動作が不安定になっていることがわかる。これに対して、 $C_{variance} = 30$ の際は、 \mathbf{w}^{ref} は $\mathbf{w}^{ref,orig}$ には追従せず、 w_{trans}^{ref} が一気に大きく上昇し、 w_{rot}^{ref} も 0 で一定ではなく動作初期に変化していることがわかる。これにより、 $|w_{trans}| + |w_{rot}|$ が 0 に近い不安定な状態を回避し、比較的安定して w_{trans} が変化していることがわかる。この動作を 10 回行った際の w_{trans} の遷移とその平均を Fig. 7.45 に示す。全体として、 $C_{variance} = 0$ のときは動作初期に不安定な動作が散見されるが、 $C_{variance} = 30$ のときはそれを回避できていると言える。一方、平均を取ると $C_{variance} = 0$ のときは w_{trans} が $w_{trans}^{ref,orig}$ に良く追従しており、 $C_{variance} = 30$ のときは指令値遷移に対してより速く $w_{trans}^{ref,orig} = 3$ に到達してしまっており、不安定な動作を回避する代わりに精度を犠牲にしているとも取れる。

実機実験

人間がジョイスティックを使って様々な動きをさせてデータ D_{train} を取得する。得られたデータ D_{train} から SPNPB を学習した際の Parametric Bias \mathbf{p}_k を PCA を通して 2 次元平面に表現した結果を

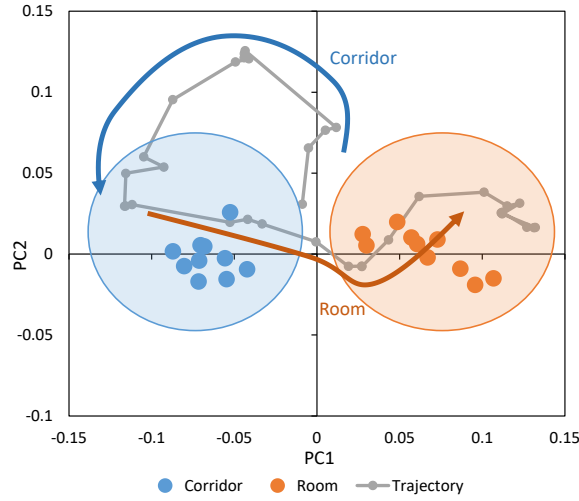


Fig. 7.46: Training experiment in Fetch [98]: parametric biases p_k trained using the collected data and the trajectory of p updated by online learning.

Fig. 7.46 に示す. Room と Corridor で p_k が綺麗に左と右に分けられていることがわかる. また, $p = 0$ の状態から, Corridor, Room の順にジョイスティックを使って動作させたときの, オンライン学習による p の遷移を Fig. 7.46 に示す. p は始め Corridor において得られた p_k の周辺に動き, その後 Room に場所を移動すると p も徐々に Room において得られた p_k 周辺に移動していくことがわかる.

これまで動作させた中で最も指令値通りに動かない動作である, ロボットを一回転させた後に後ろに進むという動作について検証を行う. 一回転させるとキャスターが能動輪に対して垂直に向き, その状態で後ろに進むと, 動作がスタックしてしまうことが多い. なお, 前進についてはサスペンションの関係が容易に進むことができる. 本実験は Room で行い, Room において更新した p を使う場合について $C_{\text{variance}} = \{0, 0.3\}$ を, Corridor において更新した p を使う場合について $C_{\text{variance}} = 0.3$ の場合の 3 種類について比較する. 本実験ではより安定的に動作させるために, Eq. 7.31 について, $\|\mathbf{w}^{\text{ref}, \text{orig}} - \mathbf{w}^{\text{ref}}\|_2$ という項を足している. また, \mathbf{w} が大きいほど必然的に分散 \mathbf{v} は大きくなると考え, Eq. 7.31 の右辺第二項を $\|\hat{\mathbf{v}}_{\text{seq}} \oslash \hat{\mathbf{s}}_{\text{seq}}\|_2$ のように書き換えている. 一回転 (右回転に統一する) した後, 並進方向について -1 m/s を指令値として送るが, 安全装置により現在の \mathbf{w} に応じて徐々に指令値 $\mathbf{w}^{\text{ref}, \text{orig}}$ は上がっていく. それぞれの条件における, 代表的な動作の一例について $\mathbf{w}^{\text{ref}, \text{orig}}$, \mathbf{w}^{ref} , \mathbf{w} の遷移を Fig. 7.47 の上図に示す. $C_{\text{variance}} = 0$ の場合には, $\mathbf{w}^{\text{ref}, \text{orig}}$ と \mathbf{w}^{ref} の間に大きな違いはないが, w_{trans} は 0 のままで完全に動作がスタックしていることがわかる. それに対して, $C_{\text{variance}} = 0.3$ かつ Room の p を使った場合は $\mathbf{w}^{\text{ref}, \text{orig}}$ と \mathbf{w}^{ref} が大きく異なり, w_{trans} も 1.5 秒後程度から動き始めていることがわかる. この条件に特徴的なのは, w_{trans} が最初前方に動くことでキャスターの向

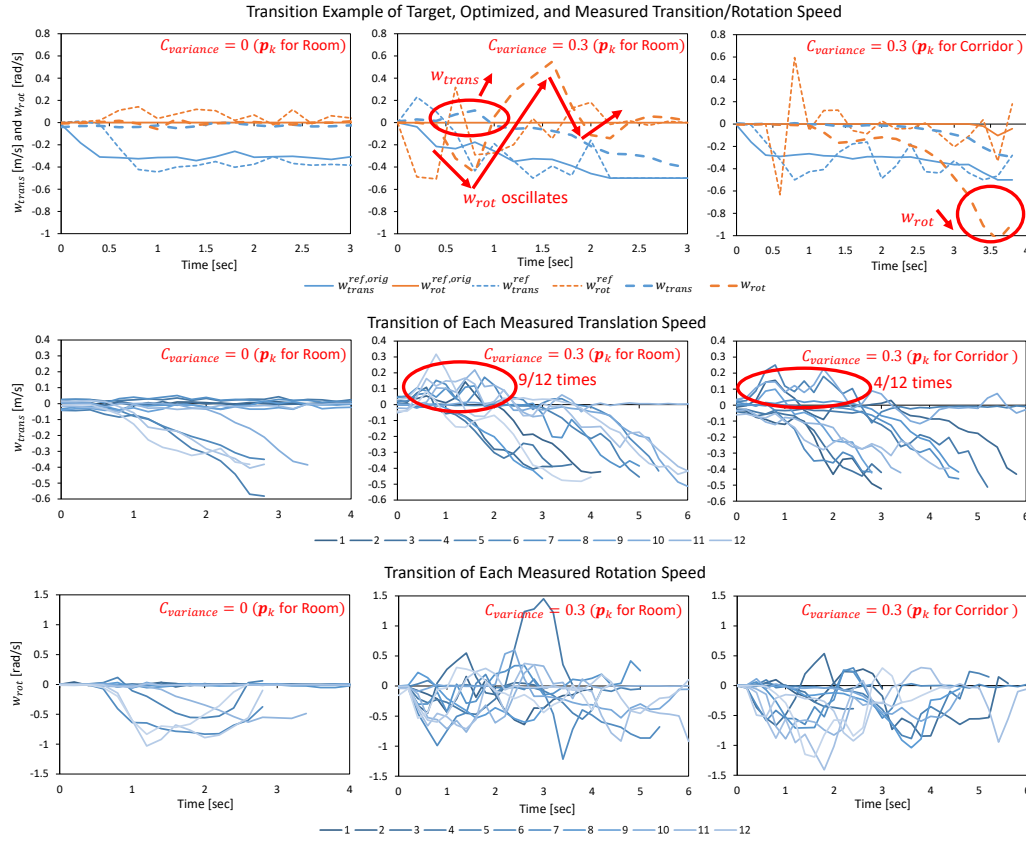


Fig. 7.47: Control experiment in Fetch [98]: transition example of original target velocity $\mathbf{w}^{ref,orig}$, optimized velocity \mathbf{w}^{ref} , and measured velocity \mathbf{w} in the upper graphs, and twelve transitions of $\mathbf{w}_{\{trans,rot\}}$ in the middle and lower graphs, when setting $C_{variance} = 0$ (\mathbf{p}_k for Room), $C_{variance} = 0.3$ (\mathbf{p}_k for Room), and $C_{variance} = 0.3$ (\mathbf{p}_k for Corridor).

きを変化させること,そして \mathbf{w}_{rot} を振動させることでスタックしないように動かしている点である。 $C_{variance} = 0.3$ かつ Corridor の \mathbf{p} を使った場合も同様に \mathbf{w}_{trans} を動作させることができています。この条件では, Room の \mathbf{p} を使った場合に比べ, \mathbf{w}_{trans} を前方に動かさず,直接後方に対して指令値を送る動作が多い。また, \mathbf{w}_{rot} が前の回転時の動作と同じ方向に大きく動く,つまり回転しながら後方に下がる場合が多い。12回の試行について $\mathbf{w}_{\{trans,rot\}}$ を Fig. 7.47 の中図・下図に示す。 \mathbf{w}_{trans} を見ると, $C_{variance} = 0$ の場合は半分以上の試行で動作がスタックしているのに対して, $C_{variance} = 0.3$ の場合は1,2回の試行を除いて後進方向への動作に成功している。なお, $C_{variance} = 0.3$ で一度前進してから後進するケースは Room の \mathbf{p} を使った場合は9/12回, Corridor の \mathbf{p} を使った場合は4/12回であった。 \mathbf{w}_{rot} を見ると,基本的に後進前の回転(右回転のため $\mathbf{w}_{rot} < 0$)と同じ方向に \mathbf{w}_{rot} が動いてしまっ

いることがわかる。これに対して, $C_{variance} = 0.3$ かつ Room の p を使った場合は w_{rot} が 0 付近で振動したような挙動になっており, 比較的真っ直ぐ進む。

7.4.4 議論

本研究の実験結果について考察する。まず, シミュレーション実験では確率的な振る舞いをする簡単な系において, 系のパラメータを明示的に与えなくても, 得られた動作データから Parametric Bias によってその際のダイナミクスが自己組織化可能なことがわかった。また, 動作データからオンラインで Parametric Bias を更新することで, 現在の系の状態を認識することができる。 p にはその系の特徴が詰まっており, SPNPB による推論から, 狙った通りの予測動作遷移, 分散が出力されることがわかった。これらを用いて分散最小化を含んだ学習制御を行うことで, 分散が高く不安定な動作を避けながらタスクを実現できることがわかった。次に, Fetch における実験では, 実機においても, 床の摩擦やテクスチャ等の暗黙的なパラメータを Parametric Bias によって自己組織化可能なことがわかった。オンライン学習により床の材質が変わったことを検知し, 自身の動作する環境を認識することができる。最後に, これまでよく動作がスタックしてしまった回転後の後進に本研究を適用し, スタックしないような動作が最適化により生成されることがわかった。直接後進すると動作がスタックするのに対して, 一度前進してキャストの向きを変化させ, 回転方向に動き続けることでスタックフリーな動作が生成された。また, これは p を変化させることで振る舞いに変化する。Corridor における p を使用した場合は, 回転しながら無理やり後進を続ける動作が散見された。Corridor は摩擦の少ない床であるため一度前進しなくてもスタックしにくいという特徴が影響したためだと考えられる。本実験では適切ではない p を利用しても大きく性能に差は出なかったが, 他の動作では影響が出る場合も考えられる。

本研究の問題点は大きく 2 つあると考える。一つ目は, 最適化に時間がかかるため, 現状では実行周期をあまり短く設定できないことにある。通常であれば $w^{ref} = w^{ref, orig}$ であるため瞬時に応答ができるが, w^{ref} を最適化するのににかかった時間だけ応答が遅れてしまう (本節では 200 msec)。この問題の解決策として, 最適化された w^{ref} を一度の順伝播で出力するような別のネットワークを作ることが考えられる。二つ目は, 動作の確率的な振る舞いを正規分布としてモデル化した点である。本実験ケースも厳密には正規分布で近似できるようなものではなく, 2 種類の動きから確率的に動作が選ばれる等, 別の系では適用できなくなる可能性がある。

7.5 身体図式の自動分割 - 筋骨格ヒューマノイドにおける筋分割

7.5.1 概要と先行研究

冗長なセンサ・アクチュエータが全身に分布するロボットにおいて、それら全てを用いて一つの制御器やニューラルネットワークを構築することは計算量・複雑性の観点から難しい。ゆえに、全センサ・アクチュエータを統合して使用する強化学習 [266] や過学習を起こしやすいオンライン学習 [178] は現在でも難しい。同時に、全身に分布するセンサ・アクチュエータは、指や腰、足等の部位ごとに機能が分かれやすいという特徴もある。そのため、全身のセンサ・アクチュエータを同時に用いる必要のあるタスクもある一方、関係が深いセンサ・アクチュエータをグルーピングし、それらのグループに対してそれぞれ制御器やネットワークを構築することが有効な場合も多い (Fig. 7.48)。グルーピングを施すことで、解釈性・扱いやすさが向上し、並列でそれぞれの学習器をオンライン学習させることも可能となる。本節では、センサ・アクチュエータ同士の機能的な接続と空間的な接続をグラフ構造に落とし込み、自動的にグルーピングする手法を開発する [267]。本手法を冗長な筋を持つ筋骨格ヒューマノイドに適用しその有効性を示す。

人体の構造だけでなく駆動方法さえも模倣した筋骨格ヒューマノイドには、人間と同様な多数の冗長な筋が存在する。この冗長性は重要な要素であり、筋が一本切れても動き続けるロバスト性や [123]、非線形弾性要素を併用することによる可変剛性制御 [125] を可能とする。同時に、全身に分布する多数の冗長な筋を一つの制御器や一つのニューラルネットワークによって管理し動かすことは、計算量や複雑性の観点から非常に困難である。

そのため、これまでの制御手法・状態推定手法では、関係性が希薄な部位ごとに筋群を適切に分割し、それぞれに対して制御器やニューラルネットワークを構築してきた。[125, 178] では、適切にグルーピングした筋アクチュエータ・センサ群それぞれにおいてニューラルネットワークを構築し、制御や状態推定・シミュレーションを実行している。適切なグルーピングによって関わるアクチュエータ・センサを少数に限ることで、小さな計算量でオンライン学習を行うことにも成功している。[123] でも同様にトルクコントローラを構築している。[268] では多関節筋がある場合における筋分割の目安を提示し、これを元に正確な関節角度推定を実行する方法について考察している。また、大抵の制御器は全身ではなく腕の一部 [226, 146] 等のみに適用されており、全身に関する適用についてはあまり議論されていない。

全身に配置された多数の冗長な筋群をグルーピングするときに助けとなる情報には、機能的接続と空間的接続がある。機能的接続とは、筋は冗長であるがゆえにそれぞれが機能的に関係しあっており、

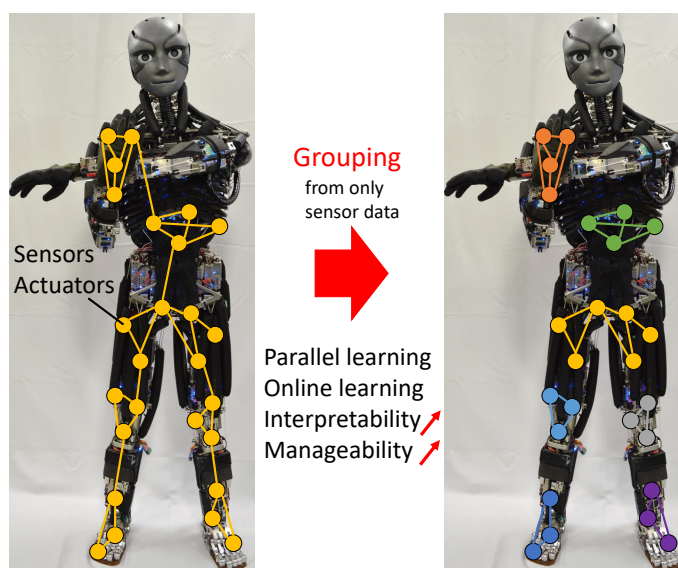


Fig. 7.48: The concept of this study [267].

その相関の強さを表す (e.g. 主動筋と拮抗筋の関係性は強い). 空間的接続とは, 神経的な接続から来る筋同士の空間的な近さを表す (e.g. 足の筋と腕の筋の空間的な接続は弱い). この2つの情報をグラフ構造に埋め込み, グラフ分割を行うことで, 適切な筋グルーピングを自動的に行う. 本手法を適用することにより, ロボットは自身のランダムな動きから機能的接続を見出し, これと空間的接続を合わせることで適切な筋グルーピングを行うことができる. また, 計算量や複雑性を下げつつも正確性を保った解釈性が高く扱いやすい制御器をグルーピングごとに自動的に作り上げていくことが可能となる.

これまで, これらグルーピングは人間が考えて行うものであったため, それをセンサデータのみから自動的に決定する本研究の問題設定は今までになく新しい. また, 筋骨格系に限っては人体の EMG を模倣する方法も考えられるが, 本研究では様々なロボットに同様に適用できるよう, 筋骨格系に特別なアルゴリズム実装は行わない. 本手法を筋骨格ヒューマノイド Musashi, Kengoro に適用し, これまで幾何モデルから人間が行ってきた筋グルーピングとの比較, グルーピング後の学習性能の比較を行う.

7.5.2 機能的接続と空間的接続を利用したセンサ・アクチュエータの自動分割

本研究では, ある冗長なセンサやアクチュエータの値を \mathbf{x} とし, その次元を N_x とする. ただし, それぞれのデータ間におけるスケールの違いを無くすため, 得られた全データを使って正規化を施すこととする. この \mathbf{x} , そして以降で説明する潜在変数 \mathbf{z} を合わせた変数群に重み付きの無向辺を張り, 乱

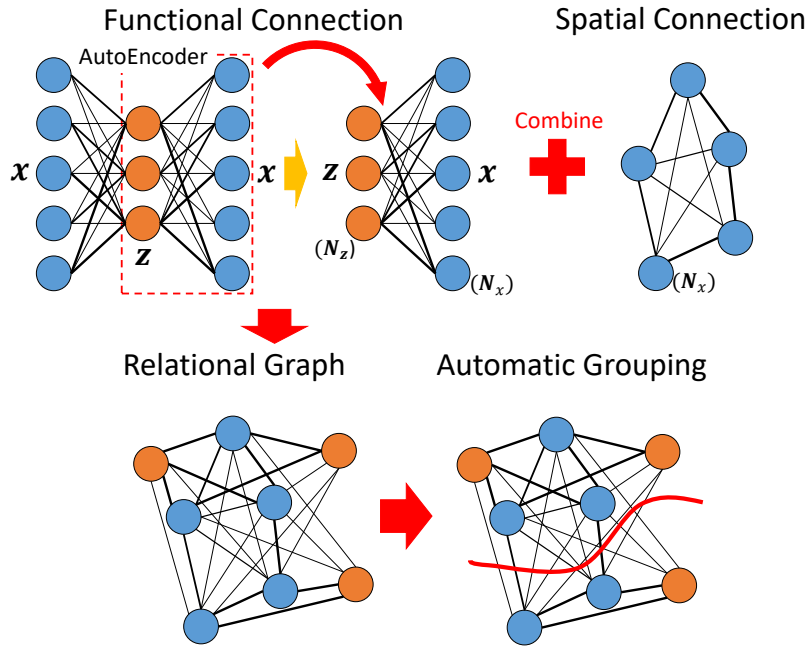


Fig. 7.49: The overall flow of automatic grouping using a relational graph with functional and spatial connections [267].

択アルゴリズムによって自動分割する方法について説明する。全体のフローを Fig. 7.49 に示す。

機能的接続による関係グラフ構築

機能的な接続とは、 x が冗長であるがゆえに、何らかの潜在変数 z (次元数は N_z , $N_z < N_x$) が存在し、 x の間の関係は z によって表されることを指す。つまり、 x の機能が z によって表現可能である。

この機能的な接続を学習させることは AutoEncoder [227] を用いることで可能である。入力を x 、ボトルネックとなる中間層を z 、出力を x とするような AutoEncoder を学習させることで、 x の機能的な接続を z を介して計算することができる。もしこの AutoEncoder が3層である場合は、2層と3層の間の重み W ($N_z \times N_x$) が、グラフ構造において z と x をノードとしたときの、ノード間の重み付き無向辺となる。これは、 z 同士、 x 同士の間には辺がないような二部グラフとなっている。もし AutoEncoder が5層のように直接 W を取り出せないような構造であったとしても、3層目と4層目の間の重みを W_1 、4層と5層の間の重みを W_2 として、 $W = W_1 W_2$ という形で W を計算することが可能である。 W の中の接続で、値が大きなものほどより機能的な結びつきが強く、同じグルーピングに分けられやすい。

空間的接続による関係グラフ構築

空間的な接続とは, x 間について, 空間的な近さや, 同じ回路に接続しているなどの制約を表す. 空間的に近い値ほど, 同じグルーピングに分けられやすいことを表現するために, この空間的な接続を x 同士の間での重み付きの辺としてグラフ内に埋め込む. このとき, 機能的接続における辺の重みが大きいほど同じグルーピングになりやすいという評価と一致するように, 空間的接続が近いほど辺の重みが大きくなるように値を設定する必要がある. 空間的な距離が d と表せたとすると, 本研究では辺の重みを $-d$ として埋め込む.

関係グラフを使った自動分割

本研究におけるグルーピングは, 辺の重みが小さい, つまり関係が希薄な辺を切って, 辺の重みが大きい, つまり関係が濃い頂点同士をまとめていくことに相当する. この問題を解く最も有力な方法として, 最小カットのアルゴリズム, 最小全域木のアルゴリズムが考えられる. 最小カットについては, source と sink の定義が必要ない [269] 等のアルゴリズムを用いることが可能である. また, 最小全域木については, Kruskal 法 [270] において所望のグルーピング数になるまで頂点をマージすることで複数グルーピングにも適用可能である. しかし, これらのアルゴリズムを本研究の機能的接続のみを含む関係グラフに適用したところ, 頂点 1 個と頂点 $N_x - 1$ 個等の極端な解が最適回となることがほとんどであり, それぞれのグルーピングが大体同じだけの頂点を含むようなバランスの良いグルーピングを作ることができなかった. これは, AutoEncoder によって得られた重みについて, 関係がない辺の重みが正確に 0 になるとは限らず, それぞれある程度の重みを持っており, 綺麗に分割ができないことが原因であると考えられる.

本研究では, 以降に示すようなグルーピング間の頂点数についての制約等を考慮したアルゴリズムを適用する (Fig. 7.50). 得られたグラフに対して, それぞれのノードについてグループのラベルを振り分ける. 本研究では, 分割したいグルーピング数を N_g とし, 一つのグループに含まれる最小の z の頂点数を N_z^{min} , 最小の x の頂点数を N_x^{min} として制約を設定する. また, グラフの全頂点の集合を V , x に含まれる頂点の集合を V_x , z に含まれる頂点の集合を V_z , 全グループの集合を G , 全エッジを E , 頂点 $v \in V$ から出るエッジを E_v , 頂点 $v \in V$ から集合 $g \in G$ に含まれる頂点への辺の集合を E_v^g , 頂点 v が含まれる集合 $g \in G$ における x に含まれる頂点の数を N_x^v , 頂点 v が含まれる集合 $g \in G$ における z に含まれる頂点の数を N_z^v とする. 擬似コードを Alg. 2 に示す.

ここで, N_{iter} は乱択アルゴリズムのイテレーション回数を表す. また, $InitializeGroup(V)$ は V のそれぞれの頂点にランダムにグループのラベルを振り分ける操作である. $RandomlyChoose(V)$ は V に

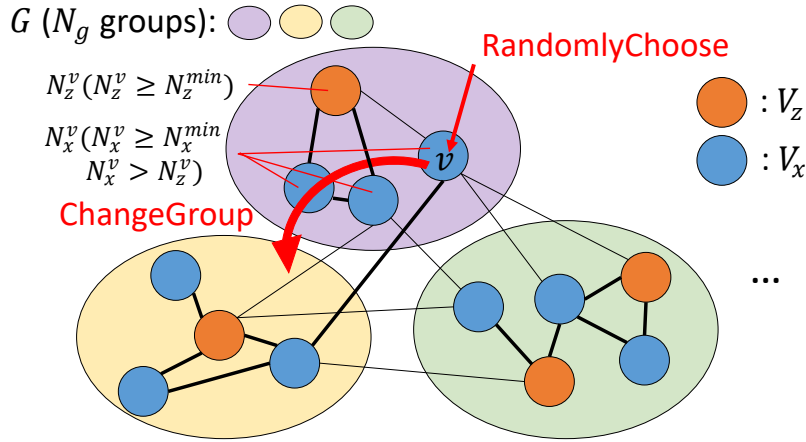


Fig. 7.50: Conceptual diagram of automatic grouping [267].

Table 7.5: Notations in this grouping method [267].

Notation	Definition
V	set of all vertices of the relational graph
$V_{\{x,z\}}$	set of vertices included in $\{x, z\}$
G	set of all groups
E	set of all edges of the relational graph
E_v	set of edges from the vertex $v \in V$
E_v^g	set of edges from the vertex $v \in V$ to the vertices in $g \in G$
$N_{\{x,z\}}^v$	the number of vertices of $\{x, z\}$ included in the group $g \in G$ to which the vertex $v \in V$ belongs

含まれる頂点をランダムに一つ選ぶ操作である。InitializeEval(G) はそれぞれのグルーピングに対する評価値のベクトルを初期化して返す関数である。SetEval(S, g, s) はグルーピング g に関する S の値を s にセットする操作である。CalcEval(E_v^g) は E_v^g に含まれる機能的・または空間的な接続を表す辺の重みから得られる評価値を返す関数である。SortByEval(S) は S を評価値の降順に並べ変える操作である。ChangeGroup(v, S, n) は頂点 v の属するグループを S の評価値が最も高いグループに変更する操作である (n については後に述べる)。

本アルゴリズムは、まずそれぞれの頂点が属するグループを適当に選択・初期化し、その中からランダムな頂点を選び、その頂点を現在のグループから評価関数をもとにいずれかのグループ(現在のグループを含む)に変更するという単純なアルゴリズムである。このとき、Line 6–10 は一つのグループに含まれる x, z の頂点の最小値制約、また、潜在変数 z の頂点の方が x の頂点数よりも少ないという制約を満たすための条件である。また、Line 12–17 は v を G に含まれるそれぞれのグループに

Algorithm 2 Automatic grouping method [267].

```

1: function GROUPING
2:   InitializeGroup( $V$ )
3:    $n_{iter} \leftarrow 0$ 
4:   while  $n_{iter} < N_{iter}$  do
5:      $v \leftarrow \text{RandomlyChoose}(V)$ 
6:     if  $v \in V_x$  AND ( $N_x^v \leq N_x^{min}$  OR  $N_x^v \leq N_z^v + 1$ ) then
7:       Continue
8:     end if
9:     if  $v \in V_z$  AND  $N_z^v \leq N_z^{min}$  then
10:      Continue
11:    end if
12:     $S \leftarrow \text{InitializeEval}(G)$ 
13:    for  $g \in G$  do
14:       $s \leftarrow \text{CalcEval}(E_v^g)$ 
15:      SetEval( $S, g, s$ )
16:    end for
17:    SortByEval( $S$ )
18:     $n \leftarrow n_{iter} / N_{iter}$ 
19:    ChangeGroup( $v, S, n$ )
20:     $n_{iter} \leftarrow n_{iter} + 1$ 
21:  end while
22: end function

```

属するように変更した場合の評価値を計算し、どのグループに属することで評価値が最も高くなるかを計算する. Line 14 では、以下のように機能的接続と空間的接続それぞれについて評価値を計算し合計する.

$$\text{CalcEval}(E_v^g) = 1/N_{Func} \Sigma w^{Func} + \alpha \Sigma w^{Spac} \quad (7.35)$$

ここで、 $w^{\{Func, Spac\}}$ は E_v^g の辺における機能的または空間的接続に関する辺の重みを表し、 N_{Func} は E_v^g に含まれる機能的接続辺の数、 α は評価値の重みを表す定数である. 機能的接続は先に述べたように、AutoEncoder で学習されるため互いに関係がなくても重みが 0 になるわけではない. よって、 E_v^g に含まれる機能的接続辺の重みの合計を評価値とすると、最も頂点数が多いグループへの辺が増え、それらに引っ張られてしまうため、重みの平均を評価値として用いる. また、空間的接続辺の重みは先に述べたように負であり、辺が増えるほど評価値が下がるため、平均を用いる必要はなく、重みの合計を評価値として用いる. 空間的接続辺の重みを $1/d$ のように正の値で表し統一的に扱うことも可能であ

るが、本研究では正しく動作しなかった。7.5.4 節において機能的接続、または空間的接続のみを利用する場合は、一方の評価値のみ計算することになる。最後に Line 19 だが、ここで頂点 v のグルーピングを変更する際、 $n = n_{iter}/N_{iter}$ によってグループ変更の挙動を変化させる。常に最も評価値が高いグループに変更させてしまうと、即座に局所解に陥ってしまうため、焼きなまし法のように、 n が 1 に近づくほど最大の評価値のグループが選ばれ、 n が 0 に近いほどランダムにグループが選ばれるようにする。つまり、 n の確率で最大評価値のグループを選び、 $1 - n$ の確率でランダムにグループを選ぶ。

本研究では、 $N_x^{min} = 2$, $N_z^{min} = 1$, $N_{iter} = 30000$, $\alpha = 10$ とする。Eq. 7.35 右辺第一項の値は大きく、Eq. 7.35 の右辺第二項の値は 0 に近い負の値となるため、 α は > 1 となるように設定することで二つの評価値をバランスさせることができる。

7.5.3 筋骨格ヒューマノイドにおける筋分割

本研究では、7.5.2 節において提案した手法を筋骨格ヒューマノイドにおける筋分割に適用する。筋骨格ヒューマノイドの構造、筋における機能的・空間的接続とは何かについて説明し、7.5.4 節においてその有効性を実験により示す。本研究では関節や筋の配置に関する事前知識がない状態において、如何に筋をグルーピングするかについて議論する。

筋肉の機能的接続

筋骨格構造における筋長 l を 7.5.2 節における x として本手法を適用する。この筋長 l は冗長な構造をしており、ある方向に関節を動かす時、その関節周りに必ず動きを担う主動筋と動きを妨げる拮抗筋が存在する。そのため、これらの筋長 l は 7.5.2 節の機能的接続に見るように潜在変数 z によって表現可能である (l を x とする場合は θ を z と同様に用いることが可能であるが、本研究は関節に関する前提知識はなく、実機においても基本的に関節角度は得られないため、本研究では潜在変数 z として扱う)。説明したように筋には多関節筋があり、そのため複数のグルーピングに筋がまたがる場合も存在する。本研究ではこれをいずれか一方のグループに所属させるが、グルーピング後にそれぞれの筋について機能的接続を見て複数グループに所属させることも可能である。筋のランダムな動きから l のデータを取得し、これを使って AutoEncoder を学習させる。

筋肉の空間的接続

人間において空間的に近い筋ほど神経的に接続が強くなるという考えを、空間的接続として取り込む。筋配置等の事前知識はないため、本来であれば回路的に遅延が少ない、つまり空間的に近いという

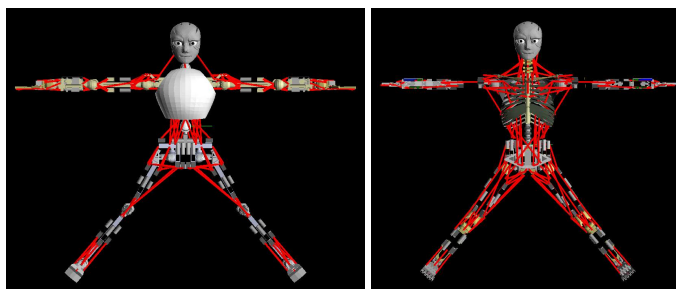


Fig. 7.51: The posture of Musashi (left) and Kengoro (right) when calculating spatial connections in this study [267].

指標をロボットにおいては持ち込むべきであるが、現状ではそれは難しい。そのため、筋配置の情報は直接には使わないが、幾何モデルにおけるそれぞれの筋経路の中心を計算し、それぞれの筋がそれぞれの筋に対してどの程度遠いか、を表すマトリックスのみ用いる。Fig. 7.51 のように、手足を広げた状態において、このときの空間的な距離を近似的に空間的接続の距離として用いることとする。つまり、それぞれの筋同士の距離を d として、辺の重みは $-\beta d$ となり、全 x 同士に対してこの辺を張る。 β は前述の AutoEncoder から得られた W と βd の平均を揃えるための係数であり、得られた全辺の重みから自動的に決まる。

7.5.4 実験

実験セットアップ

本実験で用いる筋骨格ヒューマノイドは Musashi [87] と Kengoro[86] である。その筋配置を Fig. 7.52, Fig. 7.53 に示す。Musashi は 74 本, Kengoro は 116 本の筋を持つ。足 (toe) と手 (hand) を除き, Musashi は多関節筋が 4 本と少ないのに対して, Kengoro は 26 本と、より人体に似た筋配置が成されている。また, Fig. 7.52, Fig. 7.53 において筋が色分けされているが、これらは、これまで [125, 178] 等で使われてきた musashi, kengoro の筋のグルーピングを表す。どちらもグルーピング数は 14 であり、一つの筋に 2 つの色分けが成されている場合は、どちらのグループにもまたがる多関節筋であるという意味である。これは、関節を選び、その関節に寄与する筋を全て選択していくという方法により作成可能であり、本実験の正解データとする。この事前に関節と筋配置に関する幾何モデルが得られる状態で作成された筋グルーピング (Geometric) に対して、本実験では、関節と筋配置の幾何モデルを前提としない状態で、ロボットが自身のランダムな筋の動きからこのグルーピングを行っていく (Proposed)。Geometric と Proposed により得られた筋グルーピングの一致率、また、分割前と後における学習効率・正確性の

観点から議論を進める.

ここで, Geometric と Proposed によるグルーピングの一致率を算出する場合があるが, これは幾何モデルから得られたグルーピング一つずつに対して, それと一致するグルーピングが生成されたかどうかの割合を表し, 完全に一致する割合を A_0 , 1つ異なる場合を許した割合を A_1 , 2つ異なる場合を許した割合を A_2 で表す ($0 \leq A_{\{0,1,2\}} \leq 100$). 2つのグルーピングにまたがる筋については, どちらのグループに属しても良いこととする.

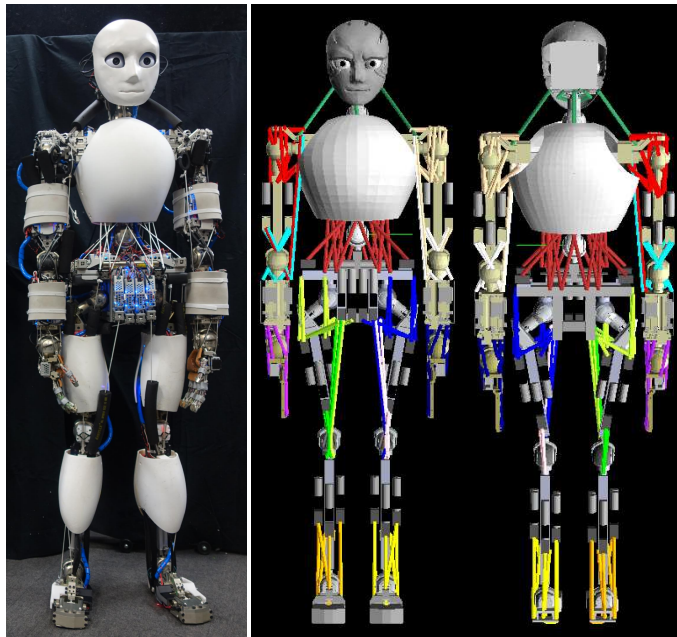


Fig. 7.52: The musculoskeletal humanoid Musashi and its correct muscle grouping when using its geometric model (Geometric) [267].

シミュレーションによる評価実験

本節におけるシミュレーションとは, 人間が作った, 筋経路を筋の始点・中継点・終点を直線で結んだ幾何モデルを用いることを指す. 幾何モデルを関節角度範囲内でランダムな姿勢にしたときの全筋の筋長 l を並べた 100000 のデータと, 7.5.3 節における空間的接続の情報の 2 つのみを元に筋分割を行う. ここで, 筋のまとまりがはっきりしている場合には空間的接続の情報は強すぎる条件となってしまうため, 本研究では筋同士の距離 d に対して, 平均 0, 分散 100 mm のノイズを加えている. l のデータは, 7.5.3 節に述べたように AutoEncoder を学習することで機能的接続に変換する. 本研究では, この AutoEncoder は 5 層とし, ユニット数は順に $M, 300, N_z, 300, M$ (ここで M は筋の数とする), 活性

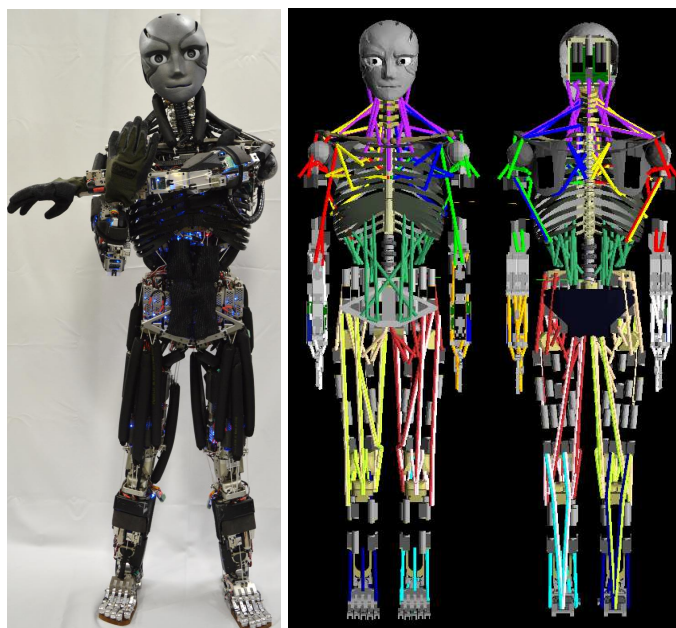


Fig. 7.53: The musculoskeletal humanoid Kengoro and its correct muscle grouping when using its geometric model (Geometric) [267].

化関数は Tanh, 最終層以外のそれぞれの層について Batch Normalization [45] を適用している. N_z については, いくつかの値を試し比較を行う. また, データは 80%を訓練用, 20%をテスト用に分け, バッチ数を 100, エポック数を 300, 更新則を Adam [46] として学習を行い, 最もテストの値が小さいモデルを使用する. 機能的接続のみを用いた場合 (Func), 空間的接続のみを用いた場合 (Spac), 両者を用いた場合 (Both) のグルーピング性能について, Musashi, Kengoro の両者それぞれについて比較を行う.

Musashi における分割の例を Fig. 7.54 に示す. ここでは $N_z = 40$ としており, 順に Func, Spac, Both における分割の全体像とその詳細を示す. それぞれのグルーピングの色については, 初期値が異なるため同じグループでも異なる色となることがある. Func, Spac, Both それぞれについて, 概ね Fig. 7.52 と同じようなグルーピングが成されていた. Func では, 図に示した (1), (2) のように, 膝と腰, 右手と左手のような, 比較的遠い別であるべき場所のグループ同士が合体して同じグループに属してしまうことがあった. これに対して, Spac については, 図に示した (3), (4) のように首と肩や右と左の股関節のような, 空間的には近いが機能的には干渉が無く別のグループになるべき場所において, 一本だけ間違ったグルーピングが成されていることが多い. Both については高い確率で Geometric と同じグルーピングが作成されたが, (5) のように一部別のグループに筋が飛んでしまうことがあった. このグルーピングを 10 回行った際における, A_0, A_1, A_2 の平均と分散を Fig. 7.55 に示す. Fig. 7.55 から分か

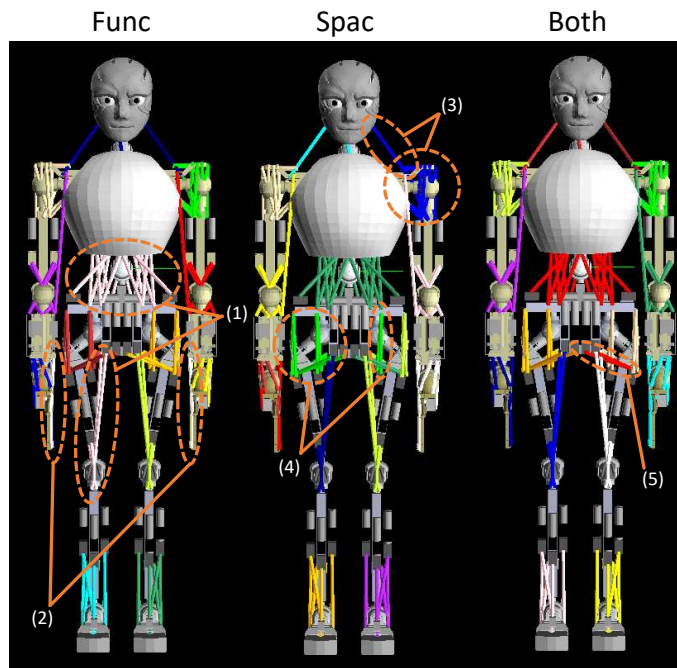


Fig. 7.54: Examples of muscle grouping when conducting the proposed grouping method of Func, Spac, or Both for Musashi (Proposed) [267].

るように、 A_0, A_1, A_2 の全てにおいて機能的接続と空間的接続を用いる Both によるグルーピングが最も Geometric と一致していた。Func については A_0, A_1, A_2 について一致率にはほとんど変化がないのに対して、Spac については A_0 から A_2 になるに従って大きく一致率が向上していた。これは、Func は離れたグループにまたがって属する複数の筋を持つため、間違いを数本許容しても意味がないのに対して、Spac では一本や二本だけ間違っ筋が近いグルーピングに属してしまうことが多かったためだと考えられる。

次に、Musashi において、 N_z を 20, 30, 40, 50 に変化させたときの Func に関する A_0, A_1, A_2 の平均と分散を Fig. 7.56 に示す。全ての metric において $N_z = 30$ が最も良く、それ以上高くても低くても、一致率が減少することが分かった。Musashi の全筋に関わる関節自由度数は 46 であり、その値よりもかなり小さな値であった。また、Both については、 $N_z = \{20, 30, 40, 50\}$ のとき、 $A_2 = \{97, 97, 95, 87\}\%$ と、 N_z が十分小さな場合は Func と比べて大きな変化は見られなかった。

この分割をする前と後について、AutoEncoder を学習させた際の損失の変化はどの程度異なるのかについて検証する。この際はデータ数は 1000 まで落とし、学習が難しく、過学習しやすい状態となっている。分割後のグルーピングについては、Fig. 7.54 の Both の結果について行い、Fig. 7.57 に示すよう

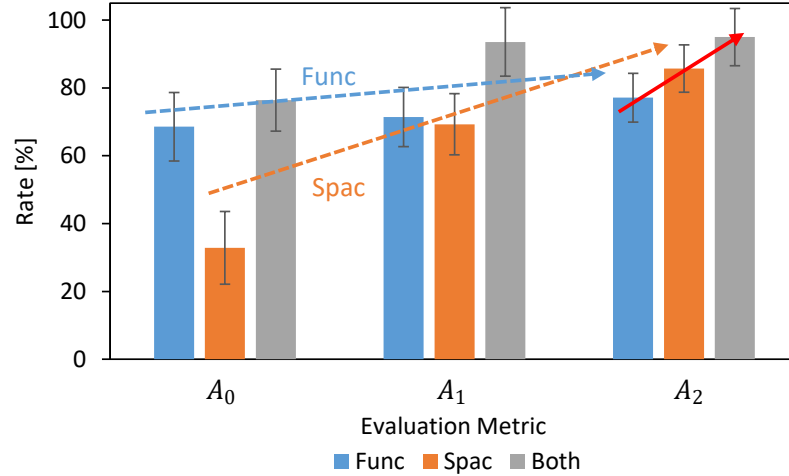


Fig. 7.55: Evaluation metric of A_0 , A_1 , and A_2 when conducting the proposed grouping method of Func, Spac, or Both 10 times for Musashi [267].

に、分割されたそれぞれのグループの \mathbf{l} と \mathbf{z} を用いて同様に AutoEncoder を学習させてみる。このとき、分割前と分割後で重みの総数は変わらない。分割後の損失については、それぞれのグループから得られた損失を \mathbf{l} の大きさに重み付け平均したものである。損失の遷移を Fig. 7.58 に示す。分割前は訓練時の損失がテスト時の損失を大きく下回り過学習しているのに対して、分割後はそれらが同じ遷移を辿り、過学習していない。関係のない余計な変数が消えることで、過学習しにくくなったと考えられる。

Kengoro における分割の例を Fig. 7.59 に示す。ここでは $N_z = 50$ としている。Kengoro の方が Musashi よりも身体構造は複雑であるが、Musashi のときと同様に、Func では比較的離れた 2 つのグループが合体することが多く、Spac では近い 2 つのグループが合体、または一本だけ近い異なる方へグルーピングされてしまうことが多かった。このグルーピングを 10 回行った際における、 A_0 , A_1 , A_2 の平均と分散を Fig. 7.60 に示す。傾向は Musashi と似ているが、Func と Spac については Musashi よりも一致率が低く、より複雑で難しい身体構造であることがわかる。同時に、Both では機能的接続と空間的接続を併用することで、Musashi の場合と同等の性能を叩き出している。

実機による評価実験

筋骨格ヒューマノイド Musashi の実機を使って実験を行う。空間的接続については前述のシミュレーション実験と同じものを用いるが、 \mathbf{l} のデータ列を実機のランダムなデータから取得する。関節や筋配置の情報を一切用いないため、それぞれの筋について長さの範囲を決め、ランダムな筋長を指令

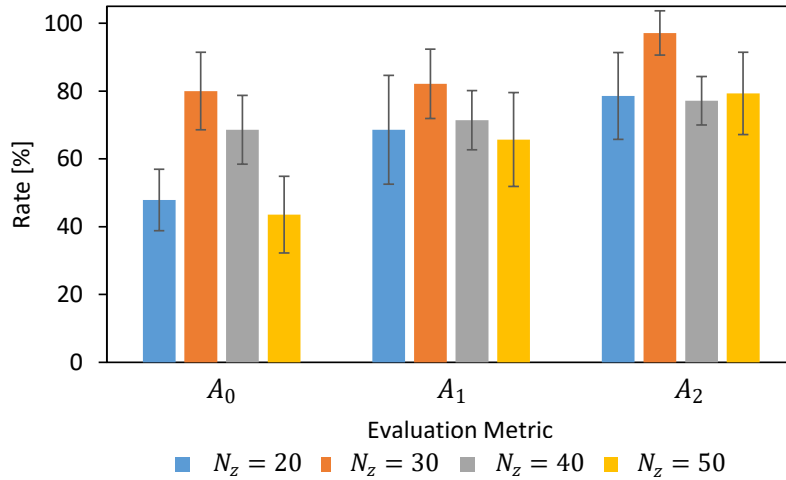


Fig. 7.56: Comparison of A_0 , A_1 , and A_2 when changing N_z to 20, 30, 40, or 50 and conducting the muscle grouping method of Func for Musashi [267].

する (Fig. 7.61). このとき、拮抗筋同士が引っ張りあい強い筋張力を発揮する状態や互いに緩む状態を抑制するため、[186]と同様の方法で筋張力の最大値と最小値を筋ごと設定している。2 Hz で約 25 分間動かして約 3000 のデータを取得し、前述のシミュレーション実験と同様に機能的接続を得る。ここで、学習の際はデータが少ないため batch 数を 50, epoch 数を 3000 とし、 $N_z = 40$ とする。このときの Func, Spac, Both による分割をそれぞれ 10 回行った際における A_0 , A_1 , A_2 の平均と分散を Fig. 7.62 に示す。Func による分割は難しく、ほとんど成功しなかった。これに対して、Both では、機能的接続を用いることで Spac よりも 10-20%前後も一致率が上がった。これは、実機ではデータ取得が難しいためデータ数が少なくノイズも大きいいため、幾何モデルと比べると大きく劣るが、空間的接続と併用することで、精度を上げられるということだと考えられる。

7.5.5 議論

本実験から、機能的接続と空間的接続を併用することで、精度高く筋群をグルーピングすることが可能であることがわかった。機能的接続だけでは空間的に遠い筋同士が同じグループに属してしまい、空間的接続だけでは空間的には違いが機能的には異なる筋同士を同じグループにしてしまう。この2つを合わせることで、Simulation では Musashi, Kengoro について A_0 は 80%前後、 A_2 は 95%前後を達成することができた。また、実機では大量のデータを取得することが難しいため、機能的接続だけでは上手くグルーピングすることが出来なかったが、空間的接続と合わせることで一致率を向上させることが可能であった。機能的接続を獲得する際の AutoEncoder については、 z の次元に適切な値が存在

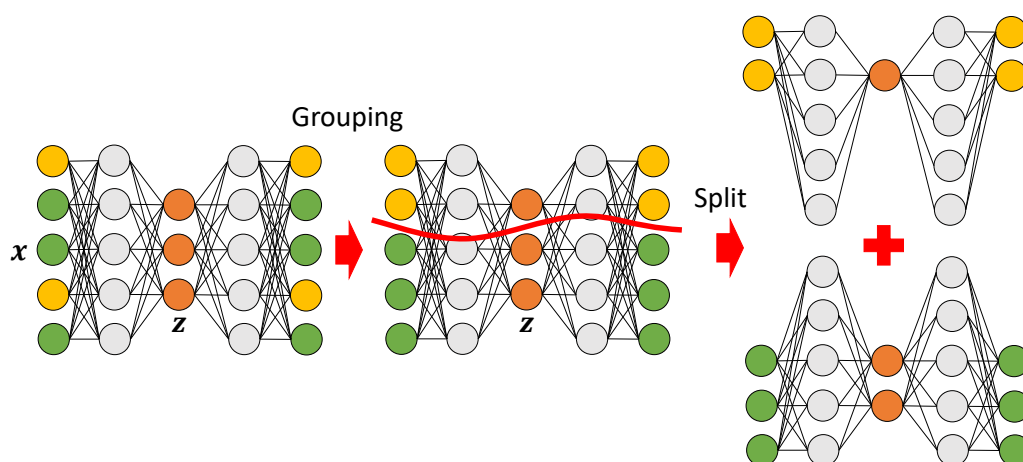


Fig. 7.57: The split of AutoEncoder for the investigation of loss transition when training it before and after muscle grouping [267].

し、これは本来の正しい次元である関節自由度数よりも小さい。

筋分割は、幾何モデルが分かれば関節とそれに関わる筋の配置から行えるため、正解データがあり評価がしやすいため本手法の適用先に選んだ。今後、筋配置や関節配置等を与えなくても、ランダムな動きから徐々にそれらの関係が分かり、筋が組織化されていくロボットが構成できると考える。また、今回は評価のためにグルーピング数を14と限定した。分割数をより小さくすると、関係の深いグルーピング同士が合体したようなグループが生成され、大きくするとさらに関節の持つ自由度数ごとに筋が分割されていくことになる。しかし、実際には分割数を限定せず、数種類を試し、分割した状態で制御器や認識器を作成し、その正確性やロバスト性のトレードオフから分割数を決定していく必要がある。これはタスクに依存するため評価は難しいが、今後取り組んでいく必要がある。

本手法は、機能が明確に分かれやすいようなものに対して適用できる手法であり、それぞれが非常に密に繋がっている場合には性能が落ちる可能性が高い。しかし、機能が分かれたものについては良い性能を示し、筋構造だけでなく、全身に分布した接触センサや慣性センサ、温度センサ等に使うことができる。今後、指だけに実装された接触センサや、腕だけに実装された特殊なセンサ群が、全身となったとき、その助けとなると確信する。また、本節では静的な関係だけを見たが、動的な関係が強い場合には、速度情報を入れたり、AutoEncoderをLSTM等を用いて再帰的にしたりする必要がある。

本手法により、自動的に筋が組織化され、その中で[178]のような学習器を構成していくという工程が自動化される。

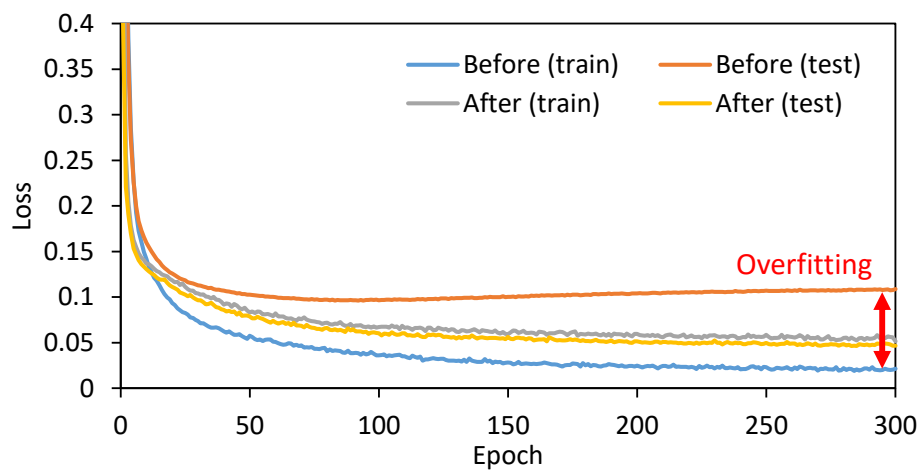


Fig. 7.58: Transition of train/test loss when training AutoEncoder before and after muscle grouping for Musashi [267].

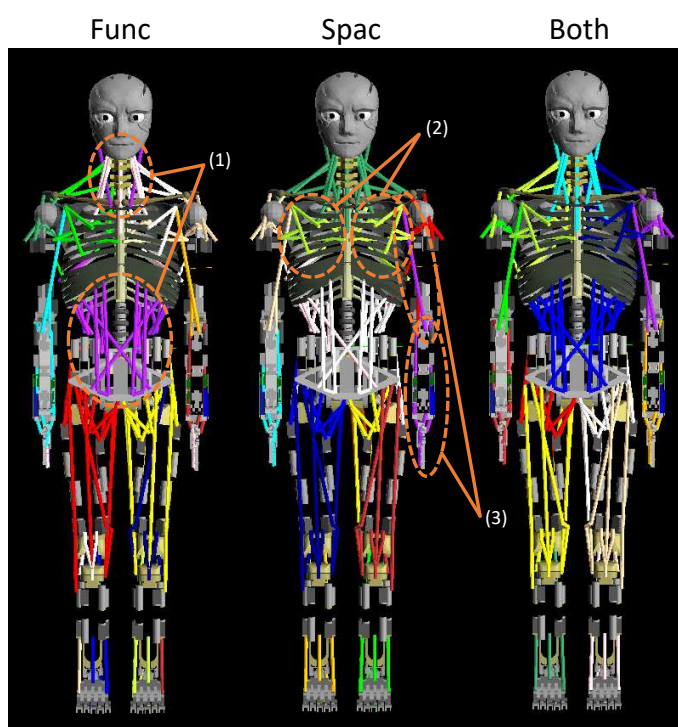


Fig. 7.59: Examples of muscle grouping when conducting the proposed grouping method of Func, Spac, or Both for Kengoro (Proposed) [267].

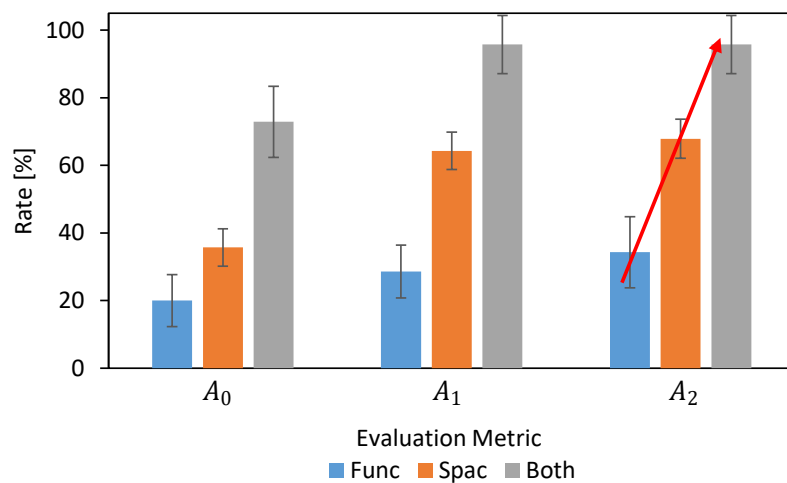


Fig. 7.60: Evaluation metric of A_0 , A_1 , and A_2 when conducting the proposed grouping method of Func, Spac, or Both 10 times for Kengoro [267].

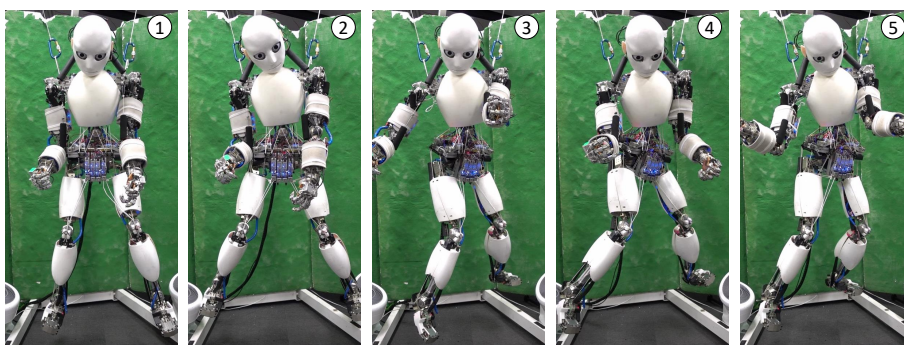


Fig. 7.61: The experiment of collecting muscle length data from random movements of the actual robot Musashi [267].

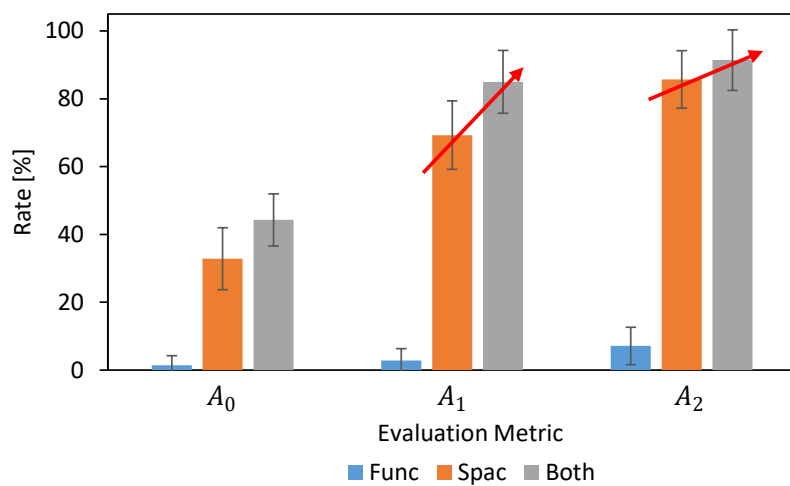


Fig. 7.62: Evaluation metric of A_0 , A_1 , and A_2 when conducting the proposed grouping method of Func, Spac, or Both 10 times for the actual robot Musashi [267].

7.6 動的な身体図式の展開による高速制御計算 - ペダル操作学習

7.6.1 概要と先行研究

筋骨格ヒューマノイドは様々な生物規範型の利点を持つと同時に、その複雑で柔軟な身体構造はモデリングが困難であり、これまで [190] や [167] 等の反射に基づく制御手法, [100] や [124] 等の自己身体モデル獲得手法が開発されてきた。前者は筋の反射によって対症療法的にモデリングの誤差を吸収するのに対して、後者は自己身体モデル自体を実機センサデータからのオンライン学習によって更新していく。しかし、後者の方法を用いても、実機と自己身体モデルの誤差を完全に無くすることはできない。そのため、ある特定のタスクを実現する制御をモデル通りに実装しても、正しくそのタスクを実現できるとは限らない。

そこで、本節では特定のタスクに特化して自己身体の制御法を学習させることで、そのタスクをより正確に実行できる手法を開発する [271]。つまり、あるタスクを実現するためのコントローラをチューニングする必要がなくなり、少ない実機の動作データから最適なコントローラを獲得することができる。本研究のネットワークは、動的な一般化多感覚相関モデルにおいて、 m と p を消し、時系列を表す再帰的ニューラルネットワークを展開して通常の再帰的でないニューラルネットワークと同等に扱うことで、その実行速度を高速化したものである。

本研究の詳細な貢献を以下にまとめる。

- 筋骨格ヒューマノイドにおける制御入力と状態の関係と、これまでの自己身体モデル獲得に関する考察
- 時系列関係を考慮した制御入力とタスク状態の関係を記述する DDC-Net (Dynamic Direct Control Network) の構成
- DDC-Net を用いた、動的に所望のタスクを実現可能な制御手法
- 筋骨格ヒューマノイドにおける自動運転のためのペダル操作実機実験による評価

以下ではまず、筋骨格ヒューマノイドの身体構造の要約、本研究と自己身体モデル獲得との関係、ヒューマノイドにおけるペダル操作について述べる。次に、本研究の提案手法である DDC-Net の構成と、DDC-Net を用いた所望の制御状態実現手法について述べる。最後に、筋骨格ヒューマノイドによる車のペダル操作を例に取り (Fig. 7.63)、実機実験により本手法の有効性を確認する。



Fig. 7.63: Autonomous driving by the musculoskeletal humanoid, Musashi [271].

7.6.2 筋骨格ヒューマノイドにおける自動運転

筋骨格ヒューマノイド

本研究で用いるのは筋骨格ヒューマノイド Musashi [87] である。上半身の筋肉はダイニーマによる筋ワイヤと非線形弾性要素からなり、下半身はダイニーマのみで非線形弾性要素はない。ダイニーマはバネのように伸び、500 N で約 8-10 mm 程度の伸びが発生し、摩擦やヒステリシス等からそのモデリングは難しい。本研究では基本的に Musashi の足首関節のみを用いる。

自己身体モデル獲得

Fig. 7.64 に、筋骨格ヒューマノイドと、通常の軸駆動型ヒューマノイド [78] における身体制御の比較を示す。ここでは、筋空間・関節空間・タスク空間の分類と、制御入力・状態の分類を設けている。

軸駆動型ヒューマノイドにおいては、関節にエンコーダが備えられており、関節状態を見ながら、関節の制御入力に値を合わせるようなフィードバックをかける。そのため、関節空間では制御入力と状態がほとんど同じになる。タスク空間を入力とした場合、そのタスクを実行する関節制御入力が決まり、関節状態がそれに追従し、行いたいタスクの状態が変化していく。関節空間においては制御入力と状態がほとんど同じであるが、一段上層のタスク空間では制御入力と状態に間に誤差が溜まっていく。DARPA Robotics Challenge [4] でタスクを失敗し転倒する事態も同じ現象と言える。

これに対して、人間のような筋骨格構造を持つ筋骨格ヒューマノイドは、筋空間という層が一段増えた、より複雑な制御構造を有している。ここでは、軸駆動型ヒューマノイドにおける関節空間と筋骨

格ヒューマノイドにおける筋空間が最下層で同じ働きであり、制御入力と状態がほぼ一致する。よって、筋骨格ヒューマノイドにおいて関節角度を制御入力として、その関節角度状態を正確に実現するのは、軸駆動型ヒューマノイドにおけるタスク空間と同じく、難しい問題となる。この問題に対して、[122, 100, 124] 等、多くの学習型制御の試みが成されている。これらの手法によって、意図した関節角度をある程度実現し、物体を掴むような動作が可能となってきた。しかしこれらは、関節と筋の関係の学習に終始し、さらに一段層を深くし、タスク空間を制御入力とした場合、層ごとに誤差が溜まっていくため、問題はさらに難しくなる。例えば、ボールを投げるためにボールの速度や軌道を制御したり、ペダル操作で車速を制御したり、といった動作である。

そこで本節では、筋骨格ヒューマノイドにおいて、タスク空間から関節空間、関節空間から筋空間という層を介した方法ではなく、制御入力から直接タスク状態を記述し、制御する手法を開発する。これにより、層を介することによる誤差を吸収し、より意図した通りのタスク実行を可能にすることができる。また、[122, 100, 124] では静的な状態のみを扱っていたが、本研究では時系列情報を考慮し、動的な制御を可能とする。その他先行研究として、EMD Net [25] や SE3-Pose-Nets [272] 等が存在するが、どれも時系列情報を扱った動的動作の実現はできていない。

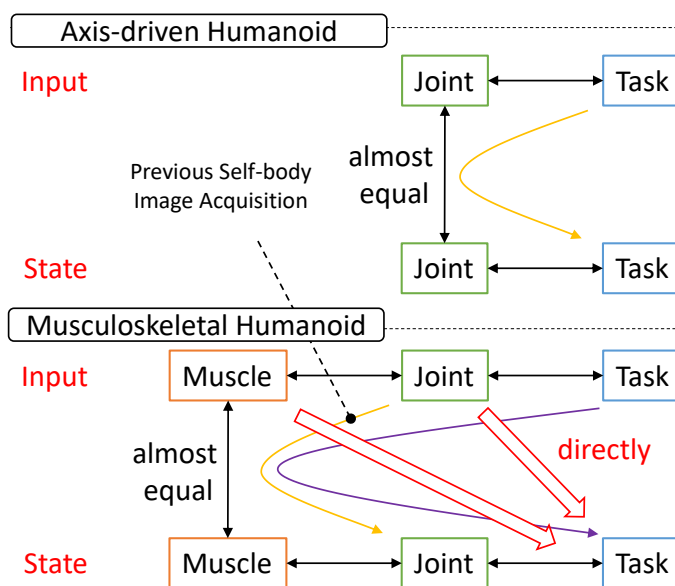


Fig. 7.64: Comparison of robot controls between musculoskeletal humanoids and ordinary axis-driven humanoids [271].

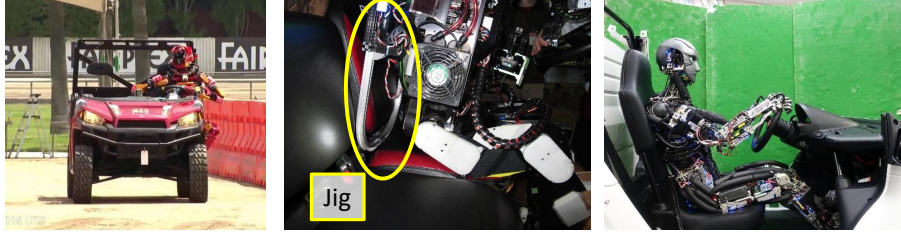


Fig. 7.65: Comparison of autonomous driving between using musculoskeletal humanoids and ordinary axis-driven humanoids [271]. In DARPA robotics challenge [4], jaxon [8] needed a jig to sit on the seat.

筋骨格ヒューマノイドによる自動運転

DARPA Robotics Challenge [4] の運転タスクにおいて、通常の軸駆動型ヒューマノイドである JAXON [8] は車のシートに座るためにジグを要した (Fig. 7.65). それに対して、筋骨格ヒューマノイドは人体のプロポーションを模倣しているがゆえに、人間のために作られた車体の中に容易に入ることができ、その柔軟な身体を環境に馴染ませることができる。そのため、筋骨格ヒューマノイドを運転操作に用いることは有用であると考え、本研究では運転において重要なペダル操作を扱う。ここでは基本的に、足首の pitch 軸によってアクセルを操作し、車体の速度を制御する。また、人間のペダル操作における感覚フィードバックは遅く、3.5 Hz かそれ以上程度と言われており [273], 本研究では 5 Hz の周期で制御を行う。

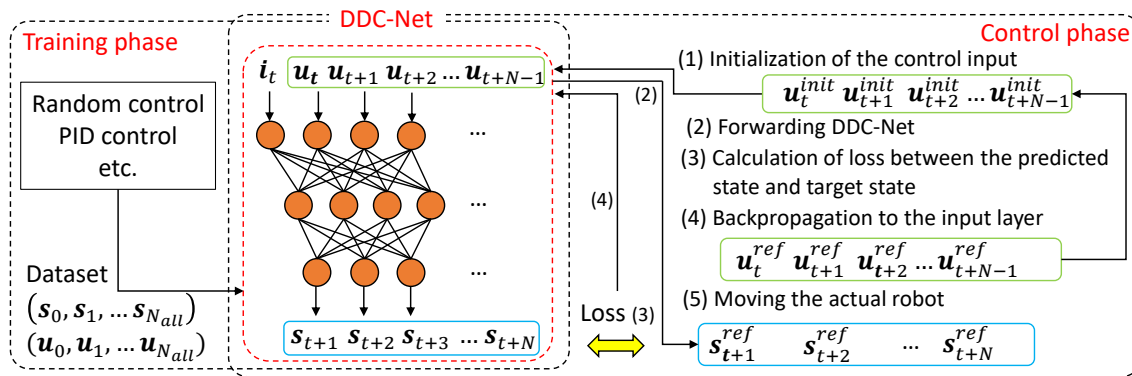


Fig. 7.66: Overview of the task-specific self-body controller acquisition method [271].

7.6.3 タスク特化型自己身体制御獲得

タスク特化型自己身体制御獲得手法の全体の構成を Fig. 7.66 に示す。DDC-Net のネットワーク構造、訓練フェーズ、制御フェーズについてそれぞれ詳細を述べる。

ネットワーク構造

本研究における DDC-Net は、以下の関数 h に等しい。

$$\left(s_{t+1}^T s_{t+2}^T \cdots s_{t+N}^T\right)^T = h\left(\left(i_t^T u_{t+1}^T u_{t+2}^T \cdots u_{t+N}^T\right)^T\right) \quad (7.36)$$

ここで、 s は N_s の次元を持つタスクの状態、 i は N_i の次元を持つ初期状態、 N は考慮する時系列情報の長さを表す定数、 u は N_u の次元を持つ制御入力を表す。初期状態に、 N 個の制御入力の時系列情報を与えることによって、その先の N 個のタスク状態の時系列情報を予測する。 $i_t = s_t$ が基本であるが、初期状態の情報を増やす場合も考えられる。

このネットワーク構造は何を用いても良いが、本研究では、5 層の全結合層からなるニューラルネットワークを用いる。最終層以外の全層に対しては、全結合層の後に Batch Normalization [45] を適用し、活性化関数としては Sigmoid を用いている。入力層の次元数は、 $N_i + N \times N_u$ であり、出力層の次元数は $N \times N_s$ となる。

本研究のペダル操作では、 $N = 30$ 、 $N_s = 1$ (車の速度)、 $N_i = 4$ (車の速度、車の加速度、足首ピッチ軸の関節角度、足首ピッチ軸の関節角速度)、 $N_u = 1$ (足首ピッチ軸の関節角度) となっており、隠れ層のユニット数は、 $\{80, 50, 20\}$ としている。

学習フェーズ

訓練フェーズにおいては、まず、ロボットをランダムな制御入力等によって動かした際のデータを蓄積する。動作データとして s と u が得られるため、これを $(i_t, u_{t+1}, u_{t+2}, \cdots, u_{t+N})$ と $(s_{t+1}, s_{t+2}, \cdots, s_{t+N})$ のようにネットワークに適する形に変換し、DDC-Net を学習させる (i は s や u 、それらの微分値等から作成する)。本研究では、損失関数として平均二乗誤差を用い、更新則には Adam [46] を用いる。全データの 1/5 をテストに用い、100 エポック学習させた際に最もテストの損失が小さなモデルを制御フェーズでは使用する。

制御フェーズ

制御フェーズにおいては、以下の 5 つの工程を実行する。

- (1) 現在のタスク状態の取得と、指令制御入力の初期値決定
- (2) DDC-Net の順伝播

(3) DDC-Net から出力された予測タスク状態と指令タスク状態の loss を計算

(4) 指令制御入力 of 初期値に対して誤差逆伝播

(5) (2)–(4) を繰り返し、最終的な指令制御入力値をロボットに指令

(1) においては、現在の状態 i_t と指令制御入力の初期値 $\mathbf{u}_{seq}^{init} = \{\mathbf{u}_{t+1}^{init}, \mathbf{u}_{t+2}^{init}, \dots, \mathbf{u}_{t+N}^{init}\}$ を決定する。ここで指令制御入力は、前ステップで最適化され最終的にロボットに指令された $\{\mathbf{u}_t^{prev}, \mathbf{u}_{t+1}^{prev}, \dots, \mathbf{u}_{t+N-1}^{prev}\}$ を一つずらして最終項を複製した、 $\{\mathbf{u}_{t+1}^{prev}, \mathbf{u}_{t+2}^{prev}, \dots, \mathbf{u}_{t+N-1}^{prev}, \mathbf{u}_{t+N-1}^{prev}\}$ を用いる。

(2)–(3) について、まずは DDC-Net を順伝播する。その後、出力された予測タスク状態 \mathbf{s}_{seq}^{pred} と指令タスク状態 \mathbf{s}_{seq}^{ref} の loss L を計算する。本研究では以下の loss を用いる。

$$L = \|\mathbf{s}_{seq}^{pred} - \mathbf{s}_{seq}^{ref}\|_2 + \alpha \text{AdjacentError}(\mathbf{u}_{seq}^{init}) \quad (7.37)$$

ここで、 α は重みづけの係数、AdjacentError は制御入力のステップ間の値の平均二乗誤差を表す。最終項は、ステップ間の制御入力の差を小さくすることで、滑らかな制御入力を導出するための工夫である。

(4) について、最後に、loss から誤差逆伝播により \mathbf{u}_{seq}^{init} を以下のように更新する。

$$\mathbf{g} = \partial L / \partial \mathbf{u}_{seq}^{init} \quad (7.38)$$

$$\mathbf{u}_{seq}^{init} = \mathbf{u}_{seq}^{init} - \beta \frac{\mathbf{g}}{|\mathbf{g}|} \quad (7.39)$$

ここで、 \mathbf{g} は L の \mathbf{u}_{seq}^{init} に対する勾配、 β は学習率を表す定数である。(2)–(4) の工程を繰り返して \mathbf{u}_{seq}^{init} を更新していく。

(5) において、最終的に求めた \mathbf{u}_{seq}^{ref} の \mathbf{u}_{t+1}^{ref} を実機に指令する。

上記が概要であるが、本研究ではさらに、(1)–(4) の工程を二段階に分けて更新する工夫を施している。第一段階では、 \mathbf{u}_{seq}^{init} に $-\delta \mathbf{u}_{batch}$ から $\delta \mathbf{u}_{batch}$ のランダムなノイズを加えて、 N_{batch} 個のサンプルを作成し、ノイズを入れないものも含め $N_{batch} + 1$ 個の初期値をバッチとして入力する。(2)–(4) の工程を、 $\beta = \beta_1$ に設定して N_1 回繰り返す。第二段階では、第一段階で最終的に最も loss が小さかった指令制御入力を初期値として用いる。 $\beta = \beta_2$ ($\beta_2 < \beta_1$) に設定し、(2)–(4) の工程を N_2 回繰り返す。最終的な \mathbf{u}_{seq}^{ref} を求める。 \mathbf{u} には最小値最大値が存在するため、その都度 \mathbf{u}^{min} から \mathbf{u}^{max} の間に値が収まるようにフィルタをかける。また、最初の試行時には \mathbf{u}_{seq}^{init} が存在しないため、全時間の \mathbf{u} を \mathbf{u}_{min} から \mathbf{u}^{max} のランダムな同一の値で埋めた \mathbf{u}_{seq}^{init} を N_{batch} 個作成してバッチとして用いる。

本研究のペダル操作においては、 $\mathbf{u}^{min} = 0$ [deg], $\mathbf{u}^{max} = 50$ [deg], $N_{batch} = 10$, $N_1 = 10$, $N_2 = 20$, $\delta \mathbf{u}_{batch} = 5$ [deg], $\alpha = 30.0$, $\beta_1 = 3.0$ [deg], $\beta_2 = 0.5$ [deg] としている。

本研究では関節角度を制御指令とし, その関節角度は [124] によって筋長に変換された後, 実際に実機に送られており, 関節角度センサを必要としない.

7.6.4 実験

まず, 本研究で用いる実験装置について説明する. 次に, 通常の PID 制御, 加速度推定を行った PID 制御によるペダル操作実験を行う. 最後に, 本研究で提案したタスク特化型自己身体制御獲得手法を用いた実験を行う.

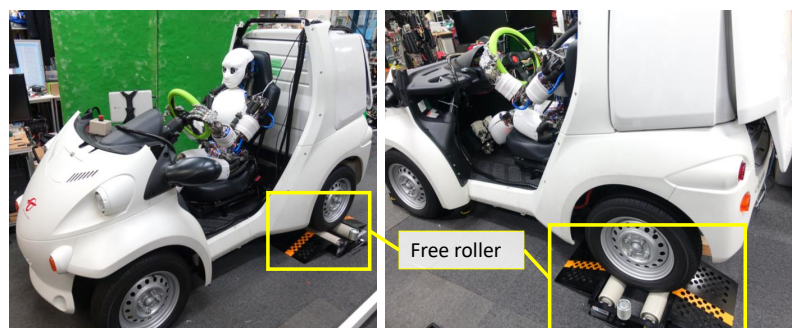


Fig. 7.67: Experimental setup of COMS [271].

実験セットアップ

本研究のペダル操作実験で用いる自動車は, トヨタ車体製の超小型 EV コムス (COMS: Chotto Odekake Machimade Suisui) シリーズの B・COM デリバリーである. 安全のため, モータトルクはソフトウェア上で 5 Nm に制限されており, 非常停止ボタンが備えられている.

実験装置の外観は Fig. 7.67 のようになっている. 通常の運転であれば画像情報からの Visual Odometry 等で車体の現在速度を取得するべきであるが, 安全のため室内環境で実験を行い, 車体速度は CAN-USB により ROS を介して COMS のソフトウェアから得ている. 駆動輪である後輪はフリーローラの上に載せており, 前輪は安全を期してストッパーにより固定されている.

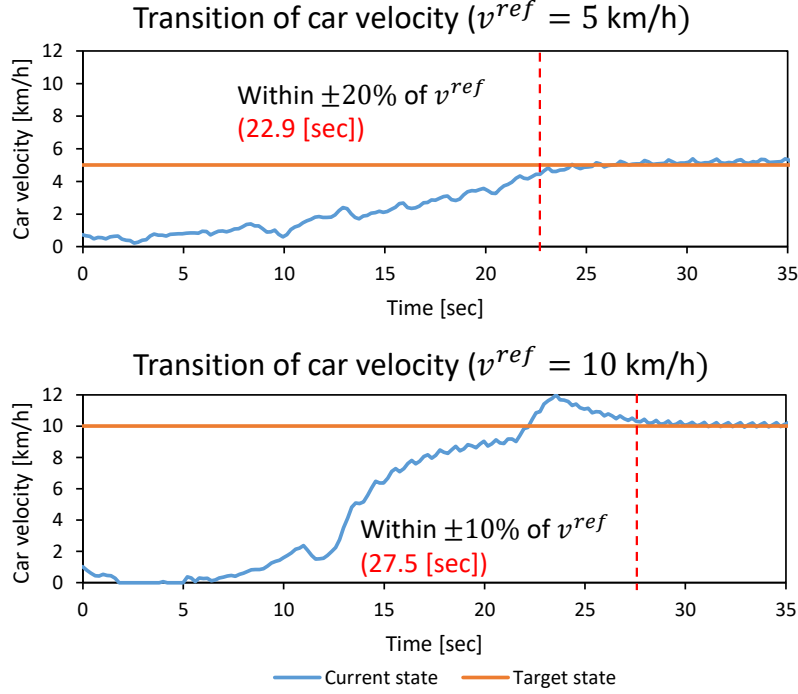


Fig. 7.68: The result of the basic PID control (**PID1**) [271].

基本制御

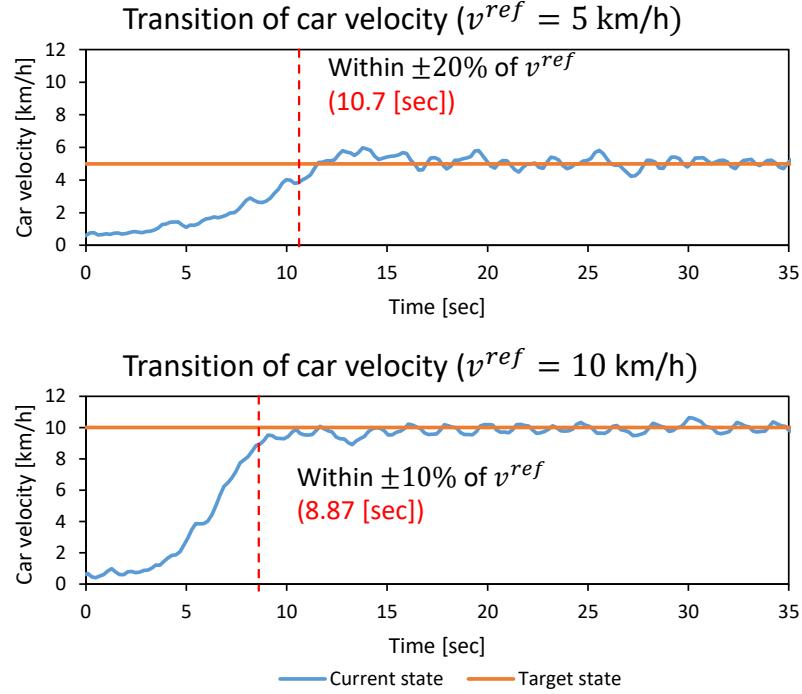
まず、従来のような通常の PID 制御 (以降 **PID1** と呼ぶ) を行った際の車体速度変化を Fig. 7.68 に示す。これは、

$$e_v(t) = v^{ref}(t) - v(t) \quad (7.40)$$

$$u = \theta_{ankle} = k_p^1 e_v(t) + k_i^1 \int e_v(t) + k_d^1 \dot{e}_v(t) \quad (7.41)$$

に基づいた制御であり、 k_p^1, k_i^1, k_d^1 はそれぞれ P ゲイン, I ゲイン, D ゲイン、 v は現在車速、 $u = \theta_{ankle}$ は制御入力である足首 pitch の関節角度を表す。 v^{ref} は指令速度であり、全実験で 5 km/h, 10 km/h の二種類の一定速度への追従実験を行っている。Fig. 7.68 からわかるように、**PID1** は指令速度への追従が遅い。それ以降指令速度からの誤差が $A\%$ に収まる最初の時間を $T_{A\%}^{conv}$ とすると、 $v^{ref} = 5$ [km/h] のとき $T_{20\%}^{conv} = 22.9$ [sec]、 $v^{ref} = 10$ [km/h] のとき $T_{10\%}^{conv} = 27.5$ [sec] となっている。車速が遅いときは速度追従が難しいため、 $v^{ref} = 5$ [km/h] では $A = 20$ 、 $v^{ref} = 10$ [km/h] では $A = 10$ を評価の基本としている。

PID1 による速度追従が遅い理由は、アクセルペダルの踏み込み量は実際には車体の速度ではなく、加速度項に大きく影響を与える量だからであると考えられる。そこで、目標加速度を推定し、その加速度に

Fig. 7.69: The result of the PID control with acceleration estimation (**PID2**) [271].

対して PID 制御を行うことを考える (以降 PID2 と呼ぶ). 具体的には, 以下のような制御を行う.

$$a^{ref}(t) = (v^{ref}(t) - v(t))/t_{delay} \quad (7.42)$$

$$a(t) = (v(t) - v(t-1))/t_{interval} \quad (7.43)$$

$$e_a(t) = a^{ref}(t) - a(t) \quad (7.44)$$

$$u = \int (k_p^2 e_a(t) + k_d^2 \dot{e}_a(t)) \quad (7.45)$$

ここで, a^{ref} は目標加速度, t_{delay} は目標速度を実現するまでの時間遅れ, $t_{interval}$ は制御間隔, a は現在加速度, k_p^2, k_d^2 はそれぞれ P ゲイン, D ゲインを表す. PID2 による速度制御の結果を Fig. 7.69 に示す. $v^{ref} = 5 \text{ [km/h]}$ のとき $T_{20\%}^{conv} = 10.7 \text{ [sec]}$, $v^{ref} = 10 \text{ [km/h]}$ のとき $T_{10\%}^{conv} = 8.87 \text{ [sec]}$ となっている. PID1 に比べて大きく速度追従の時間遅れが改善していることがわかる.

自己身体制御獲得実験

本研究で提案したタスク特化型自己身体制御獲得による結果を示す (以降 Proposed と呼ぶ). まず, 以下のようなランダムコントローラを用いて速度制御を行った結果を Fig. 7.70 に示す (以降 Random

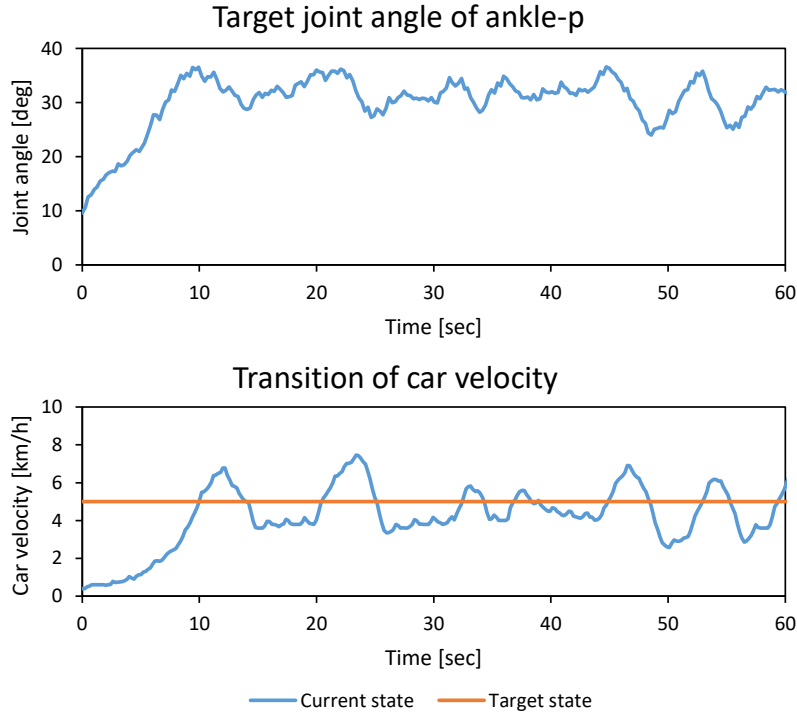


Fig. 7.70: The result of the random controller (**Random**) [271].

と呼ぶ).

$$\begin{aligned} \text{if } v(t) \leq v^{ref}(t) \\ u = u + \text{Random}(u_{min}^{random}, u_{max}^{random}) \end{aligned} \quad (7.46)$$

else

$$u = u - \text{Random}(u_{min}^{random}, u_{max}^{random}) \quad (7.47)$$

ここで, $\text{Random}(u_{min}^{random}, u_{max}^{random})$ は u_{min}^{random} から u_{max}^{random} の範囲のランダムな値である. 本研究では $u_{min}^{random} = -1$ [deg], $u_{max}^{random} = 2$ [deg] とした. Fig. 7.70 のように足首関節角度と車速は波打った動きをしており, ここで得たデータを用いて DDC-Net を学習させる. この制御は 60 秒間行い, 前述の通り 5 Hz の制御周期のため, $t_{interval} = 0.2$ [sec], つまり 300 ステップ分のデータが得られる.

上記で得られたデータを学習することで得られたタスク特化型自己身体制御を用いた速度追従実験の結果を Fig. 7.71 に示す. $v^{ref} = 5$ [km/h] のとき $T_{20\%}^{conv} = 0.9$ [sec], $v^{ref} = 10$ [km/h] のとき $T_{10\%}^{conv} = 5.3$ [sec] となっている. また, $v^{ref} = 10$ [km/h] のとき $T_{20\%}^{conv} = 1.7$ [sec] となっており, 誤差が多少残ったものの, PID1, PID2 と比べて収束は大幅に速いことがわかる.

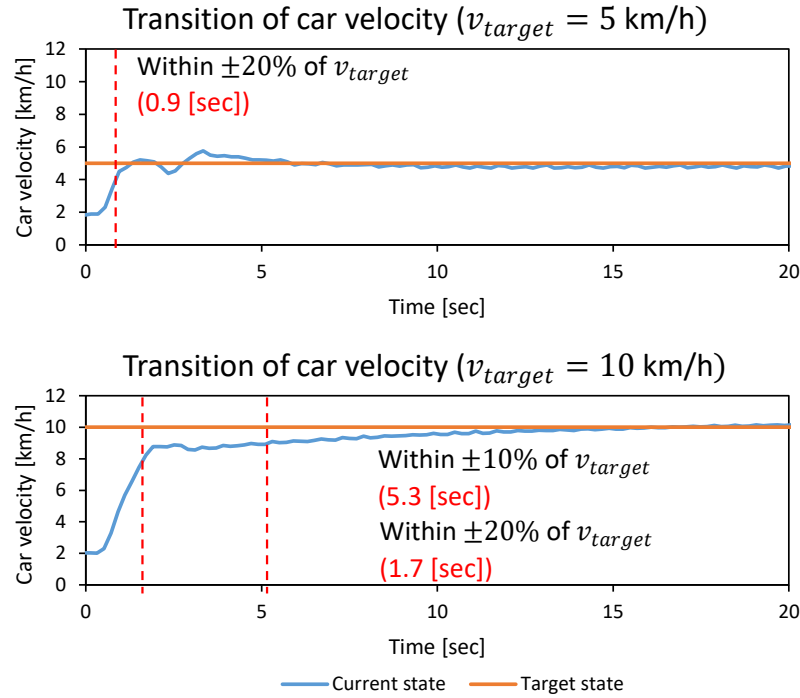


Fig. 7.71: The result of the task-specific self-body controller acquisition method (**Proposed**) [271].

7.6.5 議論

まず, 実験結果について考察する. 実験から, PID1, PID2 に比べて, **Proposed** は速度追従の収束が速いことがわかった. PID1, PID2 は, 更なるゲインチューニングにより収束を良くできる可能性が残るが, 本実験ではこれ以上ゲインを上げると激しく発振してしまった. **Proposed** は, **Random** を 60 秒間実行するだけで構築することが可能であり, 最適化パラメータは存在するものの, 最初に設定した値から変化させて挙動を確認したパラメータは α のみである. α は制御入力に対する制限の度合いを調整するパラメータであり, 小さくすると収束は速いものの振動が増え, 大きくすると収束は遅くなり振動が抑制される. 本研究では Eq. 7.37 のように loss を設定したが, その設定には様々なバリエーションが考えられ, 関節角度をある範囲に抑えたり, 制御入力の加速度変化を抑制する項を追加したりすることも可能である.

次に, **Proposed** の適用範囲に関して考察する. 本研究では, ヒューマノイドによるアクセルペダル操作に焦点を絞り実験を行ったが, 他にも様々な問題に対して適用可能である. 例えば, 運転操作ではアクセルだけでなくブレーキやハンドル操作, 運転中の身体の傾き補正等にも応用できると考える. また, 制御入力も様々に変更することができると考えられる. 関節角度指令ではなく, 関節速度指令, 関

節トルク指令, そして, 筋肉の筋張力を直接制御入力とすることも考えられる. また, 本研究ではある一定の速度に追従する実験を行ったが, $v^{ref}(t)$ は時間変化させることができる. 例えば, 滑らかに所望の場所で止まるようなブレーキをしたい場合, 躍動最小化の観点から時系列の速度指令を決定して, それに追従する制御を実行することができる.

7.7 動的な身体図式の展開による高速制御計算 - 把持安定化学習

7.7.1 概要と先行研究

これまで多くのロボットハンドが開発されてきた [242, 274, 243, 244, 275, 245, 276, 155, 116]. 数十本にも及ぶ腱により精巧なマニピュレーションを可能とするハンド [242, 274, 243] が多く存在する一方, ソフトロボティクスの台頭 [10] により, 空気圧により駆動する柔軟ハンド [244], 劣駆動な指を持つ腱駆動柔軟ハンド [275, 245, 276, 155, 116] が普及し始めている. これらは指が劣駆動でアクチュエータ数を絞り, 関節はゴムやバネ等によって構成されている場合が多い. 少ないアクチュエータでもその柔軟性により適応的な把持が可能であり, 耐衝撃性に優れ, 少数の大きなアクチュエータを用いることで高把持力を実現することもできる [155, 116].

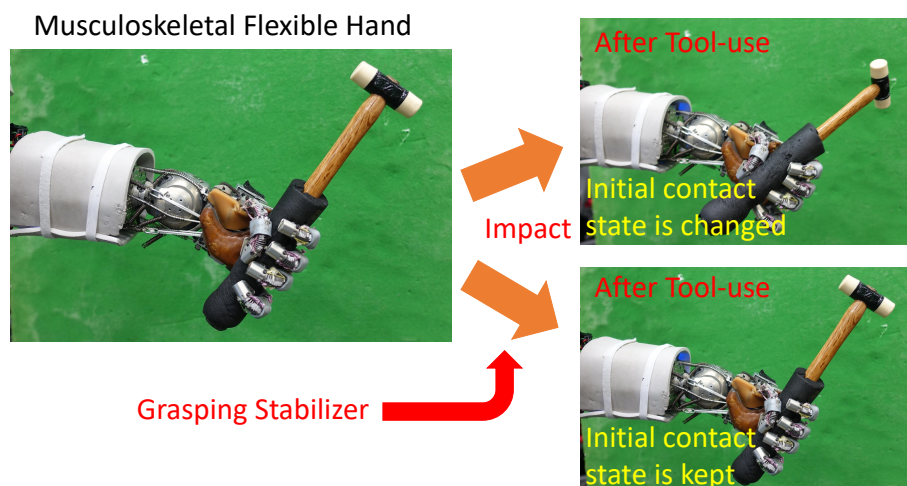


Fig. 7.72: Grasping stabilizer for tool-use [229].

これらの柔軟ハンドは基本的にはフィードフォワードに制御入力を決め, 一定の把持形態を維持したり, 一定の力を加え続けたりすることが多い. これは, 劣駆動で関節やリンクの組織自体が柔軟なハンドのモデル化は非常に難しく, センサとアクチュエータの関係を一意に決めることができず, フィードバック制御が困難なためである. そのため, 例えば道具を把持して用いる場合, Fig. 7.72 のように, 動作中に道具の慣性や衝撃から徐々に接触状態が変化し, 初期の接触状態を常に保ち続けられるとは限らない. 接触状態の変化によって, ときには把持物体を落したり, 把持した道具の姿勢が変化してしまったりすることがしばしばある.

一方, 全駆動ハンドやモデル化容易な劣駆動ハンドでは, 操作平面の限定や動作の制限を含めたうえで, 正確なモデル化と視覚・接触センサの利用によって, リアルタイムの触覚フィードバックや再把

持計画が行われている. [277] では視覚とひずみゲージを用いて, リアルタイムに2次元平面上での動作変更を行っている. [278] では, 決まった把持形態に関して把持を安定化させるための接触センサを用いた力最適化について議論している. [279] では, 手のひらは考えず, 3指の接触による平面上の動きのみ考えて安定した把持実現のための力フィードバックを行っている. [280] では, 接触センサ情報から正確にドアを開ける動作を実現している. [281, 282] では, 一度把持した際の接触センサ値から, より安定した接触を実現できるよう再把持を行う. [283] は視覚により把持が成功したかどうかを判定し, 場合によっては再把持をすることで, しっかりと衣服を広げてたたむ

これらに対して, 深層学習の台頭により, 様々な学習ベースの手法が開発されてきている. [246] は把持成功率の予測と強化学習による再把持を実現している. [247] は模倣学習を使い, シミュレーションにおいて五指ハンドによる様々なマニピュレーションタスクを実現している. [62] は劣駆動なロボットハンドに対して接触センサを用いて強化学習を適用し, 二次元平面上で二指による In-hand Manipulation を実行している. [248] は空気圧駆動の劣駆動柔軟ハンドによる把持物体の分類を行っている. しかし, 強化学習の多くはシミュレーションであり, 柔軟ハンドのシミュレーションは難しいため, 把持物体の分類等が多く, In-hand Manipulation や触覚フィードバックを実現した研究は少ない. また, 道具使用の際の把持安定化に着目した研究はない.

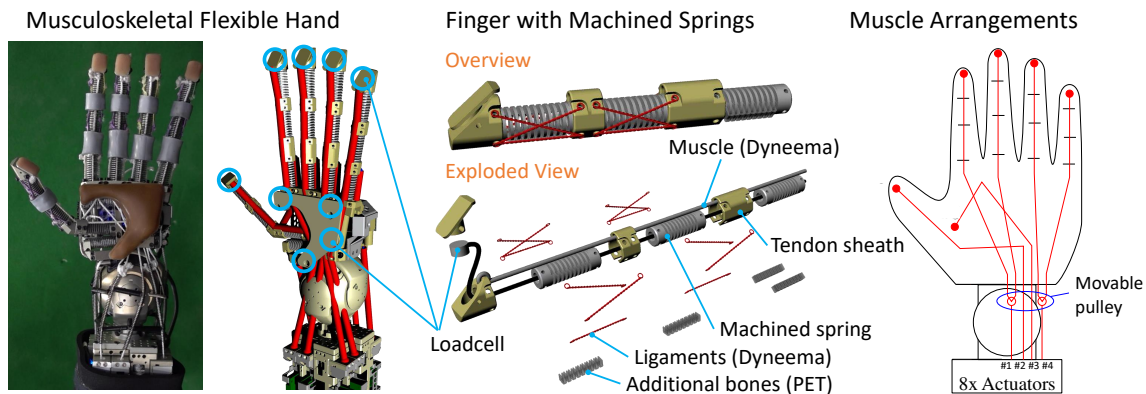


Fig. 7.73: The flexible hand of the musculoskeletal humanoid Musashi [87].

そこで本研究では, 柔軟ハンドによる道具把持の際の, 接触状態遷移予測モデル構築とそれを用いたフィードバック制御による接触状態維持フィードバック制御を行う [229]. [284, 271] をベースとして探索や損失関数, 最適化等を新たにし, 道具把持に着目した新しい把持安定化制御を提案する. 筋骨格ヒューマノイド Musashi [87] に搭載された筋骨格五指ハンド [116] に本手法を適用し, ハンマー打撃動作, 箒掃き動作, 掃除動作を行うことで, 本研究の有効性を確認する. 本研究のネットワークは,

7.6節と同じく、動的な一般化多感覚相関モデルにおいて、 m と p を消し、時系列を表す再帰的ニューラルネットワークを展開して通常の再帰的でないニューラルネットワークと同等に扱うことで、その実行速度を高速化した研究である。

7.7.2 把持安定化制御

本研究全体の手順は以下のようになっている。

1. 道具把持状態における制御入力探索動作
2. 接触状態遷移の予測モデル学習
3. 予測モデルを用いた道具使用動作における把持安定化

本研究の定式化

本研究で扱う問題について定式化を行う。本研究では筋張力 f とロードセルの接触センサ値 F を接触状態 $s^T = \begin{pmatrix} f^T & F^T \end{pmatrix}$ として扱う。また、指令筋長 l^{ref} を制御入力 u 、筋長 l と筋長速度 \dot{l} を制御状態 $i^T = \begin{pmatrix} l^T & \dot{l}^T \end{pmatrix}$ とする。本研究における接触状態遷移の予測モデルは、以下の式で表される。

$$\begin{aligned} s_{[t+1, t+T]} &= \begin{pmatrix} s_{t+1}^T & s_{t+2}^T & \cdots & s_{t+T}^T \end{pmatrix}^T \\ u_{[t, t+T-1]} &= \begin{pmatrix} u_t^T & u_{t+1}^T & \cdots & u_{t+T-1}^T \end{pmatrix}^T \\ s_{[t+1, t+T]} &= h(s_t, i_t, u_{[t, t+T-1]}) \end{aligned} \quad (7.48)$$

ここで、 h は予測モデル、 t は現在のタイムステップ、 T は何ステップ先まで予測するかを表す。

この h を実機データを用いて学習する。その後、この h を用いて把持安定化を行う。最初にフィードフォワード的に掴んだ際の接触状態を s^{keep} とする。この s^{keep} を保ち続ける制御入力 $u_{[t, t+T-1]}^{opt}$ を以下のように求める。

$$\begin{aligned} s_{seq}^{pred} &= h(s_t, i_t, u_{seq}^{init}) \\ u_{seq}^{opt} &= \arg \min_{u^{min} \leq u^{init} \leq u^{max}} L_{opt}(s_{seq}^{pred}, s_{seq}^{keep}, u_{seq}^{init}) \end{aligned} \quad (7.49)$$

ここで、 $u^{\{min, max\}}$ は u の最小値または最大値、 L_{opt} は最適化のための損失関数、 $s_{seq}^{\{pred, keep\}}$ は $s_{[t+1, t+T]}^{\{pred, keep\}}$ を省略したもの、 $u_{seq}^{\{init, keep\}}$ は $u_{[t, t+T-1]}^{\{init, keep\}}$ を省略したもの、 $s_{[t+1, t+T]}^{keep}$ は s^{keep} を T 個並べた値を表す。 $s_{[t+1, t+T]}$ はタイムステップ t のときには分からないため、制御入力の初期値 u_{seq}^{opt} を予測

モデル \mathbf{h} に入力したときの予測値 s_{seq}^{pred} を用いる。この s_{seq}^{pred} と s_{seq}^{keep} の値が近くなり、かつ実行可能な \mathbf{u}_{seq}^{opt} が出力されるような損失関数 L_{opt} を設計する必要がある。この Eq. 7.49 を、毎タイムステップ行い、 $\mathbf{u}_{[t,t+T-1]}^{opt}$ を計算し続ける。

また、Eq. 7.49 の計算には時間を要し、現在のタイムステップから次のタイムステップまでの時間を使って $\mathbf{u}_{[t,t+T-1]}^{opt}$ を求める。そのため、実機に指令する値は \mathbf{u}_t^{opt} ではなく、 \mathbf{u}_{t+1}^{opt} となる。

予測モデルのネットワーク構造

前述の予測モデル Eq. 7.48 の実装について述べる。これには様々な実装方法が考えられる。一つとして、LSTM [241] を使った方法が考えられる。つまり、 $\mathbf{s}_{t+1} = \mathbf{h}_{rnn}(\begin{pmatrix} \mathbf{s}_t^T & \mathbf{u}_t^T \end{pmatrix}^T)$ を LSTM によって表現し、それを T 回展開することで Eq. 7.48 と同じ形を得る方法である。これはモデルの小ささや展開数 T を可変にできるというメリットを有する一方で、モデルを連続的に展開するため計算に時間を要し、Eq. 7.49 の計算の際に不利となる。

本研究では、 T を固定して、入力と出力を含めた 5 層のニューラルネットワークで直接 Eq. 7.48 を表現する方法をとる。これは、一度のフォワードで $\mathbf{s}_{[t+1,t+T]}$ を直接計算できるため、計算速度の観点では有利となる。本研究では中間層のユニット数を (100, 100, 100) として、最終層以外の全ての層に Batch Normalization [45] と活性化関数 Sigmoid を挿入している。

本研究では値の大きさを揃えるために、 l の単位を [mm/10]、 F の単位を [N/10]、 f の単位を [N/200] としている。また、制御周期は速い必要はなく、Eq. 7.49 の計算に時間を使うため 5 Hz で行っており、 $T = 10$ として 2 sec 先までの状態を Eq. 7.48 が予測する。

ランダム探索行動

この予測モデルを学習させるために、実機データを取得する。道具を握る際、まずフィードフォワード的にある一定の把持姿勢 \mathbf{l}_0^{ref} を指令する。ここで、制御入力 \mathbf{u} となる \mathbf{l}^{ref} は、 \mathbf{l}_0^{ref} からの差分とする。この際のランダムな制御則を以下のように定める。

$$\Delta \mathbf{u} = \mathbf{C}_{rand} \sin(\mathbf{C}_{time} t) \quad (7.50)$$

$$\mathbf{u} \leftarrow \mathbf{u} + \text{Random}(-\Delta \mathbf{u}, \Delta \mathbf{u}) \quad (7.51)$$

$$\mathbf{u} \leftarrow \max(\mathbf{u}^{min}, \min(\mathbf{u}, \mathbf{u}^{max})) \quad (7.52)$$

ここで、 \mathbf{C}_{rand} 、 \mathbf{C}_{time} は $\Delta \mathbf{u}$ を決める係数であり、 $\text{Random}(a, b)$ は $[a, b]$ の範囲でランダムな値を出力する関数である。

$\Delta \mathbf{u}$ を一定の値に固定することも可能である。しかしこの場合問題が発生する。本研究のタスクにおいては、衝撃や外力が加わらなければ、 $\mathbf{l}^{ref} = \mathbf{0}$ を常に保ち続けることが最善である。そのため、 $\Delta \mathbf{u}$ を固定した場合、 $\mathbf{l}^{ref} = \mathbf{0}$ を保つデータが得られず、常に接触状態を崩してそれを戻す動作を繰り返すような挙動が発生してしまう。よって、このように $\Delta \mathbf{u}$ を変化させることで多様なデータを取得し、より安定して動作することが可能となる。

このデータを用いて、Eq. 7.48 を、バッチサイズを N_{batch}^{train} 、エポック数を N_{epoch}^{train} として学習させる。この際の損失関数は以下に示すような L_{origin} を用いる。

$$L_{origin}(\mathbf{s}_{seq}^{pred}, \mathbf{s}_{seq}^{keep}) = \|\mathbf{s}_{seq}^{pred} - \mathbf{s}_{seq}^{keep}\|_2 \quad (7.53)$$

ここで、 $\|\cdot\|_2$ は L2 norm を表す。

本研究の実験では、 $\mathbf{u}^{min} = -5$ [mm], $\mathbf{u}^{max} = 20$ [mm], $C_{rand} = 3$ [mm], $C_{time} = 0.02$ [1/timestep], $N_{batch}^{train} = 10$, N_{epoch}^{train} とする。

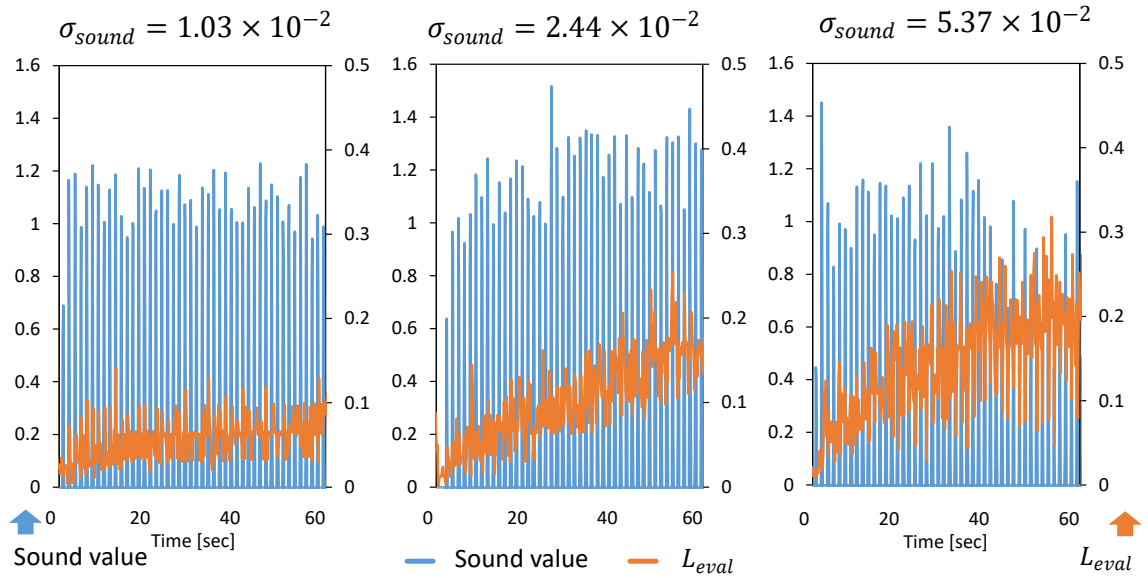


Fig. 7.74: Experimental evaluation of the correlation between L_{eval} and sound value when hitting a plate with a hammer [229].

損失関数の設計

最適化 Eq. 7.49 の損失関数 L_{opt} の設計について説明する。なお、制御入力 \mathbf{u} に関する制約については後述する、本研究では把持を安定化する、つまり、現状の接触状態を維持し続けることを目的として

いる. ここで, 本研究で使用するような接触センサや筋張力センサの性質を考慮する必要がある. これらのセンサは接触がなくなると値が0になるのに対して, 接触が強くなる方向には定格まで常に値が大きくなり続ける.

ここで, 損失関数 L_{opt} を訓練時と同様に L_{origin} として設定したとする. このとき, s_{seq}^{pred} が小さくなる方向に関しては途中でその値は0となり, 損失は $\|s_{seq}^{ref}\|_2$ より大きくならない. しかし, s_{seq}^{pred} が大きくなる方向は際限がない. ゆえに, 最適化の結果として s_{seq}^{pred} は0の方向に動きやすくなってしまふ.

そこで, 本研究では以下のように損失関数を定義する.

$$L_{grasp}(s_{seq}^{pred}, s_{seq}^{keep}) = \frac{1}{T} \sum_{i=1}^T w_i \|s_{t+i}^{pred} - s_{t+i}^{keep}\|_2 \quad (7.54)$$

$$w_i = \begin{cases} 1.0 & (s_{t+i}^{pred} \geq s_{t+i}^{keep}) \\ C_{loss} & (otherwise) \end{cases}$$

ここで, $C_{loss}(C_{loss} > 1)$ は s_{seq}^{pred} が s_{seq}^{keep} より下がる方向に対しては損失を大きくするための係数を表す. この損失関数 L_{grasp} を用いることで, 必要な接触を保ちつつ把持を安定化することができる. 本研究では $C_{loss} = 10.0$ としている.

この L_{grasp} と把持の安定性の相関を調べるために, 簡単な実験を行った. 先ほどの L_{grasp} を現在の $s_t^{current}$ のみについて考えたものを以下のように L_{eval} として定義する.

$$L_{eval} = w_0 \|s_t^{current} - s_t^{keep}\|_2 \quad (7.55)$$

Musashi がハンマーを握って木の板を連続して叩き, その際の音をフーリエ変換して特定の周波数域の音の振幅の最大値を得る. この音の値とその際の L_{eval} の値の遷移の様子を3つ Fig. 7.74 に示す. σ_{sound} は sound value の分散を表す. 左から右に行くに従い, L_{eval} の値の変化が大きくなっていることがわかる. また, それに伴い σ_{sound} の分散も大きくなっている. L_{eval} は把持の安定性に相関があり, 把持状態が変化することで音の様子も変化してくことがわかる. よって, L_{grasp} をベースとして把持安定性を最適化していく.

把持安定化制御

Eq. 7.49 の計算手順を以下に示す.

- (1) 最適化の初期値 u_{seq}^{init} の決定

(2) 損失関数 L_{opt} の計算

(3) \mathbf{u}_{seq}^{opt} の最適化

(1) について、最適化の結果は初期値に依存するため非常に重要である。本研究では一定制御入力
の初期値 N_{const} 個と、前回最適化結果にノイズを加えた値 N_{opt} を足し合わせた $N_{const} + N_{opt}$ 個の
データをバッチとして用意する。前者は、 \mathbf{u}^{min} から \mathbf{u}^{max} までを N_{opt} 等分し、それぞれの値によって
[$t, t+T-1$] 個の時系列を埋めたデータである。また後者は、前回最適化された \mathbf{u} を $\mathbf{u}_{[t-1, t+T-2]}^{prev}$ として、
この値をずらして最終項を複製した $\mathbf{u}_{\{t, t+1, \dots, t+T-2, t+T-1\}}^{prev}$ に [-0.1, 0.1] の一様なノイズを加えたデー
タを N_{opt} 個用意する。 $t = 0$ については前回最適化の結果がないため、0 で埋めた値に対してノイズ
を加えて用いている。これらの初期値から始め、最終的に最も損失 L_{opt} が小さくなったものを実機に
指令することになる。

次に、(2) の損失 L_{opt} は以下のように計算する。

$$L_{opt} = L_{grasp}(s_{seq}^{pred}, s_{seq}^{keep}) + C_{min} \|\mathbf{u}_{seq}^{init}\|_2 \\ + C_{adj} \|\mathbf{u}_{[t, t+T-2]}^{init} - \mathbf{u}_{[t+1, t+T-1]}^{init}\|_2 \quad (7.56)$$

ここで、 C_{min}, C_{adj} は定数である。Eq. 7.56 の右辺第二項は制御入力をなるべく小さくする項であり、
第三項はタイムステップ間の制御入力の差を小さくし滑らかにする項である。この損失関数 L_{opt} を
 \mathbf{u}_{seq}^{init} のバッチ内のデータそれぞれに対して計算し、最も損失が小さかった値を \mathbf{u}_{seq}^{opt} とする。

最後に、(3) では \mathbf{u}_{seq}^{opt} を以下のように最適化していく。

$$\mathbf{g} = \partial L_{opt} / \partial \mathbf{u}_{seq}^{opt} \quad (7.57)$$

$$\mathbf{u}_{seq}^{opt} \leftarrow \mathbf{u}_{seq}^{opt} - \gamma \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \quad (7.58)$$

ここで、前述のように \mathbf{u}_{t+1}^{opt} が実機に指令され、 \mathbf{u}_t^{opt} は前回指令された固定の値であるため更新しな
い。このとき、 γ の値を決め打ちしても良いが、本研究では様々な γ によってバッチを作成し、最も良
い γ を選ぶ。 γ の最大値 γ^{max} を決め、0 から γ^{max} までの値を N_{batch}^{opt} 等分し、それぞれの値によって
更新された \mathbf{u}_{seq}^{opt} を N_{batch}^{opt} 個作成する。もう一度 $\mathbf{u}_{seq}^{init} \leftarrow \mathbf{u}_{seq}^{opt}$ として (2) の Eq. 7.56 を行い、最も L
が小さかった \mathbf{u}_{seq}^{opt} を採用する。上記 (2)–(3) の工程を、 N_{epoch}^{opt} 回繰り返して \mathbf{u}_{seq}^{opt} を最適化していく。

最終的に得られた \mathbf{u}_{seq}^{opt} における \mathbf{u}_{t+1}^{opt} を実機に指令し、把持を安定化する。

本研究では、 $N_{opt} = 13$, $N_{const} = 13$, $C_{min} = 0.1$, $C_{adj} = 0.1$, $\gamma^{max} = 0.5$, $N_{batch}^{opt} = 13$, $N_{epoch}^{opt} = 10$
としている。

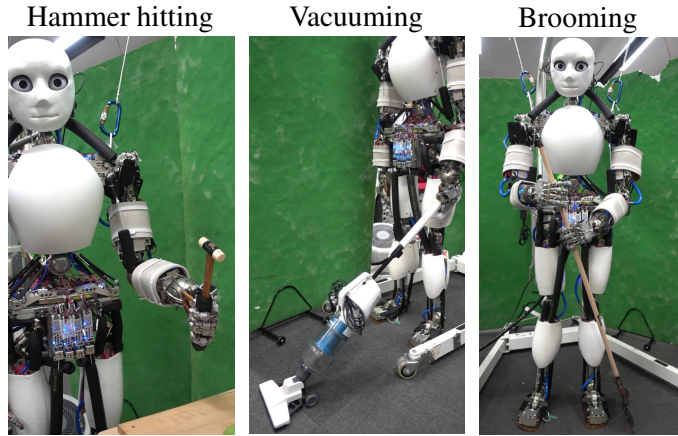


Fig. 7.75: Experiments of hammer hitting, vacuuming, and brooming [229].

7.7.3 実験

ハンマー打撃動作, 箒掃き動作, 掃除動作を行うことで, 本研究の有効性を確認する. 特にハンマー打撃動作については把持探索動作・訓練結果・把持安定化に分けて詳しく説明する.

それぞれの実験の様子を Fig. 7.75 に示す. 基本的には左手で道具を把持し, それを振ることでタスクを実行する. 箒については, 左手にのみ把持安定化制御を適用し, 右手には固定具を使用している.

ハンマー操作実験

まず, ランダムに指を動かすことによって, 予測モデル学習のための実機データを取得する. ここで, 7.7.2 節の探索行動に示したように, Δu の値を変化させることによって実機データを学習させる方法を Proposed Search, 通常のランダムウォークのように変化させずに一定値 C_{rand} を用いる方法を Constant Search と呼ぶこととする. この Proposed/Constant Search における u の動きを Fig. 7.76 に示す. ここで, #1 – #4 は Fig. 7.73 の右図におけるアクチュエータ番号と同等である. Constant Search では u が常に激しく振動しているのに対して, Proposed Search では激しく振動する区間と滑らかな区間が存在していることがわかる. 得られたデータを 8:2 で train と test 用に分けて学習し, test データに関する損失関数 L_{origin} の遷移を Fig. 7.77 に示す. 概ね同じような遷移をしており, Variable Search の方が滑らかなデータが増えるため損失は少しだけ少ないことがわかる.

得られた予測モデル h を用いて把持安定化を行う. ハンマーを持ち, 一定間隔で叩き続ける. この際の L_{eval} の遷移について, h を Proposed Search のデータによって学習して把持最適化に用いた場合, h を Constant Search のデータによって学習して把持最適化に用いた場合, 一切の把持安定化を行わな

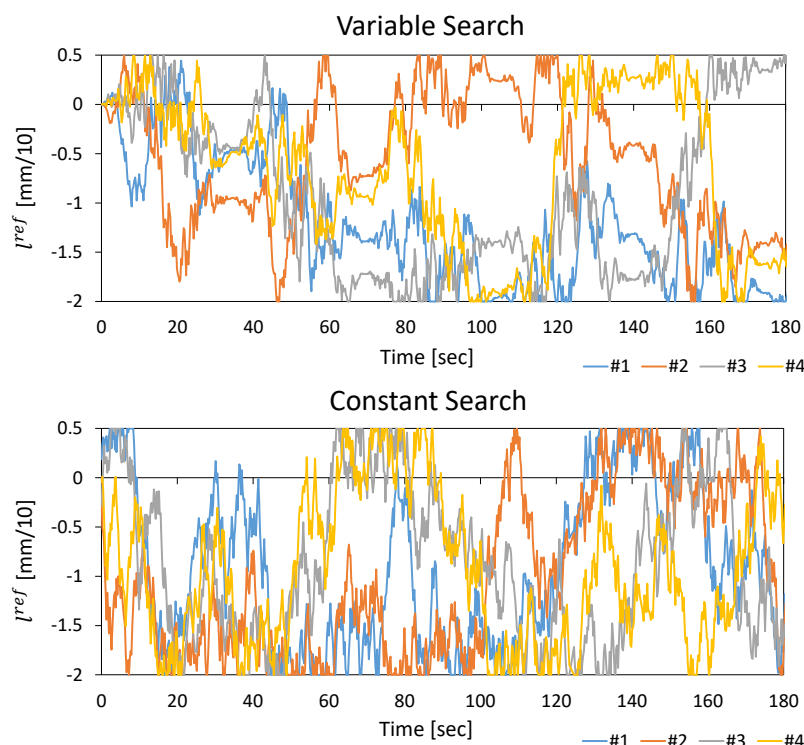


Fig. 7.76: Transition of u during random search behaviors: Variable and Constant Search [229].

い場合で比較する。その結果を Fig. 7.78 に示す。右下の図は、2 sec 間の移動平均を取った比較図となる。把持安定化をしない場合に比べて、安定化をした場合のほうが L_{eval} は小さくなり、初期の接触を保ち続けられていることがわかる。予測モデルの学習に Proposed Search を使った場合は、Constant Search を使った場合に比べて L_{eval} の平均が小さく、分散も小さいことがわかる。これは、 u について、振動しがちなデータのみを与えてしまったため、実際の最適化も振動しがちになってしまったということが考えられる。

また、これら3つについて5回ずつ実験を行い、 L_{eval} の2 sec 間の移動平均に関して、動作開始から0, 30, 60 秒後の値の平均と分散を計算し、Fig. 7.79 に示す。Fig. 7.79 の左上3つはそれぞれに関する移動平均であるが、5回全てに同じような傾向が見られる。0 sec における平均は最適化しないときが最も良く、Constant Search を使った場合が最も悪い。これは、道具使用初期は何も動作しない方が初期状態を保つことができるのに対して、Constant Search を使った把持安定化制御では動作が振動してしまいすぐに接触状態を崩してしまっていることが原因であると考えられる。しかし、60 sec 後は Variable Search を使った把持安定化制御が最も良く、把持安定化制御を入れない方法では徐々に接触状態が変化してしまっていることがわかる。

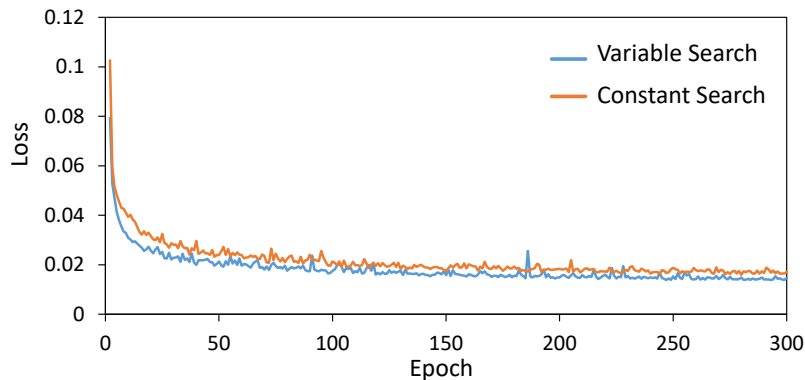


Fig. 7.77: Transition of training loss L_{origin} with data obtained using Variable or Constant Search [229].

掃除機操作実験

掃除機を前後に動かす動作を行う。以降では Variable Search を使った把持安定化制御を単純に把持安定化制御と呼び使用する。モデルは前述のように Variable Search を通して掃除機を持った状態で学習された。その結果を Fig. 7.80 に示す。70 sec の動作では、把持安定化制御を入れない場合は徐々に接触状態が変化し、かつ衝撃によって大きく振動している。それに対して、把持安定化制御を入れた場合は L_{eval} が一定値を保っていることがわかる。

箒操作実験

Musashi によって箒を掃く実験を行う。その際の L_{eval} の遷移を Fig. 7.81 に示す。把持安定化制御を入れた場合、入れない場合に関して両者とも初期に大きく L_{eval} がずれているが、把持安定化制御を入れた場合はその後一定の値に収束している。それに対して、把持安定化制御を入れない場合は 12 sec で L_{eval} が大きく上昇し、20 sec で箒掃きが失敗している。その際の様子を Fig. 7.82 であり、左手が箒から離れてしまったことがわかる。

また、この箒掃き実験を把持安定化制御を入れた場合と入れない場合に関して 60 秒間を 5 回ずつ行い、その際の成功・失敗と、失敗した場合の秒数を Table 7.6 に示す。把持安定化制御を入れた場合は 5 回中 3 回は 60 秒間の動きを成功させ、入れない場合は平均 28.6 秒で箒から手が離れてしまった。

7.7.4 議論

本研究の実験から、把持安定化制御を入れることで、道具把持を安定化することができることがわかった。ハンマー動作や掃除機使用においては、視覚的な変化は見られなかったものの、把持安定化制

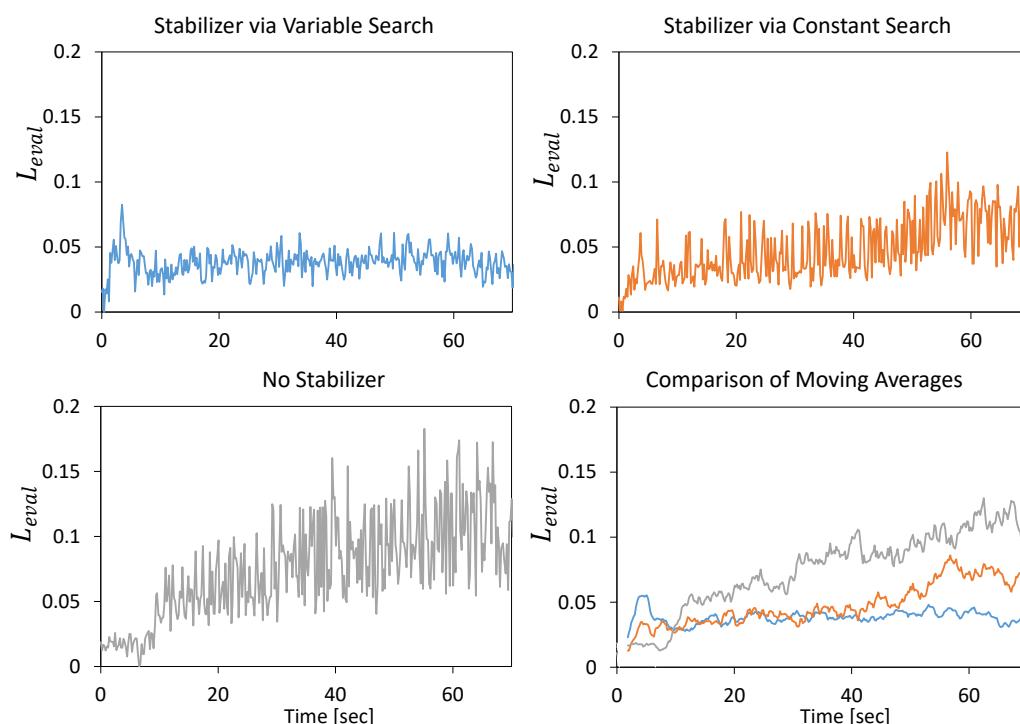


Fig. 7.78: Hammer hitting experiment: comparison of transition of L_{eval} among a stabilizer via Variable Search, a stabilizer via Constant Search, and no stabilizer [229].

Table 7.6: Quantitative evaluation of the brooming experiment [229]. \checkmark s express that the experiment succeeded over 60 sec, and the numbers express the time of failure.

Number of trials	1 st	2 nd	3 rd	4 th	5 th
with stabilizer	\checkmark	23	\checkmark	\checkmark	47
without stabilizer	20	33	4	52	34

御を入れることで、入れない場合に比べて初期接触状態を維持すること可能である。また、箒掃き動作においては、外力によって手が離れてしまうような失敗を、把持安定化制御を入れることによって緩和することが可能である。

しかし、いくつかの課題が残った。まず、親指の動きは、ランダムに動作させることで、通常人間ではあり得ないような把持・接触をしてしまう場合がある。これを解決するためには、単純に筋をランダムに動かすのではなく、親指に関わる筋のランダム性に制約を入れる必要があると考える。次に、箒タスクについて、把持安定化制御を入れても手が離れてしまう場合があった。これは床との摩擦による強い衝撃に耐えられなかったことが原因と考えられるが、それらの状況も含めたランダム探索を行い、よりデータを増やして対応していく必要がある。最後に、本研究では動作のスピードが遅いという

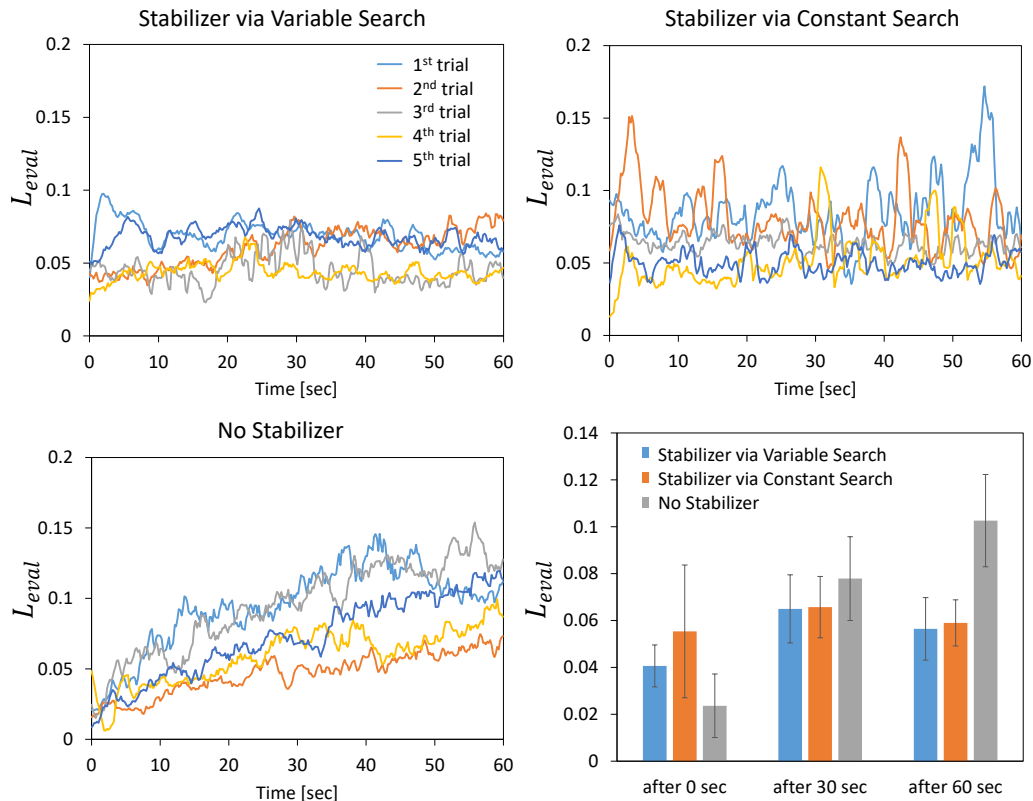


Fig. 7.79: Quantitative evaluation of moving average of L_{eval} after 0, 30, and 60 sec of hammer hitting with a stabilizer via Variable Search, a stabilizer via Constant Search, and without no stabilizer [229].

問題がある。よりダイナミックに動作させるためには、より制御周期を速くする必要があると考える。そのためには、ネットワークの簡易化による最適化の高速化等の工夫が今後必要になると考える。

本研究は、強化学習等における非常に探索空間の広い問題ではなく、把持安定化に着目することで、初期把持における接触状態の近辺を探索して状態遷移方程式を学習するのみで実機に適用することができる。そのため、3分程度の実機学習でも十分に効果を示すことができた。本研究では4本の筋長という制御入力を用いたが、より複雑なハンドでは探索空間が増え学習が難しくなるため、筋シナジー等の考え方を導入する必要があるかもしれない。

本研究で重要な点は、柔軟ハンドにおける非自明な接触センサや筋長センサと制御入力の間関係を暗に学習している点である。これまで多く行われてきた、それらをモデルから計算できるようなハンド以外にも本研究は適用可能であり、信頼性のある柔軟ハンドによる把持・道具操作等が期待される。

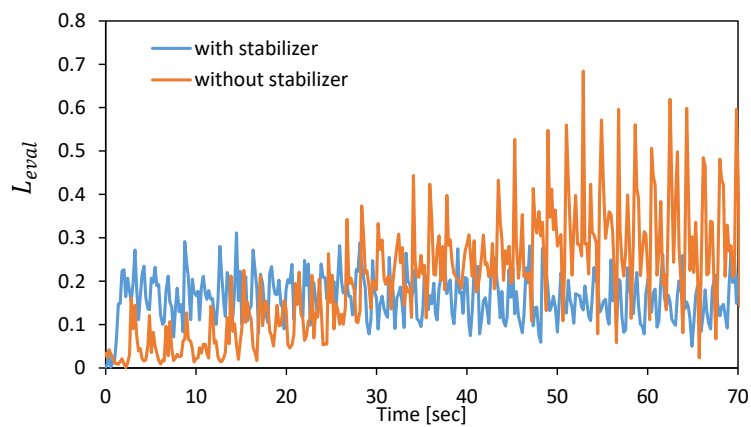


Fig. 7.80: Vacuuming experiment: comparison of transition of L_{eval} between with and without stabilizer [229].

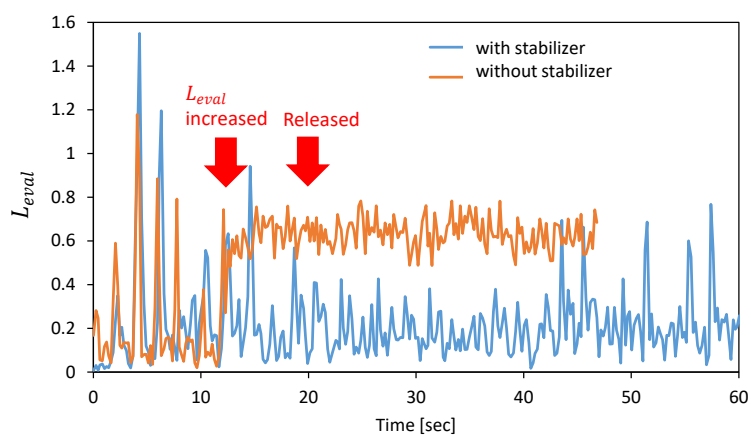


Fig. 7.81: Brooming experiment: comparison of transition of L_{eval} between with and without stabilizer [229].

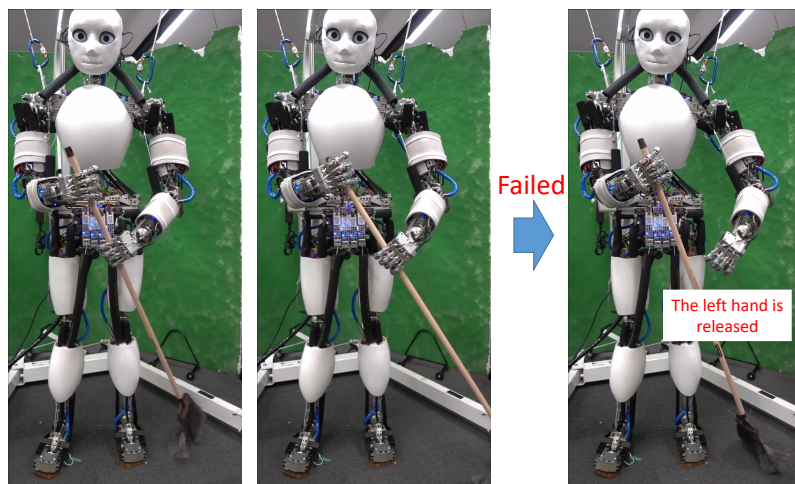


Fig. 7.82: The failure of brooming [229].

7.8 明示的パラメータを用いた動的体図式学習 - 筋温度制御

5.2 節において反射制御として述べた筋温度制御手法は、見方を変えることで動的体図式学習と捉えることができる。Eq. 5.9 と Eq. 5.10 をこれまでと同様なニューラルネットワークの構造で表現すると、Fig. 7.83 のようになる（これは実際にはニューラルネットワークではない）。

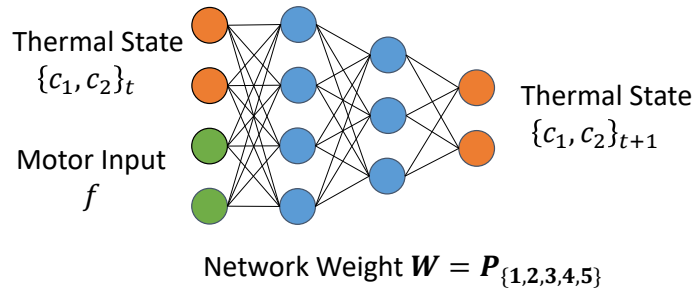


Fig. 7.83: Thermal model of Sec. 5.2 when expressing it as a neural network.

これは、現在のモータコア温度 c_1 とモータハウジング温度 c_2 、モータ入力 f を入力として、次時刻の $c_{1,2}$ を出力するような動的体図式と考えることができる。このネットワーク重み W は、実際には人間が明示的に変数化した熱モデルパラメータ $P_{1,2,3,4,5}$ に相当する。つまり、明示的パラメータを用いた体図式と言える。

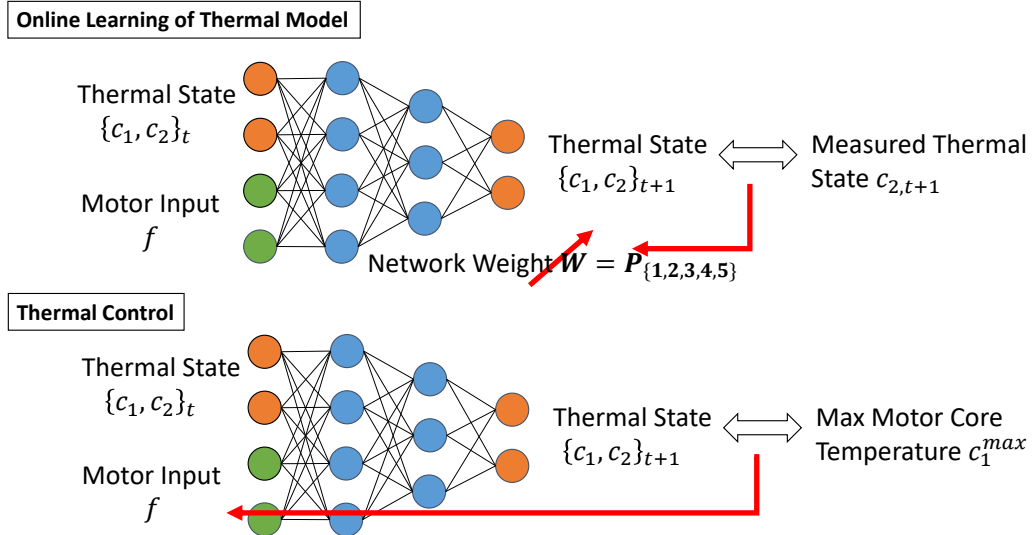


Fig. 7.84: The procedures of thermal model learning and thermal control using the thermal model of Fig. 7.83.

このモデルを使った温度モデルのオンライン学習、モータコア温度制御は、Fig. 7.84 のように表せ

る。温度モデルのオンライン学習は、現在測定可能なモータハウジング温度 c_2 について予測誤差を取り、これをもとに重み W を更新していく操作である。また、モータコア温度制御は、モータコア温度の予測列と定格のモータコア温度 c_1^{max} の誤差を取り、これをもとにモータ入力を更新していく操作である。このように、モデル化困難性の低い身体図式については、ニューラルネットワークではなく、直接状態方程式のパラメータを変数化する方法も考えられる。少数のパラメータのみ更新するため、ネットワーク全体の構造が破壊されず、オンライン学習を行っても過学習しにくいという特性がある。実験の詳細については5.2節に譲る。

7.9 本章のまとめ

本章では主に、一般化多感覚相関モデルの応用として、センサ・アクチュエータの増加と減少への対応、確率を出力とするネットワーク構成、平均と分散を出力する確率的ネットワーク構成、ネットワークの分割、時系列ネットワークの展開に基づく高速計算、明示的パラメータ化による身体図式学習の安定化について具体的な実験とともに述べた。

まず、筋骨格ヒューマノイドの冗長な筋が一本切れても関節を動かし続けられるという性質を最大限に利用した、学習型制御手法について提案した。関節角度・筋張力・筋長の冗長な相互関係を記述する Redundant Musculoskeletal AutoEncoder (RMAE) を構築し、これをオンライン学習させ、柔軟でモデル化困難な身体の状態推定・筋長制御に利用する。RMAE の予測誤差から異常検知を行い、筋を一本ずつ動かして筋破断確認を行うことで、筋破断を検知する。筋破断時には、RMAE のオンライン学習・状態推定・筋長制御のアルゴリズムを筋破断情報を元に変更することで、現状のネットワーク構造を壊さずに、現在の身体状態に適応していくことが可能である。筋破断情報の有無が顕著に制御・状態推定の誤差に影響を及ぼすことを実験から示し、一連の継続的動作実験により本研究の有効性を確認した。

次に、筋骨格ヒューマノイドにおける冗長筋駆動と容易な筋追加を活かした動作実現を行った。筋モジュール同士の結合を可能とすることで筋の増加を容易にし、タスクに応じた筋追加が可能となる。筋追加による身体図式の変化に対して自動でデータを取得、身体図式を再学習することで、追加した筋を積極的に使った動作が生成可能になる。この際、筋追加前のネットワークを保存しておき学習時に利用することで、少数のデータからでも効率的に身体図式を再学習させることが可能である。これらの全体システムを1自由度腿駆動シミュレーション、筋骨格ヒューマノイド Musashi の左腕の実機に適用し、その有用性を示すことに成功した。

次に、筋骨格ヒューマノイドの複雑な身体において危険回避を行うためのネットワーク構成とその応用について述べた。その複雑な身体において危険回避を行うためには実機データを用いたオンライン学習が有用である。筋張力に応じた安全機構が作動したときを危険状態と見なし、指令筋長から危険度を予測するネットワークを学習させていく。このネットワークを用いることで、危険を回避するように修正された指令筋長を計算したり、優先度付き逆運動学で危険な姿勢を回避したりすることが可能になる。

次に、確率的挙動・暗黙的な環境情報を含んだ予測モデル型のニューラルネットワーク SPNPB を提案し、これを用いた分散最小化を含む環境適応型ロボット制御を開発した。予測モデルにセンサ値の平

均と分散を出力させることで確率的挙動を埋め込み、様々な環境における動作データから Parametric Bias を自己組織化させることで環境情報を埋め込むことができる。環境情報を含む Parametric Bias をオンラインで更新し、出力の分散を最小化しつつタスクを実行する損失関数を設計して誤差逆伝播と勾配降下により制御入力を計算した。これにより、現在の動作環境に適応しつつ、動作の分散を最小化してより安定的な動作が実行可能であることを台車型ロボットのシミュレーション・実機において示した。

次に、冗長なセンサ・アクチュエータを全身に持つロボットについて、その機能的接続と空間的な接続から、それらを分割する手法について提案した。機能的接続を AutoEncoder を用いて獲得し、空間的接続と合わせてグラフ構造を作成することで、乱択アルゴリズムにより、関わりの少ないグルーピングごとに分割することができる。具体例として筋骨格ヒューマノイドの筋分割を挙げ、筋の拮抗関係からくる機能的な関係と、神経接続からくる空間的な接続をグラフに落とし込むことを行った。関節や筋配置の情報がない状態で、本手法によって拮抗関係を意識しつつ前腕や上腕、首や腰等の部位ごとに、筋を分割することが可能となった。分割する前に比べ、分割後は正確性をある程度保ったまま、解釈性の向上した、過学習しにくい構造を獲得できることを示した。

次に、筋骨格ヒューマノイドによる車のペダル操作を例に、タスク特化型の自己身体制御獲得手法について述べた。関節と筋の非線形な関係を学習するだけでは自己身体モデルと実機の誤差を完全に無くすることが難しいため、実際のタスク実現はより困難となる。そこで、現タスク状態から制御入力の時系列情報を入力し、次タスク状態の時系列を出力するようなネットワークを構築する。これをランダムな動作データから学習し、所望のタスク状態との誤差から入力に対する誤差逆伝播を用いることで最適な時系列制御入力を獲得可能な手法を開発した。これにより、ペダル操作において PID 制御等におけるゲインチューニングや最適制御におけるモデリングが不要となり、動的に所望の車速を安定して実現できることを確認した。今後は、モデリング困難な筋骨格ヒューマノイドにより、大きな一連のタスクを動的に実現する方法について考えたい。

次に、接触状態遷移の予測モデル構築と最適化に基づく道具把持安定化戦略について述べた。柔軟で劣駆動なハンドではセンサとアクチュエータの関係が一意に決まらないため、予測モデルを学習することで時系列に安定化のための制御入力を最適化することができる。ニューラルネットワークの誤差逆伝播を制御入力に対して用い、探索方法や損失関数の設計、最適化手法を工夫することで把持安定化が可能となる。特に、良い安定化戦略を得るためには、ランダムサーチにおける動きに緩急をつけると良い。また、接触センサの正負への値変化の異方性を考慮して損失関数を設計する必要がある。予測モデルは3分程度の探索行動で十分に構築することができ、その道具に適応した把持安定

化制御を得ることが可能である。

最後に、明示的なパラメータを使ったモータの熱モデル構築と、これによる熱モデルのオンライン学習とモータコア温度制御について述べた。モデル化困難性の低い身体図式については、人間が直接状態方程式の一部を変数化することで、過学習しにくい身体図式を構築することが可能である。

第8章

身体図式の逐次学習機能を有する知能ロボット
システムによる環境適応能力の向上

本章の概要を Fig. 8.1 に示す. 本章ではまず, 8.1 節において, 身体図式の逐次学習機能を有する智能ロボットシステムの全体像, これによる環境適応能力の向上について, その実例とともに本研究をまとめる. 次に, 身体図式学習や反射制御を含む智能ロボットシステムによる行動実現について述べる. 8.2 節では, 6.3 節における Musculoskeletal AutoEncoder を応用した筋骨格ヒューマノイドの動作修正手法について述べる. 8.3 節では, 7.6 節のペダル操作学習と 6.3 節の Musculoskeletal AutoEncoder, 画像認識や古典的な動作計画等を統合した智能ロボットシステムによる筋骨格ヒューマノイドの自動運転実験について述べる. 8.4 節では, 6.2 節の適応的道具先端操作学習を発展させた, 低剛性樹脂製ヒューマノイドにおけるバランスを考慮した全身道具操作学習と天窓を開ける統合実験について述べる. 8.5 節では, 6.3 節における Musculoskeletal AutoEncoder, 6.5 節における柔軟筋骨格ハンドの把持モデル学習を発展させた柔軟布操作学習, 画像認識や古典的な動作計画, 制御を統合した智能ロボットシステムによる, 筋骨格ヒューマノイドのテーブルセッティング実験について述べる. 最後に, 8.6 節において, 本手法の適用限界を示す意味で, 靴の変化を考慮した全身筋骨格ヒューマノイドにおけるバランス制御実験について述べる.

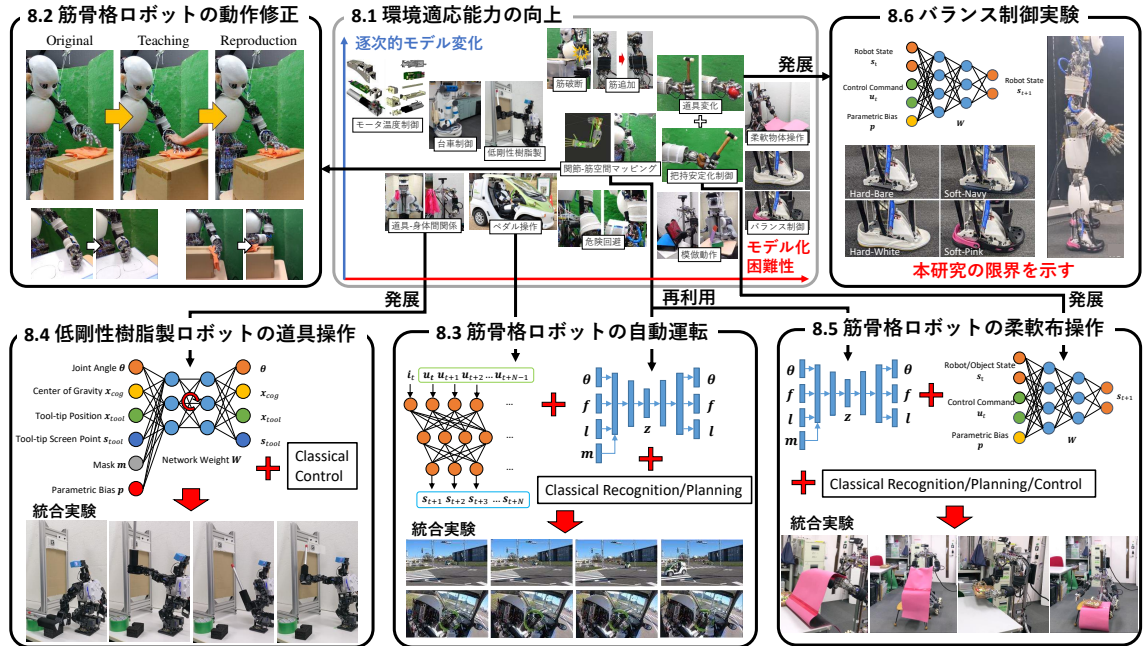


Fig. 8.1: The overview of this chapter.

8.1 身体図式の逐次学習機能を有する智能ロボットシステムによる環境適応能力の向上

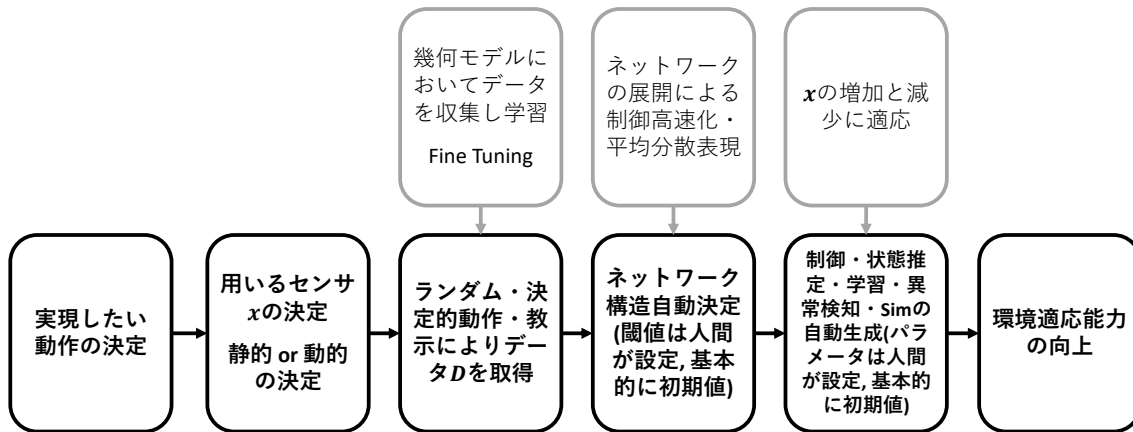


Fig. 8.2: The flow of using the developed intelligent robot system.

8.1.1 身体図式の逐次学習機能を有する智能ロボットシステム

本研究で開発した智能ロボットシステムの全体の流れを Fig. 8.2 に示す. まず, そのロボットにおいて行いたい動作を決定する. 次に, この動作に用いるセンサ・アクチュエータ群 \mathbf{x} と, この動作を静的に行うか, 動的に行うかを決定する. 次に, ランダム・決定的動作・教示によりロボットを動かし, データ D を取得する. この際, 幾何モデルにおいてデータを収集し一度学習してから, Fine Tuning することも可能である. 次に, 得られたデータ D からネットワーク構造の自動決定 (3.2 節) を行う. この際のパラメータは人間が設定するが, 基本的に初期値で良い. また, ネットワークの構造決定時に, ネットワークの展開による高速化や, 平均分散表現を出力に用いるか等のオプションを与えることも可能である. 次に, 制御・状態推定・オンライン学習・異常検知・シミュレーション等が自動生成される. この際のパラメータは人間が設定するが, 基本的に初期値で良い. また, \mathbf{x} の増加や減少に対応した学習をオプションとして与えることも可能である. これらにより, ロボットの環境適応能力が向上する.

以下に, これまでの実験から得られた身体図式学習を行う基準, 静的または動的身体図式の選び方, センサ・アクチュエータの選び方の基準について述べておく. また, 本研究において学習された身体図式の再利用性, 複数の行動目的に跨る利用について考察を述べる.

身体図式学習を行う基準

身体図式学習を使いたい場面は、柔軟性と冗長性を持つロボットの環境適応能力の向上を行いたい場合である、一方で、使いたいからといって、使った結果必ずうまく行くか、意味があるかというのは別の問題である。この身体図式学習を用いる判断基準として、以下の 3 つの基準が考えられる。

- 学習空間の広さ
- データ収集のしやすさ
- 代替手法の普及率

学習空間の広さが大きすぎると学習に必要なデータ量が著しく増大する。そのため、SLAM のような三次元点群を直接扱う場合や、動作計画のような探索空間の広すぎる場合に身体図式学習は難しい。また、データ収集のしやすさも重要である。バランス制御のようなデータ収集の際に倒れてはいけないう等の制約がある場合、データ収集は困難になり、本学習手法の適用が難しくなる。その他、ある程度連続的な長い動きをして初めて成功となるような動作への適用は難しく、これらはシミュレーションにおける強化学習等が適している。最後に、代替手法の普及率が重要である。SLAM や物体認識のアルゴリズムは非常に普及しており、その問題に特化した様々な工夫が成されている場合が多い。このような場合は身体図式学習を使う利点は大きくないと言える。

静的・動的身体図式の選び方の基準

身体図式を静的な形で構築するか、動的な形で構築するかを選択基準は単純である。動的な身体図式は静的な身体図式を包含するため、基本的には動的な身体図式を選ぶことが望ましい。一方で、静的に表現可能な身体図式を、動的に表すことによってネットワークに無駄が生じる。静的な身体図式の方が動的な身体図式に比べて過学習しにくく、必要なデータ数も少ない。つまり、基本的には動的な身体図式を考え、静的に表現できる場合についてのみ静的な身体図式を使う。

センサ・アクチュエータの選び方の基準

動作に用いるセンサ・アクチュエータ群 x の選択基準は以下の 3 つである。

- 指令値として実現したい値
- 制御入力として操作したい値
- 重複しないなるべく多くのセンサ値

まず, 制御を行う場合はその制御指令値として実現したい値を x に含める必要がある. 本研究は損失関数に含めることのできない値を扱うことはできないため, これは必須である. 次に, 制御入力として操作したい値を x に含めることも同様に必須である. 最後に, 上記 2 つと重複しないような, なるべく多くのセンサ値を x として利用すべきである. 入出力の自動決定機構によって unnecessary センサは除外されるため, 多い分には問題ない.

これまで行った動作, また, 本章で行う動作を合わせて, これらをモデル化困難性と逐次的モデル変化の観点から分類した図を Fig. 8.3 に示す. 以降でこれらをそれぞれの軸に分解して, 本研究で攻略した身体-道具-動作環境の相関複雑性と時間的変容について述べる.



Fig. 8.3: The classification of the achieved behaviors in terms of the modeling difficulty and model change over time.

身体図式の再利用性に関する考察

本研究の身体図式学習は基本的に行う動作を決め, それに必要なセンサ・アクチュエータを選び, データを収集, 学習を行う. そのため, 目的の行動に応じて必要なセンサ・アクチュエータが変わる場合は再利用が難しい.

一方で, このセンサ・アクチュエータが変わらない範囲に関しては, 目的が変わった場合も再利用が可能である. 例えば, 8.5 節において述べる布操作学習では, AutoEncoder で圧縮された画像と身体

の筋長・筋張力, また, 腕の制御入力を \mathbf{x} として扱っている. ゆえに, 画像・筋長・筋張力・腕の制御入力で表現可能なタスクは, 布操作に限らず学習データを増やすことで対応が可能である. 制御の際の損失関数を変更することで, 単純な腕の制御から, 布の制御, 道具の制御等を統一的行うことが可能である.

しかし, 現状は画像として赤を色抽出しマスクした2値画像を用いており, 赤い色の道具しか扱うことができない. また, 腕の制御入力は布操作に特化して矢状面上の2自由度としており, その範囲でしか腕の制御, 道具の制御等を行うことができない. つまり, 身体図式の再利用性を向上させるためには, \mathbf{x} として利用するセンサ・アクチュエータをより多次元かつ一般的な形式にする必要がある. 8.5節の例で言えば, 画像は何も前処理しない生画像, 制御入力は全身の関節角度等にする必要がある. 一方で, \mathbf{x} の一般化・多次元化は学習を困難にし, 必要なデータ数は指数関数的に増加する.

まとめると, 異なるタスクでも \mathbf{x} が同じであれば損失関数の形次第で身体図式は再利用することが可能であり, また, \mathbf{x} の一般化・多次元化による再利用性の向上と学習困難性の増大にはトレードオフがある.

8.1.2 身体図式の逐次学習機能による相関複雑性の攻略

Fig. 8.3 を, モデル化困難性 (相関複雑性) の軸でまとめた図を Fig. 8.4 に示す. まず, 最もモデル化が容易な例として, 7.8 節ではモータの熱モデルを扱った. ここでは, 直接熱モデルが数式として表現可能であるためニューラルネットワークは用いず, 熱モデルにおけるいくつかのパラメータのみを変数としたモデルを作成し, 学習を実行している. 次に, 軸駆動型ロボットについては, 関節モデル自体は非常に単純であるが, 後に述べる柔軟道具等の利用の観点からニューラルネットワークによる身体図式学習が行われた. 次に, 台車型のロボットについては, 環境との間の相互作用が複雑であり, そのモデルを確率的にとらえた動的な身体図式を構築することで, 分散最小化制御が可能となった. 次に, 低剛性樹脂製のヒューマノイドについては, その身体は把持した物体の重さによってたわみ方が大きく変化し, このたわみを学習することで正確な身体-道具制御が可能となった. 次に, 柔軟な道具については, 身体と道具の関係を身体図式により学習することで, その道具が柔軟性を持っていても, 正確に道具先端を制御することが可能となった. 次に, 筋骨格ヒューマノイドについては, その関節と筋の関係を身体図式により学習することで, 非線形弾性やワイヤの伸び, 変化するモーメントアーム等を考慮しつつ正確に動作することが可能となった. 次に, 柔軟ハンドについては, 関節や筋, 接触力の間の関係だけでなく, これに操作物体を含む複雑なダイナミクスが身体図式によりモデル化され, これを使った把持物体認識や把持制御, 把持検知等が可能となった. 最後に, 柔軟物体については, 筋骨格系

の柔軟な身体だけでなく、操作する物体までも柔軟であるような場合について、そのダイナミクスをモデル化し、制御することが可能となった。これらから、身体図式の逐次学習機能により相関複雑性が攻略できたと言える。

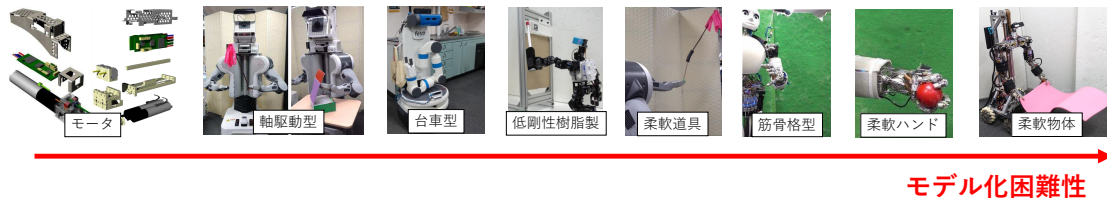


Fig. 8.4: The classification of the correlation complexity, in terms of the body, tool, and surrounding environment.

8.1.3 身体図式の逐次学習機能による時間的変容の攻略

Fig. 8.3 を逐次的モデル変化 (時間的変容) の軸について、身体・道具・動作環境を含む環境変容の分類でまとめた図を Fig. 8.5 に示す。まず、身体の変容には筋経路変化や経年劣化、筋破断・筋追加等による構造変化がある。7.8 節ではモータという身体変容の中でも最も低レイヤのモデル変容について扱った。劣悪な使用や経年劣化によって変化するモデルのオンライン学習を行っている。6.3 節、7.3 節では筋骨格構造における柔軟性による筋経路変化や経年劣化、再初期化による身体変容について扱った。身体変容に基づきオンラインでモデルを更新することで常に正確な身体図式を得ることができる。7.1 節、7.2 節では身体における筋破断や筋追加による構造変化について扱った。身体変容に基づく損失関数の変化やネットワーク構造の変化、そのオンライン学習により新たな身体構造に素早く適応することができる。次に、道具の変容には道具のプロパティ変化や身体-道具間関係の変化がある。6.5 節では道具変化による身体のセンサ間関係変化を Parametric Bias に埋め込み、これをオンラインで更新することで道具プロパティ変化に即座に対応することができる。6.2 節では道具-身体間の幾何学的関係変化を Parametric Bias に埋め込み、これをオンラインで更新することで身体と道具の位置関係を理解し、正確に道具を操作することができるようになる。最後に、周囲環境の変容には床の摩擦変化や対象物体の位置変化がある。7.4 節では台車型ロボットにおける床の摩擦変化、これに伴う台車のダイナミクスの変化について扱った。6.4 節では模倣学習と同様の枠組みで対象物体の位置・回転の変化について扱った。元々対象物体変化の情報を含む生の画像が入力として入ることで、Parametric Bias やオンライン学習の枠組みを用いなくても直接環境変容を扱えるようになる。

また、環境変容をその変化スパンの違いから分類した図を Fig. 8.6 に示す。筋破断や筋追加を含む突

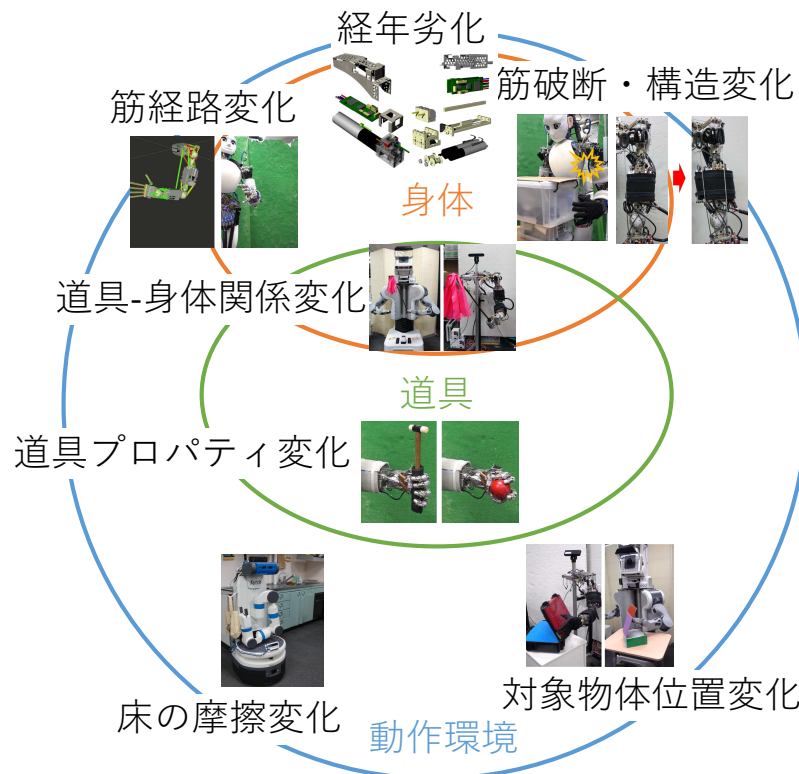


Fig. 8.5: The classification of the temporal change in terms of the body, tool, and surrounding environment.

発的な身体変化においては、ネットワーク構造自体が大きく変化する場合が多い。7.1 節, 7.2 節では、これをセンサマスクの導入による損失関数の変化やネットワーク重みのコピーと再学習により対応した。再初期化や身体-道具間関係変化、把持物体変化や床摩擦変化等の短時間における環境変容は、意図的に環境変容時のデータを取得しやすい。6.5 節や 6.2 節, 7.4 節では、この環境変容を Parametric Bias に埋め込み、短時間変容に対して Parametric Bias を更新することで対応した。関節-筋空間マッピングや筋の伸び、弾性体変化等の筋特性変化を含む中時間における環境変容は、短時間変化と比べると比較的長いスパンで特性が変化する。6.3 節や 7.3 節では、この環境変容を身体図式のオンライン学習により常に更新し続けることで対応した。最後に、モータ特性等のより長期的な環境変容においては、オンライン学習では過学習や破壊的なネットワーク構造変化の影響を受ける可能性が高い。5.2 節では、モータにおける熱モデルパラメータを直接変数化し、熱モデルの構造を保ったままモデルをオンラインで更新することで対応した。これらから、身体図式の逐次学習機能により時間的変容を攻略できたと言える。

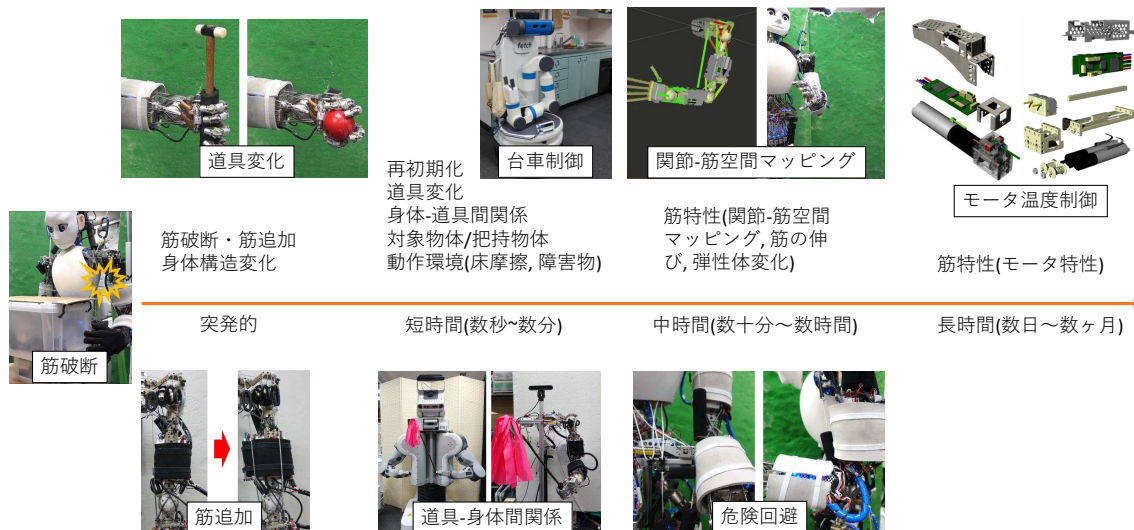


Fig. 8.6: The classification of the temporal change in terms of the span of the change.

8.2 静的身体図式を用いた動作修正手法

8.2.1 概要と先行研究

これまで筋骨格ヒューマノイドにおける様々な身体図式学習について述べてきた [100, 124, 125, 178]. これらの手法はある程度正しい指令関節角度を実現できる一方, 静的なモデルのみを扱っているため, ヒステリシスや骨格間摩擦等から, 誤差は必ず残ってしまう. この問題に対して [271] のように動的なモデルを扱う手法も開発されているが, より大きな空間の時系列は扱うことは難しい. また, 動作生成の際にロボットにおける認識誤差はつきものであり, 正確に動作できたとしても正確にタスクをこなせるとは限らない.

そのため本研究ではアプローチを変え, Fig. 8.7 のように, 筋骨格ヒューマノイドの動作中に外力を加えることでその動作を意図したものに修正し再生する方法について考える. 筋骨格ヒューマノイドはその柔軟な身体から, 制御なしに簡単に外から動きを変化させることができるため, 人間による教示との親和性が高いと考えられる. しかし同時に, 通常の軸駆動型ロボットと違う構造であるため, 通常の教示 [285] 手法を用いることはできない. これは, 通常筋骨格ヒューマノイドは球関節や複雑な肩甲骨等の存在から関節角度を直接測定するセンサがないこと, そして, 柔軟性と可変剛性制御を担保する非線形弾性が筋末端に備わっているため, 外力による動作の変位がアクチュエータ側ではなく, 非線形弾性要素側に伝達されること, の二点が大きな要因である. そのため, 外力を加えて修正した動作を, 同じように再生することが難しい. また, 筋骨格ヒューマノイドにおいても [101] のような Direct Teaching や, [286] のウェアラブルデバイスを使った教示方法が開発されているが, どちらも動作中に外部から力を受けることを考慮した手法ではない. 本研究では, 関節角度を介さず教示中の筋張力を用いて, 筋レベルでの補償制御を行うことで, 修正された動作を正確に再現する手法について提案する. 筋骨格ヒューマノイド Musashi [87] に本手法を適用し, 基本的な比較実験の他ボックス拭きや絵描きの動作を行うことで, 本研究の有効性を確認する.

8.2.2 筋長補償制御による動作修正

まず, 本研究における人間の教示による動作修正の一連の流れを以下に述べる.

- (A) ロボットの動作を作成する.
- (B) 動作中に人間が外力を加えることでその動作を修正する.
- (C) 修正された動作を再生する.

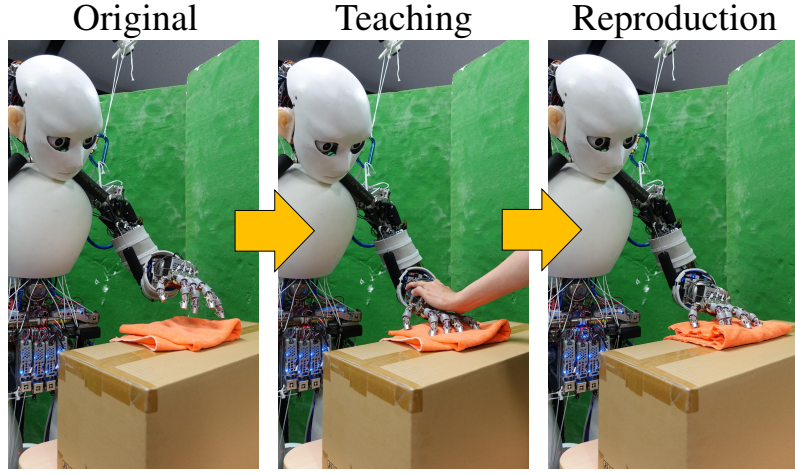


Fig. 8.7: The overall flow of this study: teaching by humans during the original movement and its reproduction.

(A) は通常人間が動きをプログラムしたり, 認識等の結果から動作を生成したりする. 指令関節角度 θ_i^{ref} が決定され, CTRL (以降に述べる) により指令筋長 l_i^{ref} , 指令筋張力 f_i^{ref} が計算される. (B) において, 動作中に測定された筋長 l_i^{data} , 筋張力 f_i^{data} を蓄積しておく. また, (B) では筋骨格構造の元々の柔軟性を利用することも可能であるが, 後に説明する筋張力制限制御により筋張力の最大値を制限することでより外力の効果を大きくすることも可能である. (C) において, 得られた $\{l, f\}_i^{\{ref, data\}}$ を用いて l_i^{ref} から変化させるべき筋長 Δl_i^{ref} を計算し, $l_i^{ref} + \Delta l_i^{ref}$ を指令することで修正された動作を再生する (これは後の筋長補償制御において説明する).

ここで簡単に, 本研究で提案する筋補償制御が併用する Musculoskeletal AutoEncoder (MAE) におけるいくつかの操作を説明する [178]. MAE における h_θ を用いることで, 現在の推定関節角度 θ^{est} を (f, l) の情報から計算する操作を EST と呼ぶ. また, h_l を用いることで, ある指令値 θ^{ref} を実現するための筋長 l^{ref} を計算する操作を CTRL と呼ぶ. なお, ここでは筋張力の指令値 f^{ref} を計算する必要があるが, これは重力補償トルク τ と誤差逆伝播法を用いた反復計算により決定される. MAE は静的なセンサ関係のみ表しているため, ヒステリシス等の問題から, EST により完全に正確な θ^{est} が推定できるわけではなく, CTRL により, 完全に正確な θ^{ref} が実現できるわけでもないことに注意されたい.

筋張力制限制御

それぞれの筋について、筋張力制限制御 (muscle tension limiter) は以下のように筋張力に応じて筋長弛緩度 $\Delta l_{e,t}^{ref}$ を計算し、実機には $l_t^{ref} + \Delta l_{e,t}^{ref}$ を指令する。

$$\begin{aligned}
 & \text{if } f_t > f^{max} \\
 & \quad \Delta l_{e,t}^{ref} = \Delta l_{e,t-1}^{ref} + \min(C_{gain}d - \Delta l_{e,t-1}^{ref}, C_{plus}d) \\
 & \text{else} \\
 & \quad \Delta l_{e,t}^{ref} = \Delta l_{e,t-1}^{ref} + \max(0 - \Delta l_{e,t-1}^{ref}, -C_{minus}d) \\
 & \quad d_t = |f_t - f^{max}|
 \end{aligned} \tag{8.1}$$

ここで、 f^{max} は筋長を弛緩させ始める筋張力 f の閾値、 $C_{\{minus,plus\}}$ はマイナス方向またはプラス方向に対する一ステップの筋長変化量を決める係数、 C_{gain} は最大弛緩量を決める係数である。つまり、 $C_{minus}d_t$ 、 $C_{plus}d_t$ で制限をかけながら、筋張力が最大値を越えないように筋を弛緩・緊張させている。外から力を加えた場合筋張力が高まり、 f^{thre} を超えた分だけ筋が伸びることで、より簡単に指示により大きく動作を修正することができるようになる。本研究では、 $C_{minus} = 0.001$ [mm/N]、 $C_{plus} = 0.003$ [mm/N]、 $C_{gain} = 2.0$ [mm/N] とし、本制御は 8 msec 周期で行う。また、 f^{thre} は実験に応じて変化させる。

筋長補償制御

通常の軸駆動型ヒューマノイドであれば、修正された動作の関節角度 θ_t^{data} を保存しておき、それを指令値 θ_t^{ref} として実機に送るだけで良い。しかし、概要で述べたように筋骨格ヒューマノイドは通常関節角度 θ_t^{data} を直接測定できないため、そのような方法は現実的でない。EST により推定された関節角度 θ_t^{est} を用いることもできるが、EST による推定と CTRL による計算の二段階を通すため MAE の誤差が蓄積してしまう (実験章において比較実験を行う)。一方、修正された動作の筋長 l_t^{data} を取っておき、それを指令値 l_t^{ref} として実機に送る方法を取ることが考えられる。しかし、外力の影響は l_t^{data} ではなくハードウェアである非線形弾性要素や筋自体の伸びに出てしまう。ゆえに、測定された筋長を送っても外力を上手く反映することができない。そこで、本研究では Musculoskeletal AutoEncoder から得られる情報を元に、筋ベースで筋長を補正する項 Δl_t^{ref} を計算し、それを足しこむことで修正された動作を再生することを考える。

元々の筋長指令 l_t^{ref} に加えて、以下の (A)-(C) の項を加えることで動作を再生する手法を提案する。これは、(A) 筋張力制限制御による伸び $\Delta l_{e,t}^{ref}$ 、(B) 非線形弾性や筋自体のハードウェアによる伸び $\Delta l_{h,t}^{ref}$ 、(C) 筋剛性制御のソフトウェアによる伸び $\Delta l_{s,t}^{ref}$ 、となっている。

(A) は単純で、Eq. 8.1 における $\Delta l_{e,t}^{ref}$ 分だけ l_t^{ref} に加えれば良い。

(B) は MAE を用いてハードウェアによる筋の伸びを以下のように計算する。

$$\begin{aligned}\theta_t^{est} &= h_{\theta}(f_t^{data}, l_t^{data}) \\ \Delta l_{h,t}^{ref} &= -(h_l(\theta_t^{est}, f_t^{data}) - h_l(\theta_t^{est}, f_t^{ref}))\end{aligned}\quad (8.2)$$

これはつまり、同じ関節角度において、筋張力を変化させたときに、どの程度ハードウェアとしての筋長が変化するかということを表している。この計算は、(i) 重力補償に必要な筋張力と (ii) 筋のハードウェア弾性の特性は、 θ^{ref} と θ^{data} の間で大きく変わらないという性質を利用している。教示の際の外力により筋張力が高まるが、動作を再生するときには、(i) の仮定において元の f^{ref} に近い筋張力を発揮するようにする。また、(ii) の仮定を置くことで、Eq. 8.2 の計算式を簡略化できている。実際、 $h_l(\theta, f)$ は [124] のように $h_1(\theta)$ と $h_2(\theta, f)$ に分解することができる。 h_1 は f が働いていないときの θ における筋長を表し、 h_2 は f が働いているときに θ を保つための筋長のハードウェアとしての伸びの補正項を表している (常に負の値を示す)。よって、 θ として同じ θ_t^{est} を代入することで、Eq. 8.2 において h_1 は打ち消し合い、その θ 回りの筋のハードウェアとしての伸びを補償する項の変化 $h_2(\theta_t^{est}, f_t^{data}) - h_2(\theta_t^{est}, f_t^{ref})$ 、つまり筋の伸びの変化のみが計算できる。しかしこれはあくまで補償項であり負の値なため、符号を逆にする必要がある。

(C) は Eq. 2.4 の筋剛性制御の式から以下のように筋の伸びを計算する。

$$\Delta l_{s,t}^{ref} = -(l^{comp}(f_t^{data}) - l^{comp}(f_t^{ref}))\quad (8.3)$$

これも (B) と同じように (i) の性質を利用して、ソフトウェアによる筋の伸びを考慮している。また、補償項であり負の値なため、符号を逆にする必要がある。

最後に、8.2.2 節における (A)-(C) は、Fig. 8.8 のように図式化すると非常に面白い関係性を表している。Original と Modified の間では、 l^{ref} を共通として、外力によって θ^{ref} と f^{ref} の間に差が出ている。また、Modified と Reproduction の間では、 θ^{ref} を共通として、 f^{ref} と l^{ref} の間に差が出ている。本研究では、Original と Reproduction の間の筋張力は大きく変わらないという (i) の仮定を置くことで、Original と Reproduction の間に似た関係を成り立たせている。ゆえに、それぞれの動作間において、一つの値を固定して、残りの二つの値の関係性を変化させるという構図が成り立つ。通常であれば

Reproduction において θ^{data} と f^{ref} から $l^{ref} + \Delta l$ を求める (Joint-based) ところだが, θ^{data} が得られず f^{data} がわかっているという性質を利用することで, $l^{ref} + \Delta l$ を求めている (Muscle-based) という図式も理解できる.

なお, 本研究におけるタイムスタンプ t は 0.2 sec 間隔で取得する.

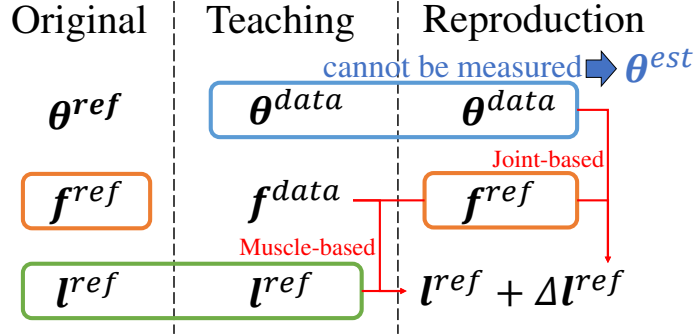


Fig. 8.8: The relationship of (θ, f, l) among original, teaching, and reproduction.

制御手法の比較

本研究の評価実験における評価する手法群を以下に列挙する.

- ALL: $\Delta l_t^{ref} = \Delta l_{e,t}^{ref} + \Delta l_{h,t}^{ref} + \Delta l_{s,t}^{ref}$
- W-HS: $\Delta l_t^{ref} = \Delta l_{h,t}^{ref} + \Delta l_{s,t}^{ref}$
- W-ES: $\Delta l_t^{ref} = \Delta l_{e,t}^{ref} + \Delta l_{s,t}^{ref}$
- W-HE: $\Delta l_t^{ref} = \Delta l_{h,t}^{ref} + \Delta l_{e,t}^{ref}$
- W-H: $\Delta l_t^{ref} = \Delta l_{h,t}^{ref}$
- W-E: $\Delta l_t^{ref} = \Delta l_{e,t}^{ref}$
- W-S: $\Delta l_t^{ref} = \Delta l_{s,t}^{ref}$
- NONE: $\Delta l_t^{ref} = 0$
- THETA: EST により θ_t^{est} を求め, CTRL によりそれを実現する l_t^{ref} を求める手法

8.2.3 実験

実験セットアップ

本研究では筋骨格ヒューマノイド Musashi [87] を用いる. Dyneema の筋ワイヤの末端には非線形弾性要素が配置されている. 本研究では主に左手の肩の 3 自由度と肘の 2 自由度を用いて実験を行

い、これらは S-p, S-r, S-y, E-p, E-y と呼ぶ (S, E は shoulder, elbow, rpy は roll, pitch, yaw を表す)。また、Musashi は通常の筋骨格ヒューマノイドと違い、実験評価のために、関節角度を直接測定可能な機構が備わっている。実験の中ではこの値は用いないが、修正された動作を正確に再現できているかどうかを評価するために用いる。

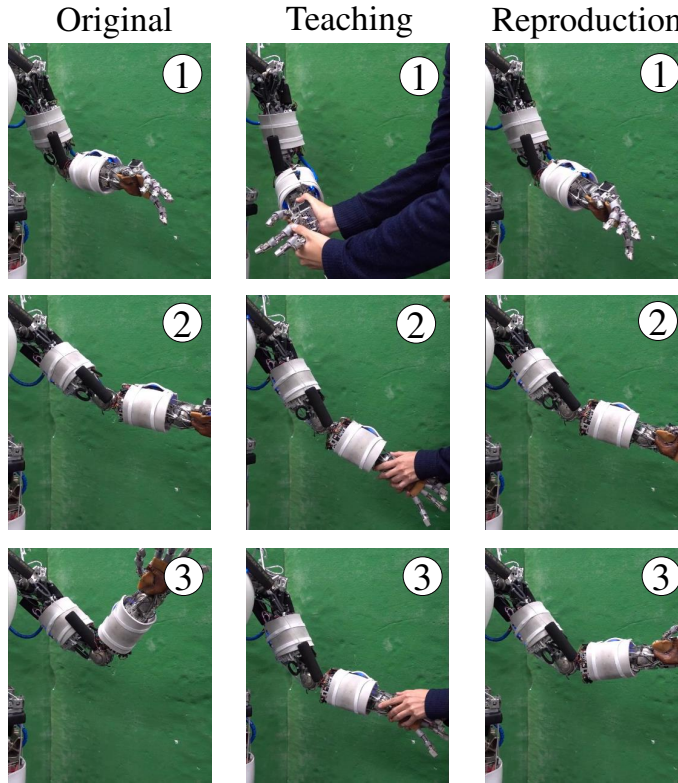


Fig. 8.9: The procedure in the comparison experiment of motion modification methods.

比較実験

本実験では 8.2.2 節で示した 9 つの手法の性能を比較検討する。Fig. 8.9 の左図に示すような基本動作を記述する。次に、中図のように外部から人間がそれを修正する。最後に、右図のように 8.2.2 節のそれぞれの手法で修正された動作を再生する。このとき、中図における教示中の関節角度 θ_t^{data} と、下図の再生時の関節角度 θ_t^{rep} を比較する。また、 $|\theta_t^{data} - \theta_t^{rep}|$ の全時間の平均を E とする。Fig. 8.9 は本節後半の実験であり、Reproduction は ALL を使った場合を示している。本実験では外力によってベースリンクが動いてしまっているため、画像による比較は参考程度である。

まず、筋張力制限制御を用いない場合における $|\theta_t^{data} - \theta_t^{rep}|$ の推移を Fig. 8.10 に示す。14 秒間の動

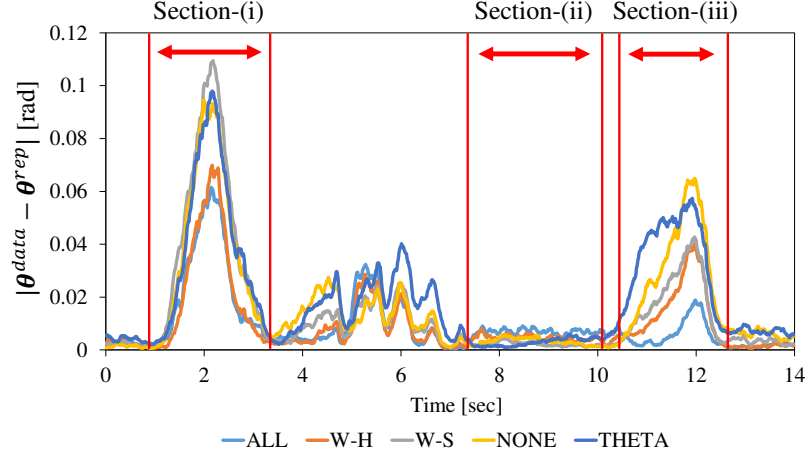


Fig. 8.10: Transition of $|\theta_t^{data} - \theta_t^{rep}|$ in the comparison experiment without muscle tension limiter.

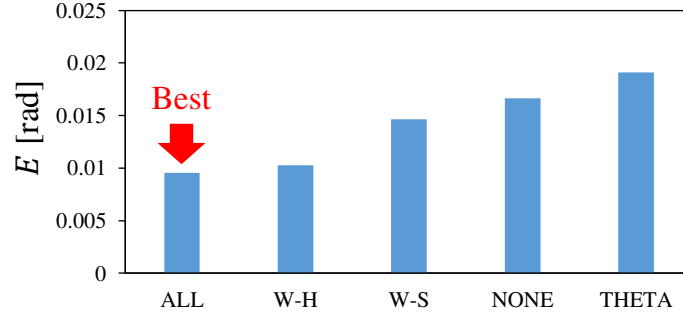


Fig. 8.11: Comparison of evaluation value E in the comparison experiment without muscle tension limiter.

作において、Section-(i) から Section-(iii) までの区間に着目する。Section-(i) では、精度は $ALL \approx W-H > W-S \approx NONE \approx THETA$ となっている。Section-(ii) では、それぞれほとんど同じ精度であるが、 ALL の精度が最も悪い。Section-(iii) では、精度は $ALL > W-H \approx W-S > NONE \approx THETA$ となっている。つまり、Section-(i) では $\Delta I_{h,t}^{ref}$ による影響が大きく、 $\Delta I_{s,t}^{ref}$ による影響が小さい。そのため、 $ALL \approx W-H$ となっており、逆に $W-S \approx NONE$ ともなっている。THETA は $NONE$ とほぼ変わらず、精度は低い。また、Section-(iii) では $\Delta I_{h,t}^{ref}$ と $\Delta I_{s,t}^{ref}$ の影響はほぼ同じ程度であることがわかる。その二つの影響を考慮した ALL が最も良い性能を表している。これに対して、Section-(iii) では大きな差ではないにしろ、 ALL の精度が最も悪い。あまり大きな外力が加わらず全体的に $|\theta_t^{data} - \theta_t^{rep}|$ が低い場合は、MAE や筋張力計測の誤差の方が支配的となっていると考えられる。これらをまとめたそれぞれの手法の E の値の比較を Fig. 8.11 に示す。精度の結果は $ALL > W-H > W-S > NONE > THETA$ となっており、THETA と比べると ALL の誤差は 1/2 程度になっていることがわかる。

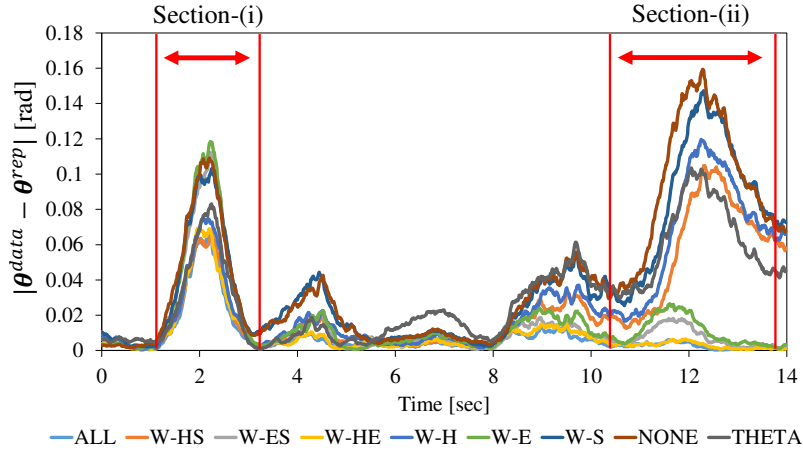


Fig. 8.12: Transition of $|\theta_t^{data} - \theta_t^{rep}|$ in the comparison experiment with muscle tension limiter.

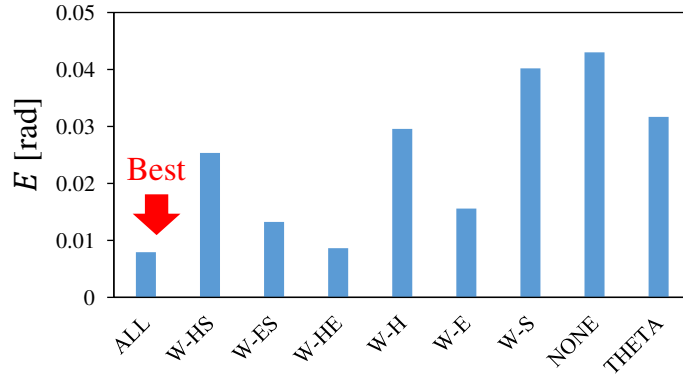


Fig. 8.13: Comparison of evaluation value E in the comparison experiment with muscle tension limiter.

次に、筋張力制限制御を用いた場合における $|\theta_t^{data} - \theta_t^{rep}|$ の推移を Fig. 8.12 に示す。なお、 $f^{thre} = 100$ [N] とする。同様に、Section-(i), Section-(ii) の区間を設ける。Section-(i) では精度は、 $ALL \approx W-HS \approx W-HE \approx W-H \approx THETA > W-ES \approx W-E \approx W-S \approx NONE$ となっている。Section-(ii) では精度は、 $ALL \approx W-ES \approx W-HE \approx W-E > W-HS \approx W-H \approx W-S \approx NONE \approx THETA$ となっている。つまり、Section-(i) では $\Delta I_{h,t}^{ref}$ の影響が最も大きかったのに対して、Section-(ii) では $\Delta I_{e,t}^{ref}$ の影響が最も大きかったことが読み取れる。THETA は区間によっては良いこともあるが、全体として見ると良いとは言えない。これらをまとめたそれぞれの手法の E の値の比較を Fig. 8.13 に示す。精度の結果は $ALL > W-HE > W-ES > W-E > W-HS > THETA > W-S > NONE$ となっており、THETA と比べると ALL の誤差は 1/3 以下になっていることがわかる。全体的な影響度合いは $\Delta I_{e,t}^{ref} > \Delta I_{h,t}^{ref} > \Delta I_{s,t}^{ref}$ であることも読み取れる。また、THETA の誤差は W-H と比べて大きな差は無かった。

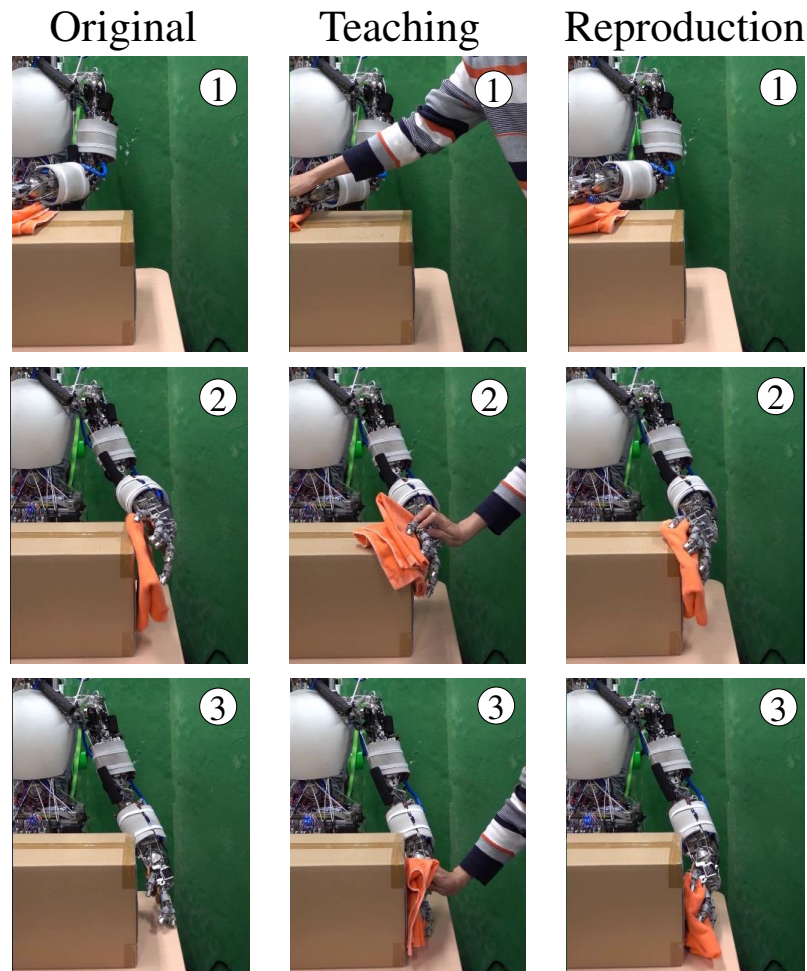


Fig. 8.14: Box wiping experiment.

箱拭き実験

本研究の手法を用いて Box の二平面を拭く動作を行う。本実験では筋張力制限制御は実行していない。Fig. 8.14 にその流れを示す。まず人間が逆運動学から Box を拭く動作を生成する。しかし関節角度の誤差によって、縦の平面を拭く際に手が箱から離れてしまい、雑巾が落ちてしまっていることがわかる。次に、その動作中に人間がそれをガイドするように力を加え、正しく箱の側面が拭けるように動作を修正する。最後に、人間のガイド無しに ALL を用いて動作を再生した結果、箱の側面に手を押し付け、最後まで拭くことができていたことが確認できた。

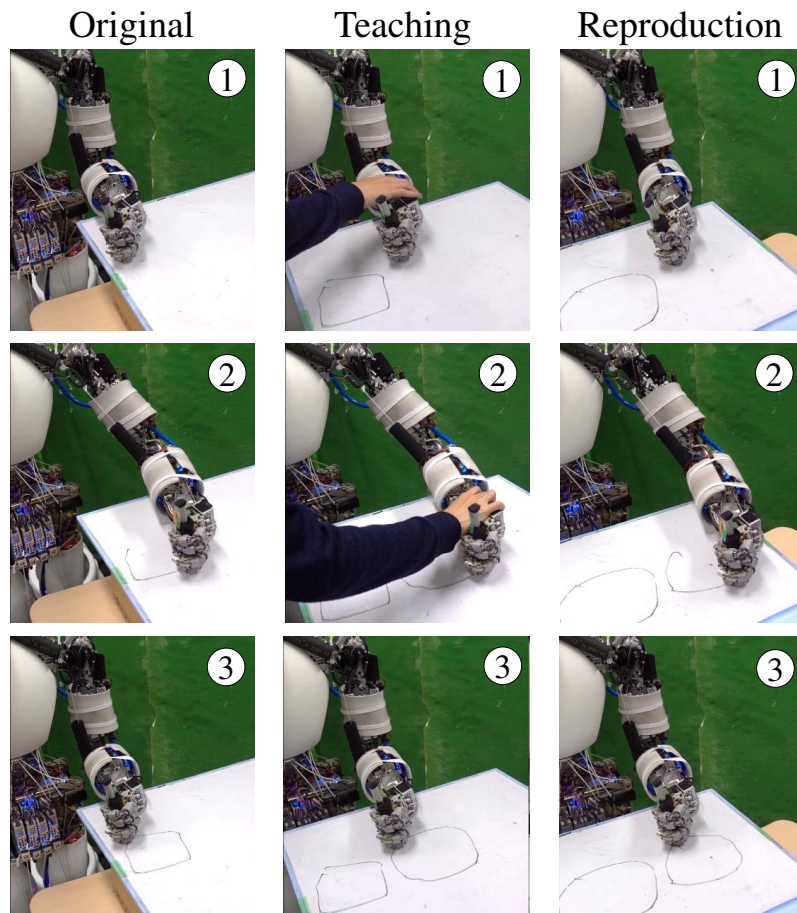


Fig. 8.15: Drawing experiment.

描画実験

本研究の手法を用いて四角を描く動作を、丸を描く動作に修正することを行う。なお、本実験では $f^{thre} = 200$ [N] として筋張力制限制御を実行している。本実験の流れを Fig. 8.15 に示す。まず、逆運動学を用いて四角を描く動作を生成する。次に、人間がそれを外からガイドし、丸を描く動作に変更する。最後に、それを本手法の ALL を用いて再生する。最終的な結果を Fig. 8.16 に示す。初期はおおよそ四角を描くことができていたのに対して、教示時は丸、また、再生時はそれと似た丸を描くことができていたことがわかる。

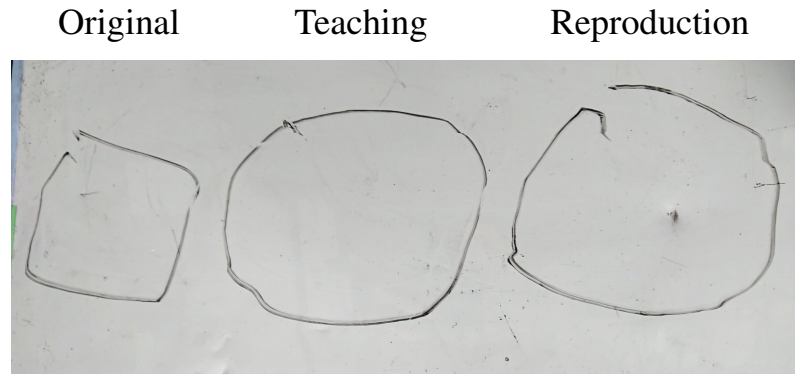


Fig. 8.16: The result of drawn objects.

8.2.4 議論

比較実験では、本手法を構成する $\Delta I_{e,t}^{ref}$, $\Delta I_{h,t}^{ref}$, $\Delta I_{s,t}^{ref}$ の項の影響について様々な知見を得ることができた。筋張力制限制御を用いない場合は $\Delta I_{e,t}^{ref}$ の影響は無くなるが、 $\Delta I_{h,t}^{ref}$ が支配的な区間、 $\Delta I_{h,t}^{ref}$ と $\Delta I_{s,t}^{ref}$ の影響が同程度の区間が見受けられた。これは、非線形弾性による影響が大きい区間と、非線形弾性による影響が小さく、ソフトウェアによる筋の伸びと同程度である区間があることに相当する。非線形弾性は経年劣化するため、徐々にクセがつき、筋張力による伸び変化が小さくなることもあり、これが影響していると考えられる。また、最初から高い筋張力がかかっている筋の場合は、その非線形性により筋張力による筋の伸びは徐々に小さくなるため、外力による伸びの影響を受けにくくなる。そして、これらハードウェアとソフトウェアの影響を考慮できる ALL が最も良い結果となり、本手法の妥当性が示された。同時に、外力の影響が少ない区間では ALL の精度が最も悪く、様々な要素を考慮できると同時に、誤差の影響も受けやすいということが示された。また、完璧に教示した関節角度を再生できるとは限らず、これは主に、MAEにおいて指令筋張力が実機によって完全に達成されるという仮定が必ずしも成り立たないことを示している。MAEは静的な要素しか考慮できないため、筋の摩擦やヒステリシスの影響で、誤差が出てしまっている。

筋張力制限制御を入れた場合は、 $\Delta I_{e,t}^{ref}$ と $\Delta I_{h,t}^{ref}$ の影響が支配的であることがわかった。区間によって、それぞれの影響度合いが変化している。元々高い筋張力を発しているときに強い外力が加わると筋張力が f^{thre} によって制限され、 $\Delta I_{e,t}^{ref}$ が支配的になる。逆に、元々の筋張力が低い場合には外力が加わっても $\Delta I_{e,t}^{ref}$ に達せず、 $\Delta I_{h,t}^{ref}$ が支配的になる。全体として見れば、その影響度合いは、 $\Delta I_{e,t}^{ref} > \Delta I_{h,t}^{ref} > \Delta I_{s,t}^{ref}$ であることもわかった。そして全体としては、全要素を考慮可能な ALL が最も良い精度を示しており、本手法の妥当性が示された。また比較実験を通して、THETAの精度は低く、筋

骨格ヒューマノイドにおいて推定関節角度を直接介して動作再生を行うのは得策ではないということがわかった。

箱拭き実験と描画実験では、本研究の実用的な有効性が示された。ここで特筆すべきは、描画実験では全ての筋に本手法を適用すると上手く絵を描くことはできなかったことである。肘の筋は大きな摩擦のため MAE の誤差が大きく、動作再生時に手が持ち上がってしまい、ペンがホワイトボードに当たらないということが起きた。そこで、肘の 2 本の筋のみ本手法を適用せずに実験を行うと、正確に教示された動作を再生することができた。つまり、本手法は MAE の誤差、つまり非線形弾性特性の推定値に敏感であるため、その値を正しく推定できるよう、学習させる必要がある。

最後に、本研究の適用範囲について議論する。本研究では主に、人間が関節角度を直接指定して動作をさせる、または、逆運動学で単純な動作を生成した後それを修正するという方法を取った。同様に、人間が教示デバイスを使って動かした際の動作を微妙に修正することもできる。また、筋骨格ヒューマノイドは関節角度を測定できないため、筋の原点の初期化が難しい。そのため、初期化を行った後に動作が微妙に変化することがあるが、これらを修正することにも使用できる。

8.3 筋骨格ヒューマノイドによる自動運転実験

8.3.1 概要と先行研究

現在, 安全で快適な移動手段として, 自動運転技術に関する様々な研究開発が行われている [287, 288]. 実際に複数の企業において, 渋滞時の追従走行や自動駐車等が実現されている. これら自動運転車には, 強力なカメラ・レーダー・GPS・プロセッサ等のシステムが搭載されており, これらを使って車を制御する.

一方で, DARPA Robotics Challenge (DRC) [4] における車走行タスクを初めとして, ヒューマノイドロボットによる自動運転の研究も存在する [289, 290]. ヒューマノイドロボットは視覚や聴覚, 力覚等に関する様々なセンサを搭載しており, これらを用いて車に乗り込み, 運転を人間の代わりに行うことができる. また, 自動運転車とは違い, ヒューマノイドロボットはその他荷物運びや高齢者のアシスト, 家事や災害対応等への応用が期待されており, 一台で様々なタスクを成し遂げられると考えられている. しかし, これらのヒューマノイドは柔軟性に欠け, かつ身体構造が人間からいくらか逸脱しているため, DRC においてハンドル操作は全て片手で行っており, 座るのにも特殊なジグを要する等の問題点があった.

ヒューマノイドロボットにも様々なタイプが存在するが, その中でもより人間の駆動システムを模倣したのが筋骨格ヒューマノイドである [85, 171, 86, 87]. 筋骨格ヒューマノイドは通常のロボットのような軸ごとに配置されたモータではなく, 筋肉を模した空気圧やモータによる腱駆動アクチュエータを用いて身体が駆動されている. その身体は軸駆動型ヒューマノイドに比べ, 筋の弾性や劣駆動性による柔軟性を持ち, 環境に身体を接触させて動作を行うのに適している. そのため, より容易に人間の使う実際の車に乗り込むことができ, DRC におけるロボットの特殊な座り方やそのためのジグ, 片手によるハンドル操作等を解消することができると考える. また, 人間同様の身体システムは人体シミュレータとしての活用も考えられ, 自動運転に関する衝撃シミュレータ [291] への応用が挙げられる.

本研究では, 筋骨格ヒューマノイドによる自動運転実現のための我々の取り組みを紹介する (Fig. 8.17). 特に, 自動運転に着目した筋骨格ヒューマノイドのハードウェア特徴, それらを動作させるための学習型ソフトウェアの特徴について主に述べる. これまで開発してきたハードウェア [87, 112, 116, 157]・ソフトウェア [125, 271, 189] を自動運転というコンテキストで捉え直し, 認識等を含む全体システムを開発する. また, これらのハードウェア・ソフトウェアを用いた自動運転に関する個々の動作実現, それらを統合した運転実験について述べていく.



Fig. 8.17: Autonomous driving by musculoskeletal humanoids

8.3.2 筋骨格ハードウェアと自動運転

Musashi の非線形弾性要素, 頭部, 手, 足に関して, 自動運転という観点に着目した利用方法を Fig. 8.18 に述べる.

非線形弾性要素を用いることで, 身体を柔軟にするだけでなく, その柔軟さを自在に変更することができるようになる. (a) に示すように, 柔軟な身体構造によって, DRC では見ることのなかった両手によるハンドル操作が可能である. また, 身体の剛性を変化可能なため, (b) のように, 衝撃に対する応答を自在に変化させることができる. 手先剛性を小さく設定したとき, 大きく設定したときに 1 m 上から 5 kg のボールを落とすと, 前者の最大筋張力は 150 N, 後者は 250 N となった. これは車衝突時に人間が力を入れて剛性を高める動作にも似ており, 人体シミュレータとしての利用が期待される.

可動眼球ユニットを有する頭部を用いることで, (c) のような広い視野範囲による認識行動, また, (d) のような高解像度なカメラを合わせてサイドミラー等に映る車や人の認識も行うことができる.

切削ばねを用いた柔軟な手を用いることで, ハンドルだけでなく, (e) のサイドブレーキや (f) の車のキー回し, (g) のウィンカーを上げる動作等様々な形の車内環境に適応することが可能である. 指の剛性可変機構を用いることで, (g) のように, 柔軟なままでは上げられない車のウィンカーを, 剛性を高くすることで正しく上げることが出来るようになる. 柔軟性を変化させることで, 様々な動作が可能になるのである.

(h) では, 面全体の力を測定することができる足を用いることで, 例えば足がスリップしてブレーキペダルの下に入ってしまったとしても, 足裏の甲にかかる力を感じて元に戻すことができる.

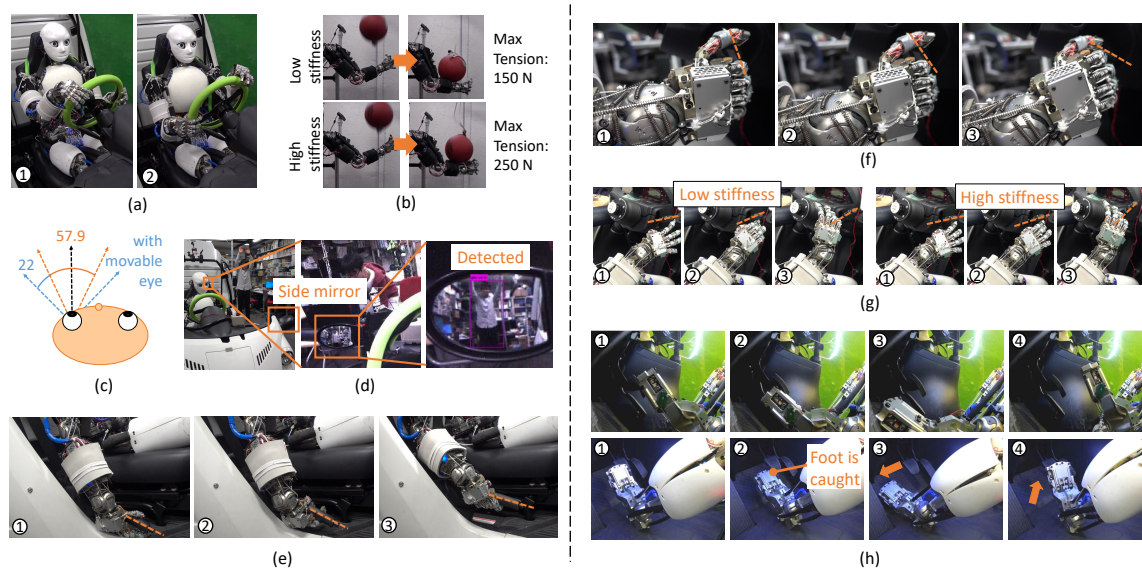


Fig. 8.18: The realization of respective components of autonomous driving using the characteristics of the developed hardware. (a) shows a steering wheel operation with both arms. (b) shows a variable stiffness control using the nonlinear elastic units. (c) shows the field of view of the movable eye unit. (d) shows the human detection experiment in the side mirror using the high resolution camera. (e) shows an experiment pulling a handbrake. (f) shows an experiment rotating a key. (g) shows an experiment operating a blinker lever by changing the stiffness of fingers. (h) shows a recovering experiment from slipping during brake pedal operation using the developed foot.

8.3.3 筋骨格ソフトウェアと自動運転

設計プロセス

これまで説明した柔軟な身体と冗長なセンサを扱うためには、(1) 学習型の動作生成手法、(2) 学習型の認識手法、(3) それらを補う速い周期における反射型制御が必要であると考えた。(1) は静的な動作を扱う動作生成手法・動的な動作を扱う動作生成手法に分けることができる。前者は状態数が少ないため広く学習を行うことができ、後者は状態数が増えるため複雑な学習は難しいが動的な要素を扱うことが可能である。また、タスクを行う上で認識が必要であり、(2) 目や耳のセンサを用いた学習型の認識手法も開発する。しかし、学習型の制御は速い周期で運用することが難しいため、安全機構等により低レイヤに(3) 反射型制御として実装する。これらは基本的なサイクルである認識・計画・動作のうちの認識と動作について扱っており、本研究では計画については認識結果に基づいた簡易な条件分岐を用いることとした。

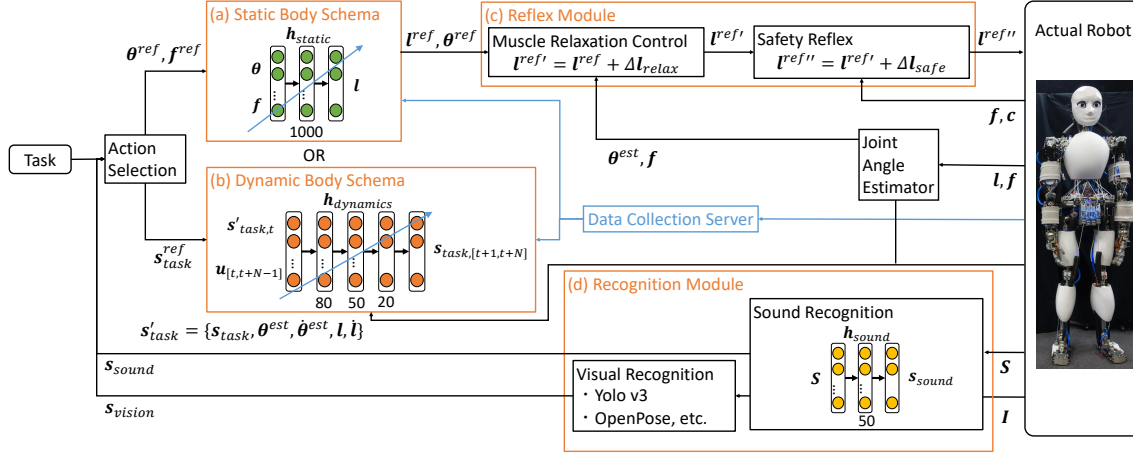


Fig. 8.19: The overview of the developed software with four modules: intersensory network module (static module), dynamic task control network module (dynamic module), reflex module, and recognition module.

ソフトウェアの詳細

Musashi の学習型ソフトウェアシステムの概要を Fig. 8.19 に示す。ここで、 I は現在画像、 S は現在音声、 s_{task} は設定したタスクの状態を表す変数、 s_{vision} は視覚から得られた現在状態、 s_{sound} は音声から得られた現在状態を表す。また、 $\{l, \theta, f, s_{task}\}^{ref}$ はそれぞれの状態の制御指令値、 θ^{est} は関節角度の推定値を表す。

基本的な要素は 4 つであり、それぞれ静的身体図式学習、動的な身体図式学習、反射制御、認識制御である。それらの概要を Fig. 8.19 に示す。特に (a), (b) のモジュールは、筋骨格ヒューマノイドの柔軟な身体構造はモデル化が難しく、身体の動かし方を実機センサデータに基づいて獲得していく必要があることに起因する。(c) は反射型の安全管理制御、(d) は目・耳による認識に関するモジュールになっている。

(a) の静的身体図式学習 [125] は以下の関数 h_{static} を学習していくことに相当する。

$$l = h_{static}(\theta, f) \quad (8.4)$$

これは θ, l, f の静的な関係を表し、6.3 節の特殊な場合である。タスクを与えると、それに応じた θ^{ref} や f^{ref} が決まる場合、それを Eq. 8.4 に代入することで、ロボットに指令すべき l^{ref} を算出することができる。CAD 上で筋の起始点・中継点・終止点を直線で結んだものを筋経路とし、筋張力による非線形弾性要素や Dyneema の伸びをモデル化することでデータセットを作成し、このネットワークを学習させる。しかし、CAD から得たモデルには大きな誤差が存在するため、これを実機センサデー

タからより正しく更新していく必要がある。そこで、実機から (θ, f, l) の情報を取得し、これを用いて h_{static} をオンラインで更新していく。すると、動作するごとにネットワークが更新されていき、より正確に θ^{ref}, f^{ref} を実現できるようになる。本研究では θ を関節モジュールから得ることができるが、通常の筋骨格ヒューマノイドは球関節等の影響から関節角度センサを備えていない。その場合は、モーションキャプチャや視覚センサ情報から θ を得る必要がある。また、このネットワーク h_{static} を用いることで、制御だけでなく、筋長・筋張力変化から拡張カルマンフィルタを用いて関節角度 θ^{est} の推定を行うことができる。

(b) の動的な身体図式学習 [271] は以下の関数 $h_{dynamic}$ を学習していくことに相当する (7.6 節)。

$$s_{task,[t+1,t+N]} = h_{dynamic}(s'_{task,t}, u_{[t,t+N-1]}) \quad (8.5)$$

ここで、 $a_{[A,B]}$ は $[A, B]$ の間におけるベクトル a を縦に並べたベクトル、 $s'_{task,t}$ は初期状態 (本研究では $\{s_{task}, \theta^{est}, \dot{\theta}^{est}, l, \dot{l}\}$ とする)、 u はロボットの制御入力 (本研究では θ^{ref} または l^{ref})、 N は時系列に展開するタイムステップ数を表す。これは、タスク状態の制御入力による動的な遷移を表すネットワークに相当する。このネットワークを実機のデータから学習する。例えばランダムな制御入力を加えたときのタスク状態変化を観察しデータセットを作成することで、このネットワークを構築することができる。あるタスク状態 s_{task}^{ref} を実現したいときは、適当な u の初期値を決定し、以下の式を実行する。

$$s_{task,seq}^{pred} = h_{dynamic}(s'_{task,t}, u_{seq}^{init}) \quad (8.6)$$

$$L = \text{MSE}(s_{task,seq}^{pred}, s_{task,seq}^{ref}) + \alpha E_{adj}(u_{seq}^{init}) \quad (8.7)$$

$$g = dL/du_{seq}^{init} \quad (8.8)$$

$$u_{seq}^{init} \leftarrow u_{seq}^{init} - \beta \frac{g}{|g|} \quad (8.9)$$

ここで、 u_{seq}^{init} は制御入力の初期値 u^{init} を並べた $u_{[t,t+N-1]}^{init}$ 、 $s_{task,seq}^{pred}$ は $s'_{task,t}$ と u_{seq}^{init} から予測された $[t+1, t+N]$ の s_{task} の時系列、 $s_{task,seq}^{ref}$ は N 個 s_{task}^{ref} を並べたものを表す。また、MSE は Mean Squared Error を、 E_{adj} は隣り合う時系列間の値の二乗誤差平均、 α は係数を表す。予測された s_{task} の時系列と指令値の間の誤差を取り、また、隣り合う制御入力が近くなるように、つまり滑らかに制御入力を変えていくように誤差を定義し、これを制御入力に対して誤差逆伝播 [140] していく。これを繰り返すことで、 u_{seq}^{init} はより正確に s_{task}^{ref} を実現していくものへと変化していき、 u_t^{init} を実機に指令することで、タスクを実行する。(a) は静的な動作に関するネットワークであるのに対して、(b) は動的な動作に関するものであり、より変数が多くなるため学習が難しい。そのため、基本的には (a) を用いて動作するが、より動的で正確な動作が要求されるタスクにおいては、(b) をオフラインで構築し実行する。

(c) の反射制御は、筋弛緩制御、安全反射機構の 2 つからなる、速い周期で実行される制御群である。筋弛緩制御は、現在の関節角度を保持したまま拮抗筋から主動筋までを順に緩ませていく制御である (5.3 節)。まず、タスクに必要な関節トルク τ^{nec} から以下の 2 次計画法を解くことで必要な筋張力 \mathbf{x} を導出する。

$$\underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{x}^T W_1 \mathbf{x} + (G^T \mathbf{x} + \tau^{nec})^T W_2 (G^T \mathbf{x} + \tau^{nec}) \quad (8.10)$$

$$\text{subject to} \quad \mathbf{x} \geq \mathbf{f}^{min} \quad (8.11)$$

ここで、 W_1, W_2 は重み行列であり、 \mathbf{f}^{min} は最小の筋張力を表す。これを昇順にソートし、より小さい筋張力、つまり必要のない拮抗筋から順に少しずつ緩ませていく。具体的には、弛緩度合い Δl_{relax} を増やしていき、最終的に実機に指令する筋長にそれを加算する。筋張力が \mathbf{f}^{min} より小さくなれば、緩める筋を次の筋へと移動していき、関節角度がある一定以上変化してしまったらこの制御を止める。本制御はロボットの静止時にのみ働き、逆に動作時には筋張力を降順に並べ、必要のある筋から Δl_{relax} を減らしていく。これにより、モデル誤差による無駄な拮抗筋にかかる筋張力を抑えることができる。また、例えば手を机の上に置くような動作では、身体と環境が拘束されるため、拮抗筋だけでなく、主動筋を緩ませても関節角度は変化しない。ゆえに、拮抗筋だけでなく主動筋の筋張力も削減でき、より身体を休め、長時間の行動が可能となっていく。安全反射機構は単純で、高い筋張力・筋温度によりモータが破損しないようにするための制御である。以下のように計算された Δl_{safe} を指令筋長へと加算する。

$$\Delta l_{safe}^{ref} = K_f \max(\mathbf{f} - \mathbf{f}^{lim}, \mathbf{0}) + K_c \max(\mathbf{c} - \mathbf{c}^{lim}, \mathbf{0}) \quad (8.12)$$

$$\Delta l_{safe} \leftarrow \Delta l_{safe} + \max(\Delta l^{min}, \min(\Delta l^{max}, \Delta l_{safe}^{ref} - \Delta l_{safe})) \quad (8.13)$$

ここで、 $\{\mathbf{f}, \mathbf{c}\}^{lim}$ は安全反射機構を駆動させ始める筋張力・筋温度の閾値、 $K_{\{f, c\}}$ は係数、 Δl_{safe}^{ref} は伸張させるべき理想の値、 $\Delta l^{\{min, max\}}$ は 1 タイムステップにおける Δl_{safe} の変化の最小値と最大値を表す。このように筋長変化を制限しながら筋温度と筋張力を見て弛緩させることで、振動を防ぎつつ、モータの破損を抑制することができる。

(d) の認識制御は物体の認識と、音声の認識に分けられる。本研究では、物体認識は Yolo v3 [292] を用いている。主に用いるラベルは、“Car”、“Human”、“Traffic Light”の 3 つである。また、音声認識は音声をメルスペクトラムに変換し、それを入力として音声のクラスを出力するようなネットワーク \mathbf{h}_{sound} を学習させている [158]。本研究では主に“Noise”と“Car Horn”のみを識別している。

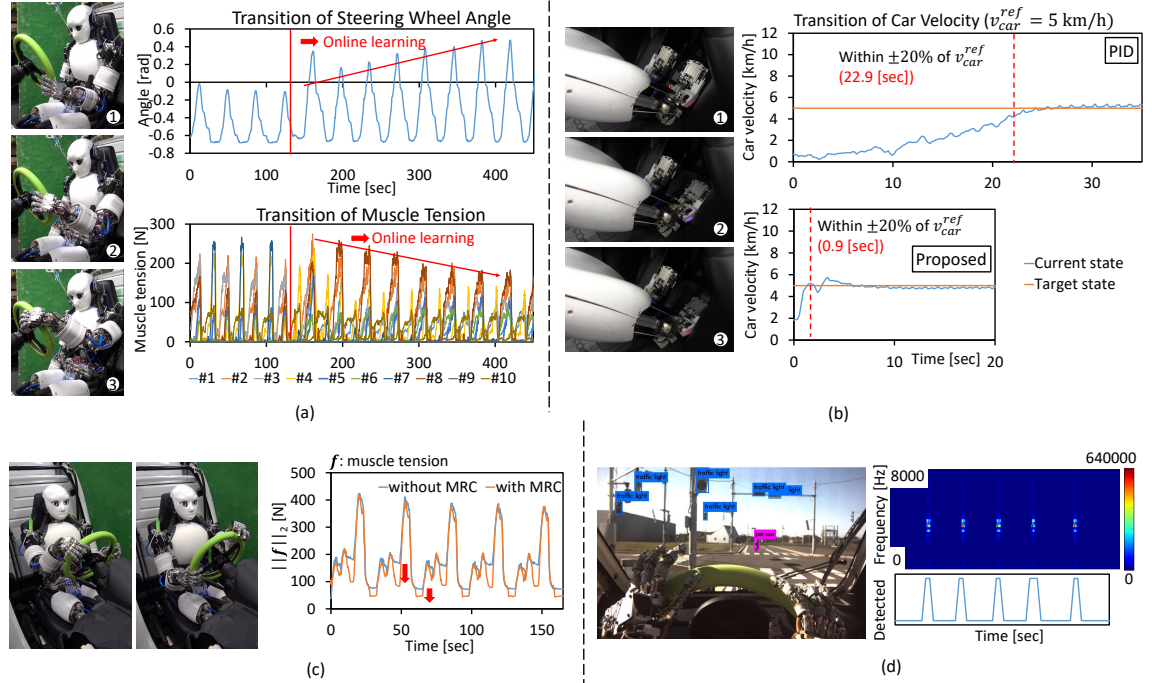


Fig. 8.20: Respective components of autonomous driving using the developed software. (a) shows a steering wheel operation experiment using the online learning of static body schema. (b) shows a pedal operation experiment using the trained dynamic body schema. (c) shows a steering wheel operation experiment with and without muscle relaxation control. (d) shows visual recognition of traffic lights and a human, and sound recognition of a car horn.

自動運転への応用

それぞれの制御コンポーネントの自動運転に着目した利用について Fig. 8.20 に示す。

静的身体図式学習は様々な動作において適用可能であるが、特に、環境との接触を要するハンドル操作において有用である。(a) ではオンライン学習を実行させると同時にハンドルを回し続ける実験を行った。その際のハンドル角度の遷移、Musashi の肩と肘に備わる 10 本の筋の筋張力の遷移を表している。動作を継続するごとに徐々に回転するハンドルの角度が大きくなり、また、筋張力が徐々に下がっていることがわかる。よって、動作する中で実機における θ, f, l の関係が正しく学習されていくことがわかる。

動的身体図式学習は動的な動作に有用であるが、その中でもペダル操作は適応的な速度調整が必要であり応用が期待される。(b) では 1 分間程度ランダムにペダルを踏み走行したときのデータを用いて動的身体図式学習を学習させ、それを用いてペダル操作を行う (7.6 節と同様の実験である)。また、

本実験は車の後輪をフリーローラの上に起き、室内で実験している。ここで、 v_{car} は車速、 θ_{ankle} は足首の関節角度を表し、 $s_{task} = v_{car}$ 、 $u = \theta_{ankle}$ とする。 $v_{car}^{ref} = 5[km/h]$ と設定したうえで、通常の PID 制御 (PID)、動的体図式学習 (Proposed) を用いた制御を実行した。関節角度 θ_{ankle}^{ref} は最終的に静的体図式学習によって筋長 l^{ref} に変換されて実機に送られている。手動でチューニングされた PID と Proposed を比べると、それ以降 v_{car} と v_{car}^{ref} の誤差が 20% 以下となる時間は 22.9 sec と 0.9 sec となった。つまり、タスク状態と制御入力に関する状態方程式を獲得して最適制御を実行することで、より速い制御指令値への追従が可能となる。

反射制御は静止するような動作が多い場合は非常に有効である。例えば (c) のような両手での運転動作では常にハンドルを回転させているとは限らず、真っ直ぐの道ではハンドルをほとんど回さないことが多い。その場合、筋弛緩制御 (Muscle Relaxation Control, MRC) によって拮抗筋が徐々に緩み、内力が減っていく。また、ハンドルという環境に手をかけているため主動筋を弛緩させても現在関節はほとんど変わらない。グラフには MRC を入れた場合と入れない場合における左腕の肩と肘の 10 本の筋の筋張力の L2 ノルム $\|f\|_2$ の遷移を示す。MRC を入れた場合は入れない場合に比べて静止時で筋張力が削減されていることがわかる。これにより、筋温度の上昇が抑えられ、より長時間の動作が可能となる。

認識制御は本研究では主に交差点における人・信号の認識、“Car Horn” の認識に利用する。(d) の左図は交差点における認識結果であるが、しっかりと信号や人等を認識できていることがわかる。また、(d) の右図では車のクラクションの際の音声スペクトラムとその際の認識結果を示している。正しく“Car Horn” を認識できていることがわかる。

8.3.4 実験

実験セットアップ

本研究のペダル操作実験で用いる自動車は、トヨタ車体製の超小型 EV コムス (COMS: Chotto Odekake Machimade Suisui) シリーズの B・COM デリバリーである (Fig. 8.21)。安全のため、モータトルクはソフトウェア上で 5 Nm に制限されており、非常停止ボタンが備えられている。ペダル操作は (a) に示すように右足左足でそれぞれアクセルとブレーキを踏むようになっている。実験は (b) の東京大学柏キャンパスの実験施設で行った。(c) に示すように、COMS の内部にはバッテリー、ロジックの電源装置、サーボの電源装置、Wi-Fi ルータ、認識制御用の PC が積まれている。COMS は電気自動車であり、将来的にはロボットの電源は全て車体から得られるようになる可能性がある。(d) に示すように、

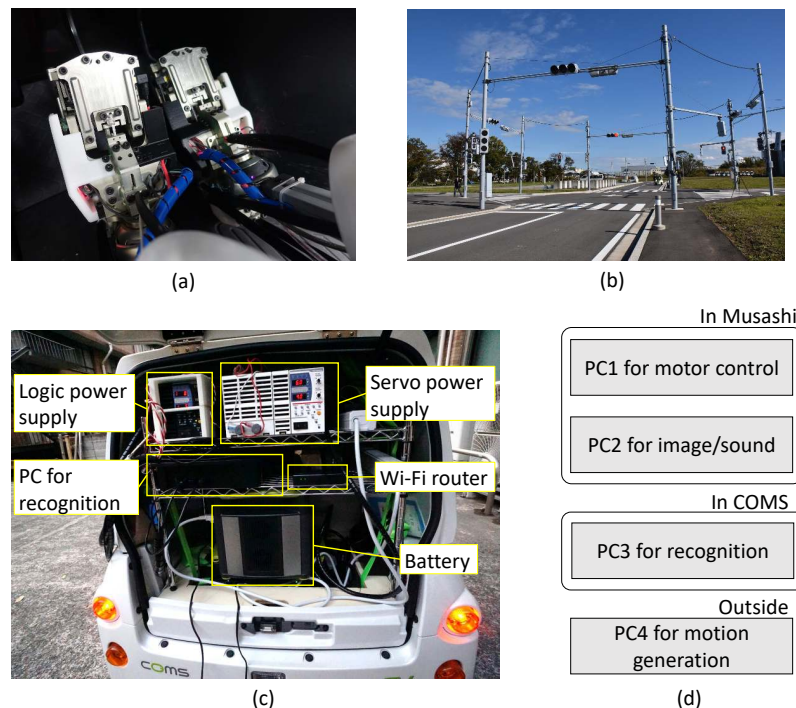


Fig. 8.21: Experimental setup of autonomous driving by the musculoskeletal humanoid Musashi: (a) pedal operation configuration, (b) experimental environment, (c) experimental configuration in COMS, and (d) configuration of PCs.

Musashi の頭部にはモータ制御用、画像・音声取得用の PC が備えられており、認識制御は COMS 搭載の PC 上で行い、それ以外の動作生成関係は外部の PC4 で行っている。

認識を含むペダル操作実験

ペダル操作、認識を統合した実験を行った (Fig. 8.22). 動作の遷移図を Fig. 8.22 の (a) に示す。動的な身体図式学習を用いたペダル操作で走行を行い、人を認識したらブレーキをかけ、また走りだし、クラクションを聞いたらブレーキをかけ、また走りだす。それら一連の動作が、Fig. 8.22 の (b) に示され、成功していることがわかる。(c) はそれぞれ車速 (上図) とアクセルペダルを踏み込む右足首ピッチの角度 (下図) を示している。(d) は人認識の様子とそのときの認識結果を示している。人認識は、人の Bounding Box がある閾値よりも大きく、かつ中心にあるときだけ人として認識している。また、(e) はクラクションのスペクトルとその認識結果を示しており、認識が成功していることがわかる。(d)、(e) におけるそれぞれの検知時に、ブレーキが踏まれ、車速が 0 になっていることがわかる。ここで大きな問題は、(c) に示したように、指令車速に対して、現在車速の追従が悪い点である。これは、動的な身体図

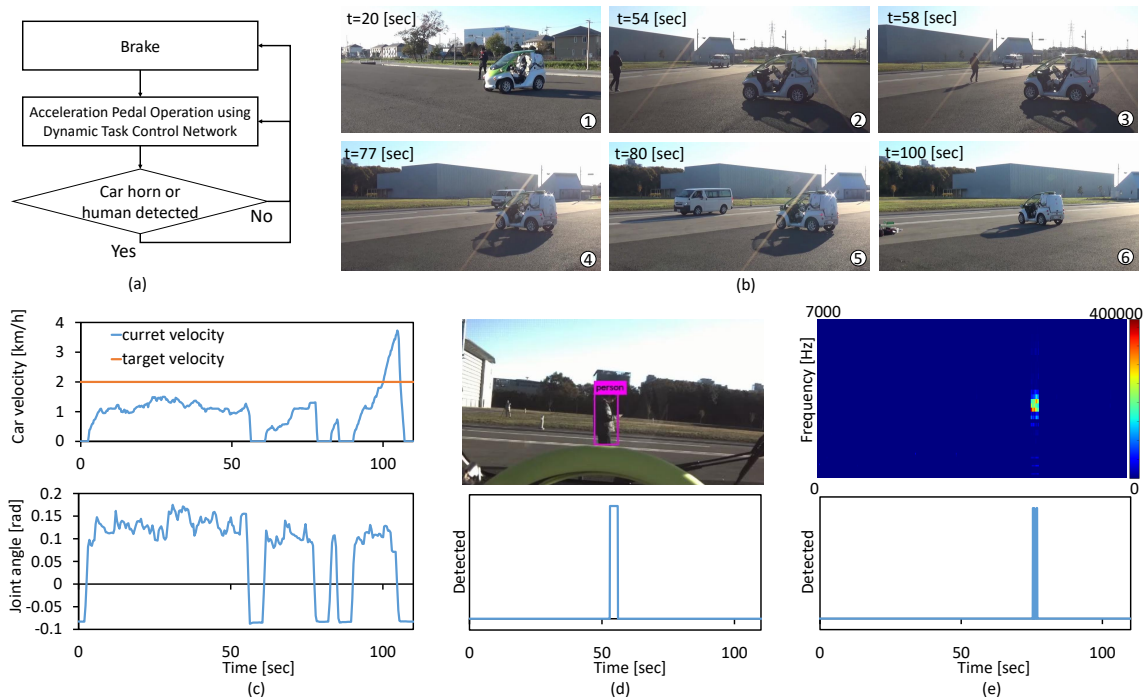


Fig. 8.22: Pedal operation experiment with recognition. (a) shows the experimental motion flow. (b) shows the experimental appearance. (c) shows the current and target car velocity (upper graph) and the joint angle of the right ankle pitch for the acceleration pedal (lower graph). (d) shows the visual recognition result. (e) shows the sound spectrum and its recognition result.

式学習を学習させた環境と、実際に走行を行った環境の違いによるものである。 $t = [0, 60]$ [sec] ではより摩擦が大きく進みにくい路面なため速度が落ち、 $t = 100$ [sec] では下り坂になっており加速が大きくなってしまったと考えられる。実験室の中でフリーローラの上に車体を載せているときには問題にならなかった、環境の違いが大きく問題となっている。学習型の制御は動作が学習したデータに依存するため、実験室の中だけでなく、より屋外での実験を重ね、データの取得とこれらの問題を解決できる手法が今後望まれる。

認識を含むハンドル操作実験

ハンドル操作、認識を統合した実験を行った (Fig. 8.23)。動作の遷移図を Fig. 8.23 の (a) に示す。赤信号のときは止まっており、青信号になったら発進、静的身体図式学習を用いたハンドル操作によって交差点を曲がる。本実験では車速が速いと交差点を曲がり切れないため、車はクリープによって動作し、信号を認識したらブレーキを無くすことのみを行う。また、ハンドルを最初は右に切り、途中で

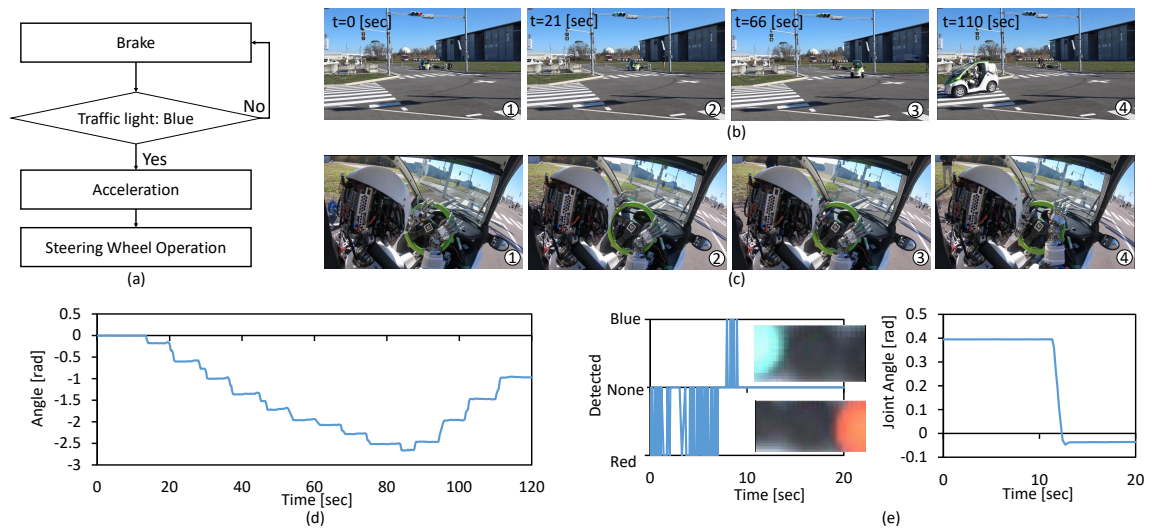


Fig. 8.23: Steering wheel operation with recognition. (a) shows the experimental motion flow. (b) shows the experimental appearance. (c) shows the sequence of steering wheel operation. (d) shows the transition of the steering wheel angle. (e) shows the traffic light recognition result and the joint angle of the left ankle pitch for brake pedal operation.

オペレータの指令によって左に切り始めるようになっている。(b) は実験の様子を表しているが、青信号になったら走り始め、90 度の交差点を曲がりきることができていることがわかる。(c) はハンドル操作の様子を表している。ハンドル操作のシーケンスは、両手で回せるところまで回し、左手を離して元の位置まで戻し、同様に右手も離して元の位置まで戻し、また両手で回すという一連の動作を繰り返す。その際のハンドルの角度遷移は (d) のようになっている。約 70 秒かけて 180 度近くまでハンドルを回転させることができています。(e) の左図は信号認識の様子を表しており、物体認識によって信号を切り抜き、その中の青の画素と赤の画素の比率で信号の色を認識している。赤または青の画素がない場合は None となっており、信号が赤から青に切り替わる瞬間を読み取れていることがわかる。それと同時に、(e) の右図に示すようにブレーキを踏む左足の足首ピッチが戻り、車が発進している。本実験における問題点は、ハンドル操作の遅さである。この交差点を曲がるのに約 2 分間を要しており、今後クロスハンドルをスムーズに行える方法が必要となると考える。

8.4 低剛性樹脂製ヒューマノイドによる道具操作実験

8.4.1 概要と先行研究

これまで様々なロボットが開発されてきたが、産業用ロボットに多く見られる高剛性で力の強いロボットが多数存在する一方 [77, 8], ゴムや樹脂によりリンクや関節が構成された低剛性なロボットも存在する [293, 97]. これらは *Soft Robotics* のように柔軟性を追求して開発される場合もあれば、値段や軽量化の観点から樹脂によってロボットを構成する場合もあり、その理由は様々である. 高剛性なロボットはモデル化によって容易にかつ正確に動作する一方、低剛性なロボットはその低剛性ゆえに関節やリンクが姿勢や力に応じてたわみ、そのモデリングが難しいといった問題がある. 特に、関節のみ低剛性な場合は、リンクに関する幾何モデルは正しいとして一般的な運動学や動力学が適用可能である一方、リンクまで低剛性な場合はよりモデル化が困難を極め、それらの適用が難しくなる. もちろん、これら関節やリンクのたわみを考慮したような手法は開発されている一方 [294], 多数の仮定が必要な点や、多リンクにした際の計算量等の観点からまだ問題は多く残る. 特に、低剛性なロボットによる道具利用においては、物体の把持によって身体のたわみ方が変化し、ある姿勢に対する道具の先端位置や重心位置等が大きく変化してしまう. また、そのたわみ方は道具の重量や長さ等によっても変化するため、これらのモデリングは困難を極める (Fig. 8.24).

そこで本研究では、低剛性なロボットによる道具利用に向けた、関節角度や視覚情報、触覚情報の相互関係を記述するニューラルネットワークを用いた、たわみを考慮した道具先端位置の制御手法を開発する. また、このたわみは把持した道具の長さや重さによっても変化するため、その変化を現在のセンサ情報から推定し、より正確に制御を行う. 学習型での道具利用には道具選択 [205] や動作計画 [214], 道具理解 [212] 等について様々な学習型の手法が研究されているが、そのどれもが剛な身体での実験であり、また、重心位置等を考慮したものはほとんどない. 模倣学習の文脈では柔軟関節を考慮した手法はあるものの [61], 柔軟リンクは考慮しておらず、また、模倣学習であるため人間のデモンストレーションが必要な点で、ロボット自身が身体のセンサ間の相互関係を学習する本研究の目的とは異なる. この他にも強化学習を利用した手法 [217] や自己教師あり学習を使った手法 [218] があるものの、関節やリンクの柔軟性、それに伴う身体重心変化等を考慮した研究はない. また、先行研究として逐次的な道具把持状態の変化を考慮した手法が開発されている [220] が、本研究ではこれに接触覚や異なる視覚情報を加え、物体の重さやそれに伴う身体のたわみを考慮している.

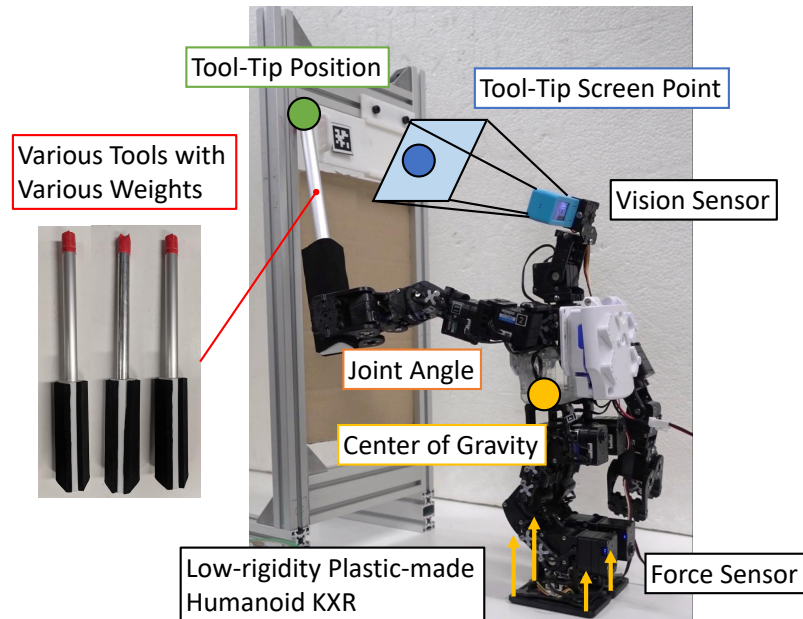


Fig. 8.24: Concept: learning the relationship among joint angle, center of gravity, tool-tip screen point, and tool-tip position for adaptive robotic tool-use of robots with low rigidity.

8.4.2 低剛性樹脂製ヒューマノイドによる全身道具操作

本研究で提案する Whole-body Tool-use Network with Parametric Bias (WTNPB) を含む全体システムを Fig. 8.25 に示す.

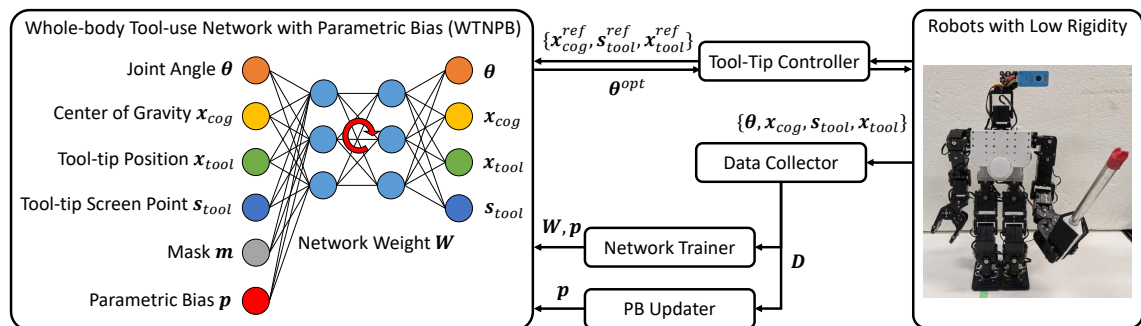


Fig. 8.25: System overview using WTNPB.

ネットワーク構造

WTNPB のネットワーク構造を式で表現すると以下になる．

$$(\boldsymbol{\theta}, \mathbf{x}_{cog}, \mathbf{x}_{tool}, \mathbf{s}_{tool}) = \mathbf{h}(\boldsymbol{\theta}, \mathbf{x}_{cog}, \mathbf{x}_{tool}, \mathbf{s}_{tool}, \mathbf{m}, \mathbf{p}) \quad (8.14)$$

ここで、 $\boldsymbol{\theta}$ はロボットの関節角度における指令値、 \mathbf{x}_{cog} は重心位置、 \mathbf{x}_{tool} は 3 次元空間上での道具の先端位置、 \mathbf{s}_{tool} はカメラ画像中での道具の先端位置、 \mathbf{m} はマスク、 \mathbf{p} は Parametric Bias を表す．つまり、これは $\mathbf{x}^T = \begin{pmatrix} \boldsymbol{\theta}^T & \mathbf{x}_{cog}^T & \mathbf{x}_{tool}^T & \mathbf{s}_{tool}^T \end{pmatrix}$ とした場合の一般化多感覚相関モデルになる． $\boldsymbol{\theta} = \begin{pmatrix} \theta_{s-p} & \theta_{s-y} & \theta_{e-p} & \theta_{a-p} \end{pmatrix}^T$ は肩の pitch と yaw の角度 (s-p, s-y)、肘の pitch の角度 (e-p)、そして足首の pitch の角度 (a-p) の 4 つの角度の指令値とする． $\mathbf{x}_{cog} = \begin{pmatrix} x_{cog} & y_{cog} \end{pmatrix}^T$ はそれぞれの足裏の四隅に配置された 1 軸の力センサから算出する (道具操作中は足平の位置は変更しないため、足は揃っているという仮定を置いている、また、z 方向は無視する)． \mathbf{x}_{tool} は道具先端位置を AR マーカ等により認識した際の、その 3 次元空間上での位置を表す． \mathbf{s}_{tool} は道具先端位置を色認識等によって認識した際の、その画像上の 2 次元の位置を表す．実行可能な \mathbf{m} の集合 \mathcal{M} はネットワークの自動入出力決定機構により自動的に決定される． \mathbf{p} は道具の長さや重さの違いによるダイナミクスの変化を表現する Parametric Bias である．

本研究では、ネットワーク構造は 7 層とし、ユニット数については、入力 \mathbf{x} の 11 次元、 \mathbf{m} の 4 次元、 \mathbf{p} の 2 次元を足し合わせた 17 次元、中間層は $\{200, 50, 8, 50, 200\}$ 、出力は \mathbf{x} の 11 次元とした．活性化関数は Tanh、更新則は Adam [46] とする．ネットワークの入力と出力は訓練時に得られたデータを使って正規化されている．また、この中間層のユニット数 8 の部分が潜在変数 \mathbf{z} に相当する．

WTNPB の訓練

まず、道具の長さや重さが異なるような、様々な道具状態 k ($1 \leq k \leq K$, K は訓練に使う全道具状態数) について、様々な姿勢におけるデータ $D_k = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_k}\}$ を収集する (N_k は道具状態 k に関するデータ数)．また、それぞれの道具状態 k について Parametric Bias \mathbf{p}_k を用意する (全ての \mathbf{p}_k は 0 に初期化されている)．データ $D_{train} = \{(D_1, \mathbf{p}_1), \dots, (D_{N_k}, \mathbf{p}_{N_k})\}$ が収集され、この D_{train} を用いて \mathbf{h} を学習させる．このとき、 \mathbf{p}_k はデータ D_k については共通であり、異なる道具状態については異なる変数である．学習の際はネットワークの重み W と同時に \mathbf{p}_k も誤差逆伝播法によって更新する．これにより、 \mathbf{p}_k に道具状態に関する情報が埋め込まれる．

本研究では学習は 2 段階で行う．まず、シミュレーション上で道具状態を変化させてデータを収集し、 W と \mathbf{p}_k を計算する．ここでは、 $\boldsymbol{\theta}$ を設定した関節角度範囲内でランダムに動かすことでデータを

収集する. s_{tool} や x_{cog} については, カメラやリンク重量等のモデルから計算する. 関節トルクに応じた関節誤差を簡易的にモデル化し, 3.0τ (τ の単位は [Nm]) だけ関節がたわむと仮定してデータを取得している. その後, シミュレーションで計算された W のみ残し, p_k を 0 に初期化する. 最後に, 実機においてデータを収集し, 学習を行う. なお, 道具先端に AR マーカをつけることで, x_{tool} と s_{tool} を含めた x の全ての値をデータとして収集する. ここでは, 二種類の方法でデータを収集する. 一つは GUI から人間が θ を直接指定し, そのときのデータを収集していく方法である. もう一つはシミュレーションにおいて θ をランダムに指定し, このとき重心が支持領域内に含まれ, かつ道具先端位置がカメラから見える場合の θ を実機に送って動かす. 重心位置に関する制約を強くすることで, たわみによりシミュレーションと実機が異なっても, 倒れない範囲でデータを収集することが可能である. 実機で得られるデータは少数なため, シミュレーションからの Fine Tuning により対応する.

WTNPB のオンライン学習

新しく道具を持った際に, 自身の身体を動かすことで, この道具の長さや重さに関する情報を更新する必要がある. そのため, Parametric Bias p のオンライン学習を行う. x のいずれかの値について, その値が取得できるかつ, 直前に収集された値よりもある程度離れている場合 ($\|x_i - x_i^{prev}\|_2 > C_i^{online}$, x_i^{prev} は直前に収集された x_i , C_i^{online} は閾値) に, そのタイミングで取得できている x の全ての値をデータとして収集する. 得られたデータ数 N^{online} が, N_{thre}^{online} を超えてから学習を開始し, その後新しいデータ収集される度に学習を行う. 重み W を固定し, p のみを, バッチ数を N_{batch}^{online} , エポック数を N_{epoch}^{online} として更新する. この際の更新則は, 学習率を 0.01 とした Momentum SGD とする. データは N_{max}^{online} ($N_{thre}^{online} \leq N_{max}^{online}$) を最大値とし, それを超えたデータは古いものから順に削除していく ($N^{online} \leq N_{max}^{online}$). ネットワークの重み W を固定し, 小さな次元である Parametric Bias のみ更新することで, 過学習を防ぎつつ道具状態のみを更新することができる. なお, このとき x のうち得られたデータのみを使うマスク m が実行可能なマスク集合 M に含まれている必要がある. この m を使って z を計算し, 出力した x に対して, 得られたデータについてのみ損失を計算している. x のうち得られるデータが多ければ多いほどオンライン学習はうまく動作する. 本研究では訓練データ収集時のみ道具先端位置を AR マーカを使って取得するため, このオンライン学習時には x_{tool} は得られない. よって, $\{\theta, x_{cog}, s_{tool}\}$ から p を学習する.

本研究では, $C_{collect} = \{10 [\text{deg}], 3 [\text{mm}], 20 [\text{mm}], 100 [\text{px}]\}$, $N_{thre}^{online} = 5$, $N_{batch}^{online} = N^{online}$, $N_{epoch}^{online} = 5$, $N_{max}^{online} = 100$ とした. また, データ収集の実行周期は 5 Hz である.

WTNPB による制御

道具先端位置の制御は誤差逆伝播と勾配降下法を使った最適化により行う。以下のように最適化を行う。

$$L = \|\mathbf{x}_{tool}^{pred} - \mathbf{x}_{tool}^{ref}\|_2 + \alpha \|\mathbf{x}_{cog}^{pred} - \mathbf{x}_{cog}^{ref}\|_2 \quad (8.15)$$

$$\mathbf{z}^{opt} \leftarrow \mathbf{z}^{opt} + \gamma \partial L / \partial \mathbf{z}^{opt} \quad (8.16)$$

ここで、 \mathbf{x}_{tool}^{ref} は道具先端位置の指令値、 \mathbf{x}_{cog}^{ref} は指令重心位置 (本研究では $(0 \ 0)^T$)、 \mathbf{z}^{opt} は最適化する \mathbf{z} の値、 $\mathbf{x}_{\{tool, cog\}}^{pred}$ は \mathbf{z}^{opt} から予測された $\mathbf{x}_{\{tool, cog\}}$ 、 α は損失関数の重み (本研究では 0.01)、 γ は学習率を表す。 \mathbf{z}^{opt} の初期値は $\mathbf{m} = (0 \ 1 \ 0 \ 1)^T$ として $\mathbf{x}_{\{tool, cog\}}^{ref}$ と \mathbf{h}_{enc} から予測された値を用いる。 \mathbf{z}^{opt} を損失 L から更新していき、最終的に計算された \mathbf{z}^{opt} から得られた $\boldsymbol{\theta}$ を実機に指令することで、重心位置を考慮しながら道具の先端位置を制御することができる。なお、もし指令関節角度を現在の値からなるべく動かしたくなければ $\|\boldsymbol{\theta}^{pred} - \boldsymbol{\theta}^{cur}\|_2$ 、カメラ画像中での道具の指令先端位置を得られる場合は $\|\mathbf{s}_{tool}^{pred} - \mathbf{s}_{tool}^{ref}\|_2$ を L に加えても良い。

8.4.3 実験

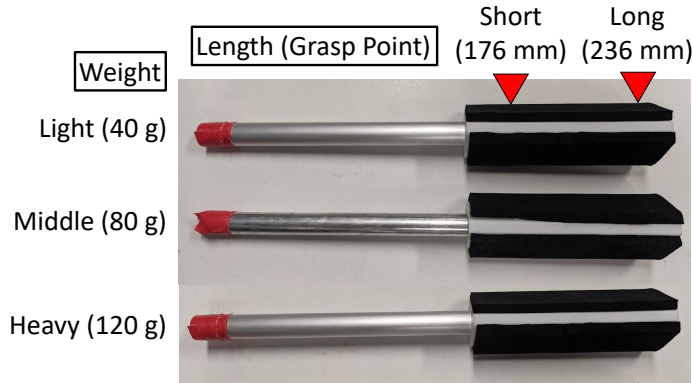


Fig. 8.26: Six types of tool states with various weights and lengths used in this experiment.

実験セットアップ

本研究では Fig. 8.24 に示す低剛性な樹脂製ヒューマノイド KXR [97] を実験に用いる。Fig. 8.26 に示すように、3 種類の重さ (Light: 40 g, Middle: 80 g, Heavy: 120 g) の道具を用意する。また、それらの道具の持つ位置に応じて、2 種類の長さ (Short: 全長 176 mm, Long: 全長 236 mm) を定義し、これらの

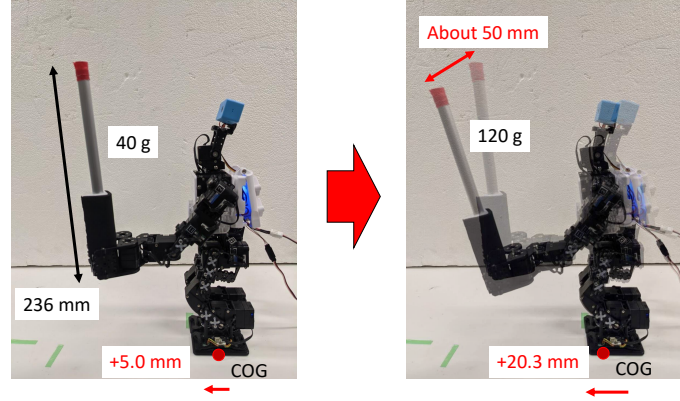


Fig. 8.27: The change in tool-tip position and center of gravity when handling tools with different weights.

組み合わせである計 6 種類の道具状態を扱う. Fig. 8.27 に, 異なる重さの道具を持った際の道具先端位置と重心位置の変化を示す. 同じ長さの道具でも, ハードウェアが低剛性であるがゆえに, 40 g の道具と 120 g の道具を持った時では道具先端位置は約 50 mm, 重心位置は約 15 mm も異なる. 同様に, 道具の長さが変わればその道具先端位置 x_{tool} や s_{tool} が変化すると同時に, 重心位置 x_{cog} も変化する.

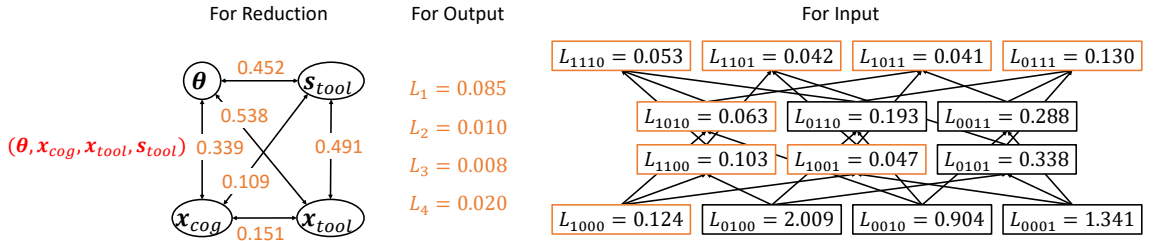


Fig. 8.28: The automatic network decision of WTNPB.

シミュレーション実験

シミュレーションにおいて関節角度をランダムに動かし, 一つの道具状態に対して 500, 全体で 3000 のデータを取得した. このときのネットワーク構造の自動決定について Fig. 8.28 に示す. まず, それぞれのセンサ値について相互情報量を計算し, \mathbf{x} に不要な値がないことを確認した. 次に, L_i を計算し, そのどれもが小さいため, \mathbf{x} における全ての値を \mathbf{x}_{out} として用いることに決定した. 最後に, それぞれのマスク \mathbf{m} について L_m を計算し, $C_{thre}^{in} = 0.15$ と設定することで, 実行可能なマスク集合 $\mathcal{M} = \{(1 \ 1 \ 1 \ 0)^T, (1 \ 1 \ 0 \ 1)^T, (1 \ 0 \ 1 \ 1)^T, (0 \ 1 \ 1 \ 1)^T, (1 \ 0 \ 1 \ 0)^T, (1 \ 1 \ 0 \ 0)^T, (1 \ 0 \ 0 \ 1)^T, (1 \ 0 \ 0 \ 0)^T\}$ が決定され, これに必要な \mathbf{x}_{in} として, \mathbf{x} の全ての

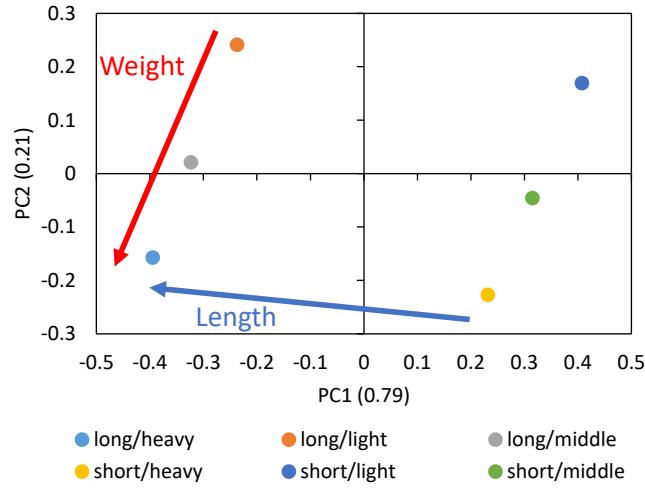


Fig. 8.29: The trained parametric bias in simulation experiment.

値が設定された。この際に得られた Parametric Bias の配置を Fig. 8.29 に示す。それぞれの Parametric Bias が、道具の重さと長さの軸に沿って自己組織化していることがわかる。

次に、この Parametric Bias をオンライン学習することで、現在の道具状態を正確に認識できることを示す。道具を Long/Light の状態、Short/Heavy の状態に設定し、ランダムな関節角度を送りながら Parametric Bias のオンライン学習を実行した。その際の Parametric Bias の遷移を Fig. 8.30 に示す (long/light-traj と short/heavy-traj)。現在の PB 値が、訓練時の Long/Light、または Short/Heavy の PB 値へと徐々に近づいていることがわかる。また、動作時に得られるセンサの種類の違いにより、どうオンライン学習が変化するかを考える。道具先端に AR マーカがついていれば $\{x_{tool}, s_{tool}\}$ が得られるが、AR マーカなしで色認識のみの場合は s_{tool} しか得られない。また、視覚に道具先端が入らない場合は、 $\{\theta, x_{cog}\}$ の値しか得られない。そこで、本研究では得られるデータが A: $\{\theta, x_{cog}\}$, B: $\{\theta, x_{cog}, s_{tool}\}$, C: $\{\theta, x_{cog}, x_{tool}, s_{tool}\}$ の 3 種類の場合で、どのように PB 値が遷移するかを検証する。ここでは、Short/Middle における PB からスタートし、Long/Middle の道具を扱った場合について検証する。その結果を同様に Fig. 8.30 に示す。C (short2long-C) は前述の long/light-traj や short/heavy-traj とセンサの種類が同じであり、素早く PB が遷移し、正確に道具状態を把握できている。B (short2long-B) は C よりも遷移のスピードが落ち、正確に道具状態を把握するのに時間を要している。A (short2long-A) は正しい方向に PB 値自体は進んでいるものの、正しく PB を認識するところまでには至っていない。

最後に、Parametric Bias のオンライン学習を含んだ制御実験を行う。道具状態を Long/Light として認識している状態から始め、実際の道具状態を Long/Heavy, Short/Heavy の順で変更する。なお、ここでは道具変化にすぐ対応できるように、 $N_{max}^{nonline} = 5$ とした。指定した範囲内のランダムな x_{tool}^{ref} を与え、こ

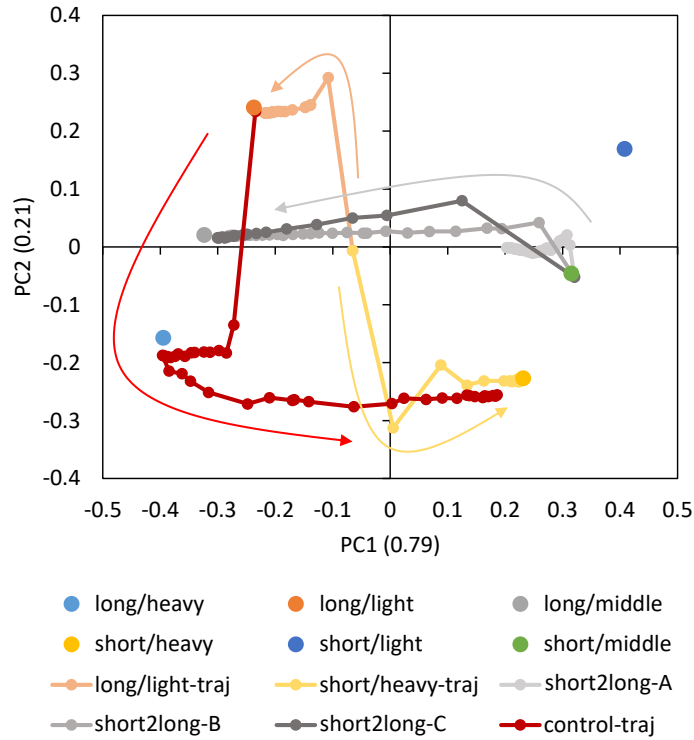


Fig. 8.30: The trajectory of parametric bias during online learning in the simulation experiment.

れに追従するよう、前述の WTNPB を使った動作制御を実行する。このときの Parametric Bias の遷移を Fig. 8.30 の control-traj に示す。Long/Light から Long/Heavy, Short/Heavy の順に Parametric Bias が遷移していることがわかる。また、このときの制御誤差 $\|\mathbf{x}_{tool}^{ref} - \mathbf{x}_{tool}\|_2$ と重心位置誤差 $\|\mathbf{x}_{cog}^{ref} - \mathbf{x}_{cog}\|_2$ 、それぞれの 5 ステップ分の平均を Fig. 8.31 に示す。まず、Long/Heavy の状態では、PB 値が更新されることで、制御誤差が少し、また、重心位置誤差が大幅に減少していることがわかる。PB 値が Long/Light から Long/Heavy に変化し、道具の長さは変わらないが重さが大きく変化したため、制御誤差に比べて重心位置誤差がより大幅に減少したのだと考えられる。その後、Short/Heavy の状態に移行した際には、大きく制御誤差が変化するが、重心位置誤差に大きな変化はない。同様に PB 値のオンライン学習により、制御誤差が大幅に減少していく。一方、重心位置誤差については大きく変化していない。

実機実験

実機において GUI を用いたデータ取得 (60 データ) と関節角度のランダム指定を用いたデータ取得 (20 データ) により、全部で約 480 のデータを取得した。この際に得られた Parametric Bias の配置を Fig. 8.32 に示す。それぞれの Parametric Bias が、道具の重さと長さの軸に沿って自己組織化している

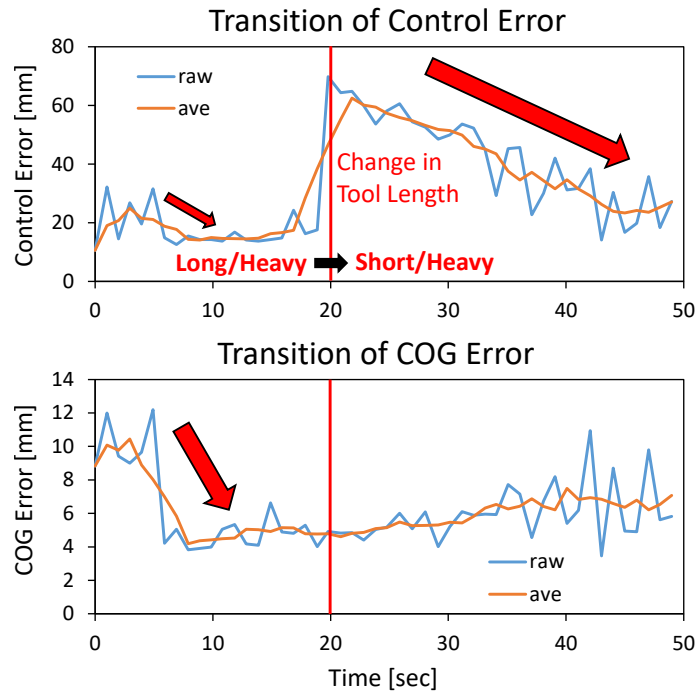


Fig. 8.31: The transition of errors of control and center of gravity in the simulation experiment.

ことがわかる。

次に、この Parametric Bias をオンライン学習することで、現在の道具状態を正確に認識できることを示す。道具を Long/Light の状態、Short/Heavy の状態に設定し、ランダムな関節角度を送りながら Parametric Bias のオンライン学習を実行した。その際の Parametric Bias の遷移を同様に Fig. 8.32 に示す (long/light-traj と short/heavy-traj)。現在の PB 値が、訓練時の Long/Light、または Short/Heavy の PB 値へと徐々に近づいていることがわかる。

最後に、制御誤差に関する評価を行う。Long/Middle の道具状態において、幾何モデルを用いて全身逆運動学を解いた場合、関節のたわみを含むシミュレーションデータから WTNPB を学習させた場合、これを実機において Fine Tuning した場合について制御誤差を比較した結果を Fig. 8.33 に示す。なお、WTNPB を用いる際の Parametric Bias は訓練時の Long/Middle の値である。幾何モデルが最も誤差が大きく、関節のたわみを含むシミュレーションによる学習後、実機による学習後の順で制御誤差が小さくなっていることがわかる。

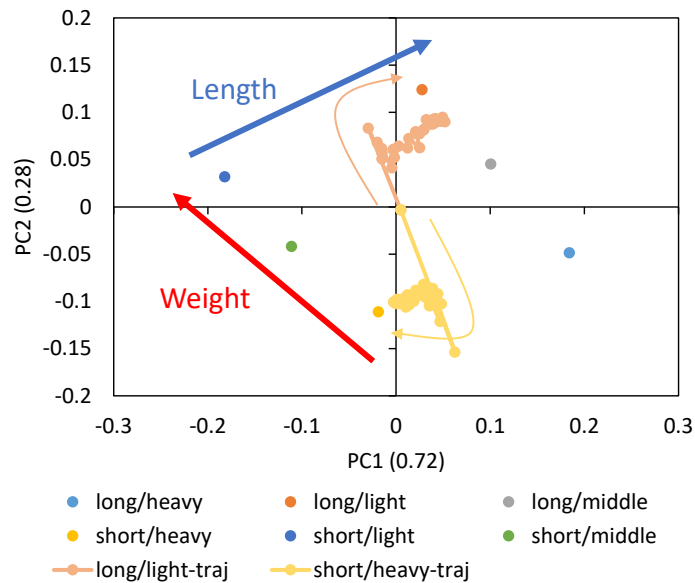


Fig. 8.32: The trained parametric bias and the trajectory of the parametric bias during online learning in the actual robot experiment.

統合実験

実機においてオンライン学習と制御を合わせた最終実験を行う。これは道具を手に取り、高い位置にある窓をその道具を使って開けるような動作である。この際の全体システムを Fig. 8.34 に示す。古典的な AR マーカ、または色認識の視覚制御が働き、これをもとに動作計画が行われる。道具の把持や歩行の部分については古典的な制御が働き、道具操作の部分については WTNPB が働く。常に実機からのデータを取得し、これをもとに WTNPB における PB のみを更新していく。

実験の様子を Fig. 8.35 に示す。Short/Heavy の PB 値の状態から始め、Long/Light の道具を把持する。ランダム動作から現在の PB 値をオンライン学習し、その後横に歩いて窓の前に立つ。窓についた AR マーカから窓の 3 次元位置を認識し、そこから y 方向に -60 mm の地点に道具先端位置を伸ばした後、y 方向に 80 mm 動作させ、窓を開ける。一連の動作は成功し、実機データによる WTNPB の学習、また道具状態を表す Parametric Bias のオンライン学習、道具位置と重心位置を考慮した制御により、低剛性なロボットにおける道具操作タスクが可能であることが示された。

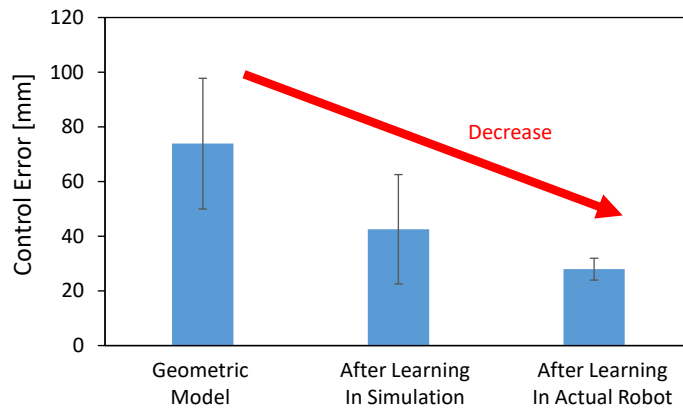


Fig. 8.33: Comparison of control errors among the case using a geometric model, using WTNPB after learning in simulation, and using WTNPB after learning in the actual robot.

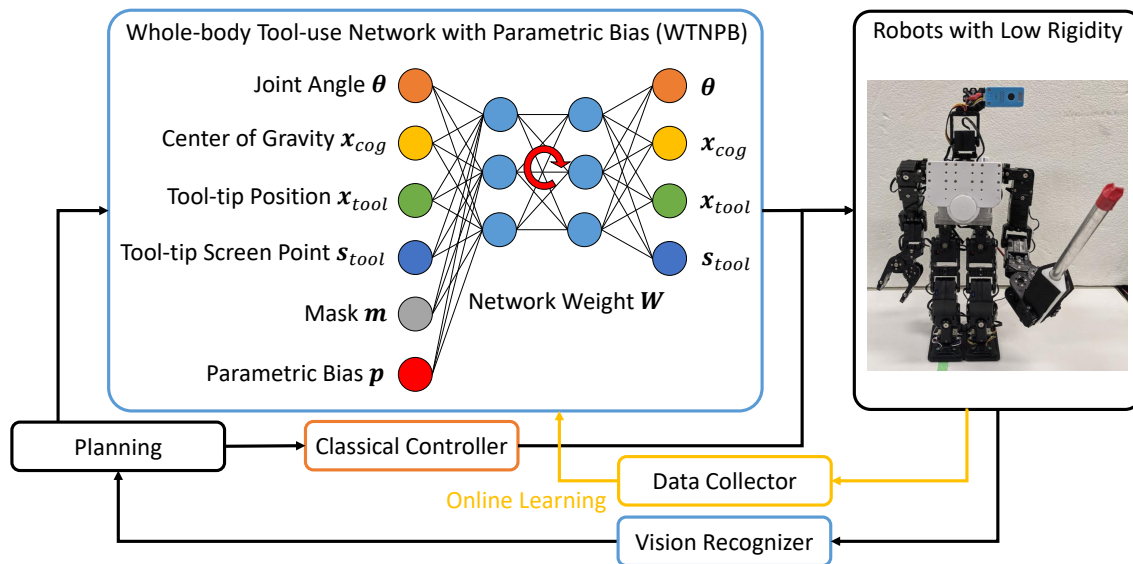


Fig. 8.34: System for the integrated experiment opening the upward window.

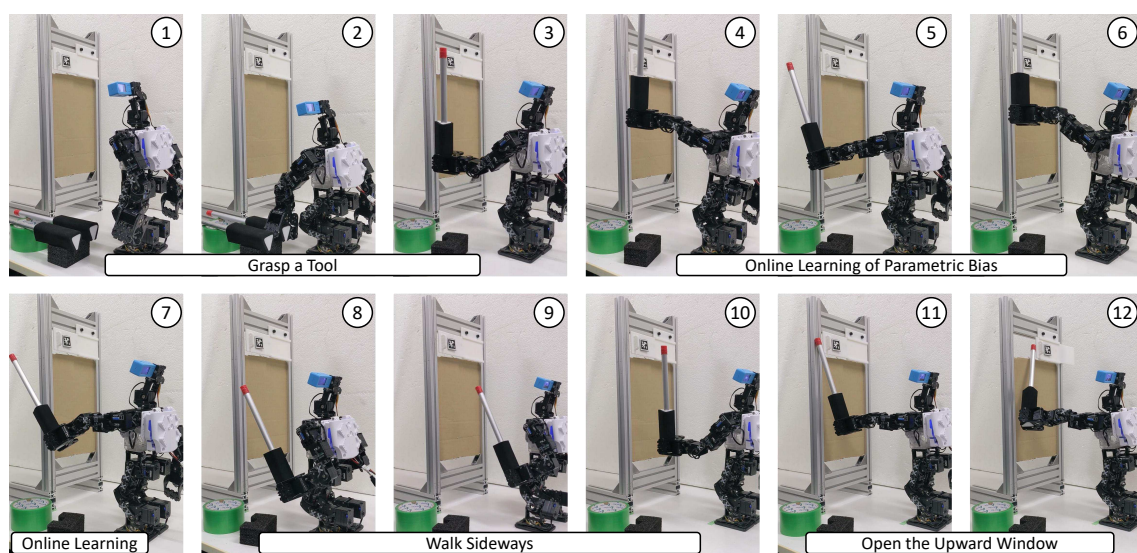


Fig. 8.35: The integrated experiment opening the upward window.

8.5 筋骨格ヒューマノイドによる柔軟布操作実験

8.5.1 概要と先行研究

モデル化の難しい布のような柔軟物のマニピュレーションはロボットにおける大きな課題の一つである。これら柔軟物体マニピュレーションには大まかに、静的マニピュレーションと動的マニピュレーションが存在する。また、それぞれについて、model-based な手法・learning-based な手法が様々な開発されてきた。静的なマニピュレーションは古くから取り組まれており、様々な model-based な手法が存在する [295, 296, 297]。learning-based な手法も近年盛んに取り組まれており、実機への応用にも成功している [298, 25]。一方、静的マニピュレーションに比べてより難しい動的マニピュレーションの例は多くはない。model-based な手法としては、山川らの cloth folding や knotting の研究がよく知られている [59, 299]。機械学習ベースの手法としては、深層予測モデルを使ったもの [284]、強化学習を使ったもの [300] がある。なお、[284] は最大 2 自由度の実機のみを使っており、[300] は多自由度ロボットを使っているものの、シミュレーションのみでの実行である。本研究ではこれらの中でも、ベッドシーツやレジャーシートを大きく広げるような、より難しい動的マニピュレーションについて扱う。後に述べるようなモデル化の容易ではない要素を複数扱うため、試行錯誤型の learning-based な手法を開発する。その中でも、実機への適用を行うため、多数の試行が必要な強化学習ではなく、深層予測モデルによる動的布操作を行う。

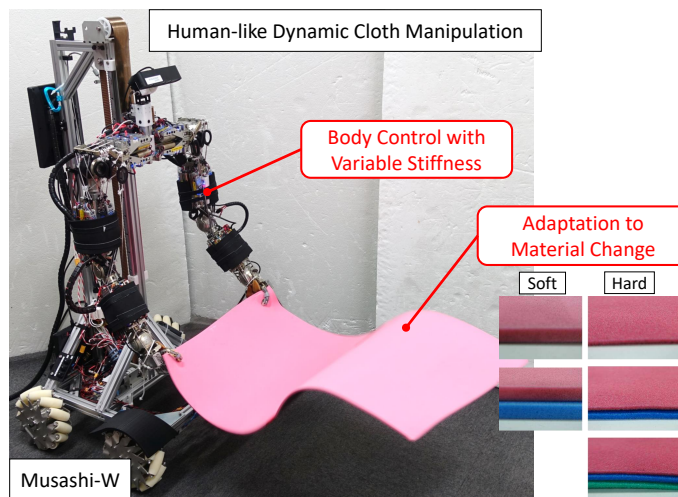


Fig. 8.36: Dynamic cloth manipulation by the musculoskeletal humanoid Musashi-W considering body control with variable stiffness and adaptation to material change.

これまでの研究において、人間のような動的布操作に足りない点がいくつかある。人間は、その柔

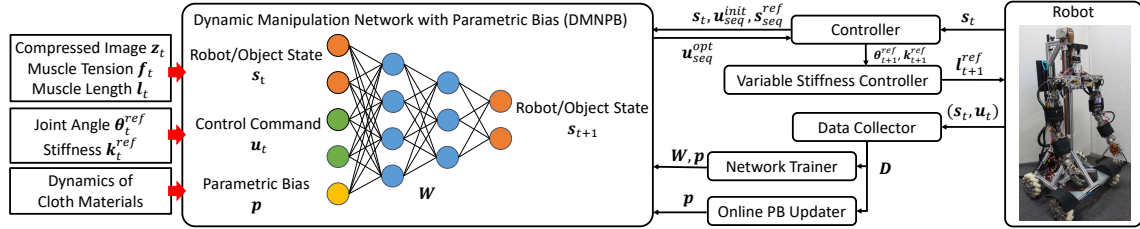


Fig. 8.37: The overview of our system: dynamic manipulation network with parametric bias (DMNPB), controller using DMNPB for dynamic cloth manipulation, variable stiffness controller for musculoskeletal humanoids, data collector for DMNPB, and online updater of parametric bias (PB).

軟な身体を使って手先を高速に動かし、また、操作対象の素材が変化しても、少し動かして特性を理解したうえでマニピュレーションを行うことができる。本研究ではこれを踏まえて、(1) より動的なマニピュレーションのための身体制御、(2) 操作対象物体の素材変化への対応、という 2 点に焦点を当てる (Fig. 8.36)。(1) は、人間のよう柔軟かつその柔軟さを自在に操ることのできる可変剛性制御が可能なロボットによるマニピュレーションを目指す。本研究では冗長な筋肉と非線形弾性要素により可変剛性制御可能な筋骨格ヒューマノイド Musashi-W (MusashiDarm [87] に車輪が付いたロボット) によりマニピュレーションを行う。このとき、身体の制御入力として、追加で剛性値を用いることで、どのように動的布操作が変化するかを考察する。(2) は、Parametric Bias [72] を用いた操作対象の素材変化への適応を目指す。Parametric Bias はニューラルネットワークにおける追加のバイアス項であり、様々な動作データについて複数の attractor dynamics を抽出することを目的として、主に模倣学習に利用されてきた [301]。本研究ではこれを、布の素材、物理的特性情報を埋め込むために用いる。新しい布を持ったとき、これを少し操作することで物理的特性を同定し、これによって正確に動的布操作を行う。本研究では、この (1) と (2) を組み込んだ深層予測モデルを構築し、より人間らしい動的布操作を行うことでその有効性を示す。

本研究の貢献は以下である。

- 制御入力への身体剛性値の追加とその効果の考察
- Parametric Bias による布の素材変化への適応
- 可変剛性と素材変化を考慮したより人間らしい動的布操作

8.5.2 可変剛性と素材変化を考慮した動的柔軟布操作

本研究で用いるネットワークを、Dynamic Manipulation Network with Parametric Bias (DMNPB) と呼ぶ。全体システムを Fig. 8.37 に示す。

DMNPB のネットワーク構造

本研究で用いる DMNPB は以下のように数式で表せる.

$$s_{t+1} = h_{dmnpb}(s_t, u, p) \quad (8.17)$$

ここで, t は現在のタイムステップ, s は操作物体やロボットの状態, u はロボット身体への制御入力, p は Parametric Bias, h_{dmnpb} はロボットと操作物体状態の制御入力による時系列変化を表す関数である. s は, 本研究の布操作では, 操作物体である布の状態と, ロボットの状態を用いる. 布の状態は, 得られた現在の画像 I_t を AutoEncoder [227] で圧縮した z_t を用いる. ロボットの状態は, ロボットごとに異なるべきであるが, 本研究で扱う筋骨格ヒューマノイドに合わせて, f_t と l_t とした (f_t, l_t は現在の筋張力・筋長を表す). よって, $s_t^T = (z_t^T \quad f_t^T \quad l_t^T)$ である. また, 本研究では $u^T = (\theta^{ref,T} \quad k^{ref,T})$ とした (ここで, θ^{ref} は指令関節角度, k^{ref} は指令身体剛性値を表す). 指令関節角度, 身体剛性の詳細については後述する. Parametric Bias は暗黙的な dynamics の違いを埋め込むことのできる値であり, 同一の素材同士については共通, 異なる素材同士については互いに異なるような値である. 様々な布の素材を使ってマニピュレーションを試すことで, p には操作物体の素材のダイナミクスに関する情報が埋め込まれる. なお, それぞれのセンサ値は得られた全データを使って正規化してからネットワークに入力している.

本研究において DMNPB は 10 層とし, 順に 4 層の全結合層, 2 層の LSTM 層 [241], 4 層の全結合層からなる. ユニット数については, $\{N_u + N_s + N_p, 300, 100, 30, 30 \text{ (LSTM のユニット数)}, 30 \text{ (LSTM のユニット数)}, 30, 100, 300, N_s\}$ とした (なお, $N_{\{u,s,p\}}$ は $\{u, s, p\}$ の次元数とする). 活性化関数は Tanh, 更新則は Adam [46] とした. 画像を圧縮する際は, 128×96 の 2 値画像 (布部分を色抽出している) について, カーネルサイズが 3, スライドが 2 の畳み込み層を 5 回適用し, 全結合層で順にユニット数 256, 3 まで次元を削減したあと, 同様に全結合層・逆畳み込み層によって画像を復元していく形を取っている. 最終層以外については Batch Normalization [45] が適用され, 活性化関数は最終層以外については ReLU [138], 最終層は Sigmoid, 更新則は Adam [46] とした. p の次元は, 使用した布の素材数よりも十分に小さい値とし, 本研究では 2 とした. また, Eq. 8.17 の実行周期は 5Hz とする.

筋骨格ヒューマノイドにおける可変剛性

本研究で扱う可変剛性制御可能な筋骨格ヒューマノイドについて, その制御入力である θ^{ref}, k^{ref} の詳細を述べる. 本研究では以下のような関節角度 θ , 筋張力 f , 筋長 l の関係を用いる [124].

$$l = h_{bodyimage}(\theta, f) \quad (8.18)$$

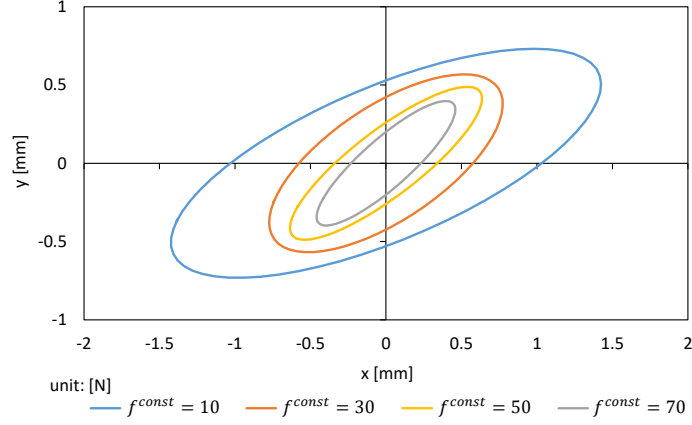


Fig. 8.38: Operational stiffness ellipsoid when adding force of 1 N while changing $f^{const} = \{10, 30, 50, 70\}$ [N].

この写像 $\mathbf{h}_{bodyimage}$ は実機におけるデータを使って学習により獲得される。この学習されたネットワークを使って制御をする際は、指令関節角度 θ^{ref} 、指令筋張力 \mathbf{f}^{ref} を決め、これに対応した筋長 $\mathbf{l}^{ref} = \mathbf{h}_{bodyimage}(\theta^{ref}, \mathbf{f}^{ref})$ を計算する。ただし、この値は測定されるべき筋長であるため、実際には、筋剛性制御 [121] におけるソフトウェアの筋の伸びを考慮した \mathbf{l}^{send} が最終的に実機に送られる [124]。

ここで、 \mathbf{f}^{ref} の与え方について考える。通常 \mathbf{f}^{ref} は実現したい θ^{ref} に必要なだけの値を入力すべきである。一方 [124] では、初めに全筋に対する \mathbf{f}^{ref} に一定の値 f^{const} を与え、ある程度指令関節角度を実現した後、その際に発揮されている筋張力 \mathbf{f} を \mathbf{f}^{ref} としてもう一度入力することで、より正確な指令関節角度を実現していた。そして、この最初に指令する f^{const} の値によってある程度身体剛性を変化させることができる。また、本研究では最終的に \mathbf{s} と \mathbf{u} の関係は学習によって獲得されるため、 θ^{ref} は正確に実現する必要はない。よって、本研究の \mathbf{u} に含まれる身体剛性値 \mathbf{k}^{ref} として f^{const} を用いることとし、 $\mathbf{l}^{ref} = \mathbf{h}_{bodyimage}(\theta^{ref}, \mathbf{f}^{const})$ によりロボットを動作させることとする。この f^{const} を変化させながら学習用データを取得することで、身体剛性値を考慮した動的布操作が可能となる。

なお、以下に実際に f^{const} によってどの程度手先の作業空間剛性が変化するかを実験した結果を載せる。Musashi-W の左腕の肘を 90 度に曲げた状態の θ^{ref} において、 f^{const} を $\{10, 30, 50, 70\}$ [N] に変化させた際の、矢状面における手先の剛性楕円を Fig. 8.38 に示す。グラフは、全方向から力を 1 N かけた時の手先変位を表している。 f^{const} の変化によって大きく手先の剛性楕円の大きさが変化していることがわかる。よって本研究では剛性値 \mathbf{k}^{ref} として f^{const} を用いる。

DMNPB の訓練

\mathbf{u} をランダムに指令する, または人間がロボットを GUI や VR デバイスにより操作し, その際の \mathbf{s} と \mathbf{u} のデータを集めていく. 同じ布を持った状態で行った, ある一回のまとまったマニピュレーションの試行 k について, データ $D_k = \{(s_1, \mathbf{u}_1), (s_2, \mathbf{u}_2), \dots, (s_{T_k}, \mathbf{u}_{T_k})\}$ を得る ($1 \leq k \leq K$, K は全試行回数, T_k はその試行 k に関する動作ステップ数とする). そして, 学習に用いるデータ $D_{train} = \{(D_1, \mathbf{p}_1), (D_2, \mathbf{p}_2), \dots, (D_K, \mathbf{p}_K)\}$ を得る. \mathbf{p}_k は試行 k に関する Parametric Bias であり, その一回の試行中については共通の値で, 異なる試行については別の値となる変数である. このデータ D_{train} を用いて DMNPB を学習させる. 通常の学習ではネットワークの重み W が更新されるが, 本研究では W と \mathbf{p}_k を同時に更新していく. これにより, \mathbf{p}_k にはそれぞれの試行におけるダイナミクスの違い, つまり, 本研究では操作した布のダイナミクスが埋め込まれることになる. 学習の際の損失関数は mean squared error であり, 全 \mathbf{p}_k は初期値を 0 として最適化される.

布素材のオンライン推定

新しい布を操作するとき, そのダイナミクスがわからなければ, 正しくマニピュレーションを実行することはできない. そこで, 少し布を動かした際に, どのように布の形状が変化するかデータの取得し, これを元に, 布のダイナミクス, つまり \mathbf{p} を推定する必要がある. まず, 学習時と同様にその布をランダムや GUI, 後に述べる制御器により操作したときのデータ D_{new} を得る. このデータを用いて, 学習時とは異なり, W を固定したうえで, \mathbf{p} のみ更新していく. つまり, 布のダイナミクスに関する項だけを更新することで, DMNPB のダイナミクスを D_{new} に合致させていく. このときの更新則は SGD とした.

DMNPB による動的布操作

DMNPB を使った動的布操作制御について述べる. まず, 布の指令状態となる画像 I^{ref} を取得し, AutoEncoder により \mathbf{z}^{ref} に圧縮する. \mathbf{f}^{ref} と \mathbf{l}^{ref} は $\mathbf{0}$ とし, これらを合わせて \mathbf{s}^{ref} を生成する. 次に, DMNPB の展開数 $N_{seq}^{control}$ を決め, 最適化すべき \mathbf{u} の $N_{seq}^{control}$ ステップの時系列 $\mathbf{u}_{seq}^{opt} (\mathbf{u}_{[t, t+N_{seq}^{control}-1]}^{opt})$ の略) に初期値 \mathbf{u}_{seq}^{init} を設定する. 以下のように損失関数を計算し, これをもとに W, \mathbf{p} は固定した状

態で \mathbf{u}_{seq}^{opt} を更新していく.

$$L = h_{loss}(\mathbf{s}_{seq}^{ref}, \mathbf{s}_{seq}^{pred}) \quad (8.19)$$

$$\mathbf{g} = \partial L / \partial \mathbf{u}_{seq}^{opt} \quad (8.20)$$

$$\mathbf{u}_{seq}^{opt} \leftarrow \mathbf{u}_{seq}^{opt} - \gamma \mathbf{g} / \|\mathbf{g}\|_2 \quad (8.21)$$

ここで, h_{loss} は損失関数 (後に説明する), \mathbf{s}_{seq}^{ref} は \mathbf{s}^{ref} を $N_{seq}^{control}$ ステップ並べた指令布状態の時系列, \mathbf{s}_{seq}^{pred} は現在の状態 \mathbf{s}_t に \mathbf{u}_{seq}^{opt} を順に入れていったときに予測された \mathbf{s} の時系列 $\mathbf{s}_{[t+1, t+N_{seq}^{control}]}^{pred}$, $\|\cdot\|_2$ は L2 ノルム, γ は学習率を表す. つまり, 予測される \mathbf{s} の時系列を指令値に近づけるように時系列の制御入力を更新していく. なお, \mathbf{W} は訓練時に得られたもの, \mathbf{p} はオンライン学習により得られたものを使う. ここで, γ は一定値でも良いが, 本研究ではより早い収束性のため, 複数の γ を使って \mathbf{u}_{seq}^{opt} を更新し, その中で最も L の小さかった \mathbf{u}_{seq}^{opt} を使うという処理を $N_{iter}^{control}$ 回繰り返す. γ は, $[0, \gamma_{max}]$ を対数的に等間隔に $N_{batch}^{control}$ 等分した値を用いる (γ_{max} は γ の最大値). また, \mathbf{u}_{seq}^{opt} の初期値 \mathbf{u}_{seq}^{init} は, 前ステップで最適化された値 \mathbf{u}_{seq}^{prev} ($\mathbf{u}_{[t, t+N_{seq}^{control}-1]}^{prev}$ の略) を一ステップ分左ヘシフトし, 最後の項を複製したもの $\mathbf{u}_{\{t+1, \dots, t+N_{seq}^{control}-1, t+N_{seq}^{control}-1\}}^{prev}$ を用いる. これにより, これまでの最適化結果を考慮したより早い収束が得られる. 最終的に得られた \mathbf{u}_{seq}^{opt} の現在のタイムステップにおける指令値 \mathbf{u}_t^{opt} を実機に送る.

ここで, 動的布操作のような動的な動作に特徴的である, 指令状態に対して制御入力の次元が小さなタスクについて考える. これは例えば, ベッドメイキングの際にシーツを大きく広げるような動作や, レジャーシートを空中で広げる動作である. この布を広げた状態の画像 \mathbf{z}^{ref} を含む指令値を \mathbf{s}^{ref} として, これを実現しようとするを考える. このとき, もし一度でうまくいかなかった場合, 制御入力の次元が小さいため, そこから微修正ができるとは限らず, また最初から布を手前にもってきて一気に開くような一連の動作を生成する必要がある. つまり, 一度最初の動きで損失がある程度下がっても, さらに損失を下げるためには一度大きく身体を動かし損失を大きくする状態を経由する必要がある. そのため, \mathbf{s}^{ref} を並べたベクトルを \mathbf{s}_{seq}^{ref} としただけでは, 一度大きく広げようとした後は, ほとんど動かない \mathbf{u}_{seq}^{opt} が生成されてしまう. また, 空中で布を広げるような状態を \mathbf{s}^{ref} とした場合は, その状態が常年实现できるわけではないため, 同じような問題が起きる. つまり, 何度も周期的に広げる動作を行うことで, 指令状態を実現する必要がある. この特徴を踏まえたうえで, 本研究では周期的運動の周期ステップ数 $N_{periodic}^{control}$ を決め, その周期ごとに h_{loss} を変化させる. 本研究では以下のように h_{loss} を表現する.

$$h_{loss}(\mathbf{s}_{seq}^{ref}, \mathbf{s}_{seq}^{pred}) = \|\mathbf{m}_t \otimes (\mathbf{z}_{seq}^{ref} - \mathbf{z}_{seq}^{pred})\|_2 + w_{loss} \|\mathbf{f}_{seq}^{pred}\|_2 \quad (8.22)$$

ここで, $\{z, f\}_{seq}^{\{ref, pred\}}$ は $s_{seq}^{\{ref, pred\}}$ の $\{z, f\}$ のみを抜き出したもの, w_{loss} は重み付けの係数である. $m_t (\in \{0, 1\}^{N_{seq}^{control}})$ は $N_{periodic}^{control}$ ステップごとに 1 が出現し, その他は 0 のベクトルである. そしてこれは, 一ステップごとに左へシフトし, 右からは $N_{periodic}^{control}$ に応じて 0 または 1 が挿入される (e.g. もし $N_{seq}^{control} = N_{periodic}^{control} = 4$ ならば順に $(0, 1, 0, 0) \rightarrow (1, 0, 0, 0) \rightarrow (0, 0, 0, 1)$ になる). これにより, $N_{periodic}^{control}$ ごとに布の状態 z を指令値に近づけることができ, 周期的かつ, 一瞬しか実現できない状態を扱う動的布操作が可能となる. Eq. 8.22 の右辺第二項は, なるべく大きな筋張力が出さないようにするための項である.

本研究では, $N_{seq}^{control} = 8$, $N_{batch}^{control} = 30$, $\gamma_{max} = 1.0$, $N_{iter}^{control} = 3$, $w_{loss} = 0.001$, $N_{periodic}^{control} = 8$ とした.

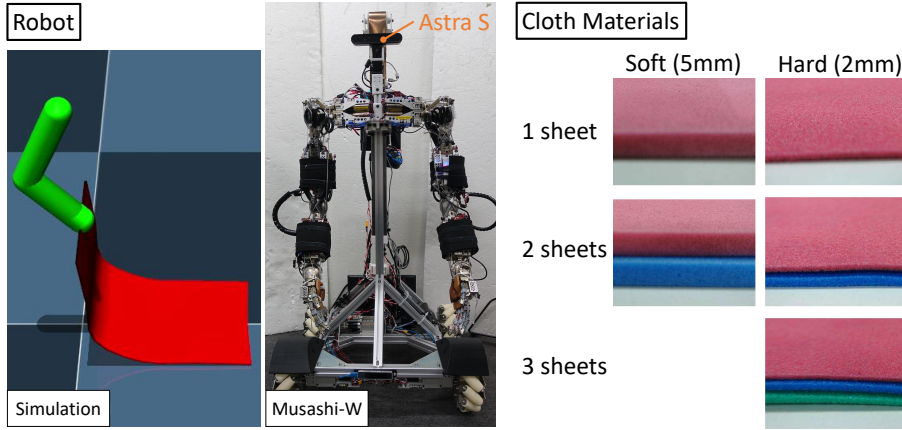


Fig. 8.39: Experimental setup: the simulated simple robot and the musculoskeletal humanoid Musashi-W used in this study and cloth materials of soft and hard type polyethylene foam.

8.5.3 実験

実験セットアップ

本研究で用いるロボット・布の種類を Fig. 8.39 に示す. 本研究ではまず検証として, Mujoco [257] を用いたシミュレーションにおいて, 簡単な 2 自由度ロボットを使った布操作実験を行う. この際, 布は 3×6 の質点の集まりとしてモデル化されており, それら質点の間の damping C_{damp} と布全体の重さ C_{mass} を変更することができる. 次に, 筋骨格双腕 MusashiDarm [87] にメカナム台車と z 軸のスライダーが追加された Musashi-W を用いて実験を行う. 頭部にはカメラとして, Astra S (Orbbec 3D Technology International, Inc.) がついている. 本研究ではポリエチレンフォームの soft type (厚さ 5 mm)

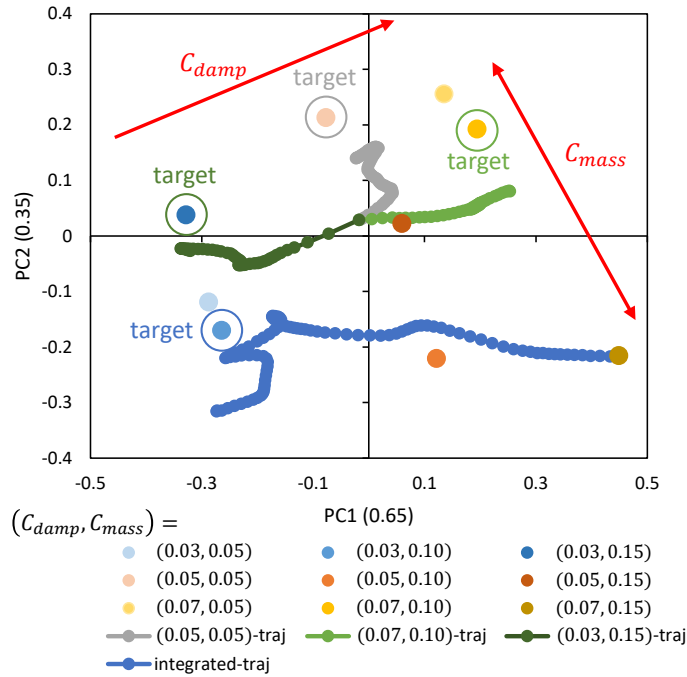


Fig. 8.40: Simulation experiment: the trained parametric bias when setting $C_{damp} = \{0.03, 0.05, 0.07\}$ and $C_{mass} = \{0.05, 0.10, 0.15\}$, the trajectory of online updated parametric bias when setting $(C_{damp}, C_{mass}) = \{(0.05, 0.05), (0.07, 0.10), (0.03, 0.15)\}$, and the trajectory of online updated parametric bias when setting $(C_{damp}, C_{mass}) = (0.03, 0.10)$ for the integrated experiment.

と hard type (厚さ 2 mm) (TAKASHIMA カラーシート, WAKI Factory, Inc.) の二種類の素材を布として扱う。本来であれば布を用いるべきであるが, Musashi-W の関節速度制限から今回はポリエチレンフォームを用いた。なお, soft type は 1 枚または 2 枚, hard type は 1 枚, 2 枚, または 3 枚重ねたものを用意する (soft-1, soft-2, hard-1, hard-2, hard-3 のように表記する)。また, 手で持ちやすいように金属棒を布の一旦にくくりつけている。

Simulation と Musashi-W の腕は矢状面でのみの動作とし, 肩と肘の pitch 軸 2 自由度のみを動かす。Simulation についてはハードウェアの剛性値に関するパラメータはないが, Musashi-W については剛性値を含めて制御入力 3 自由度である。画像は布の赤 (表部分は赤, 裏は青や黒の異なる色になっている) を色抽出・2 値化・拡大・縮小・リサイズを施した後, AutoEncoder により圧縮される。

シミュレーション実験

布の特性を $C_{damp} = \{0.03, 0.05, 0.07\}$, $C_{mass} = \{0.05, 0.10, 0.15\}$ と変更しながら θ^{ref} をランダムに指令した場合 (Random) を約 50 秒間, GUI (関節角度を GUI で動かす) で人間が動作させた場合を

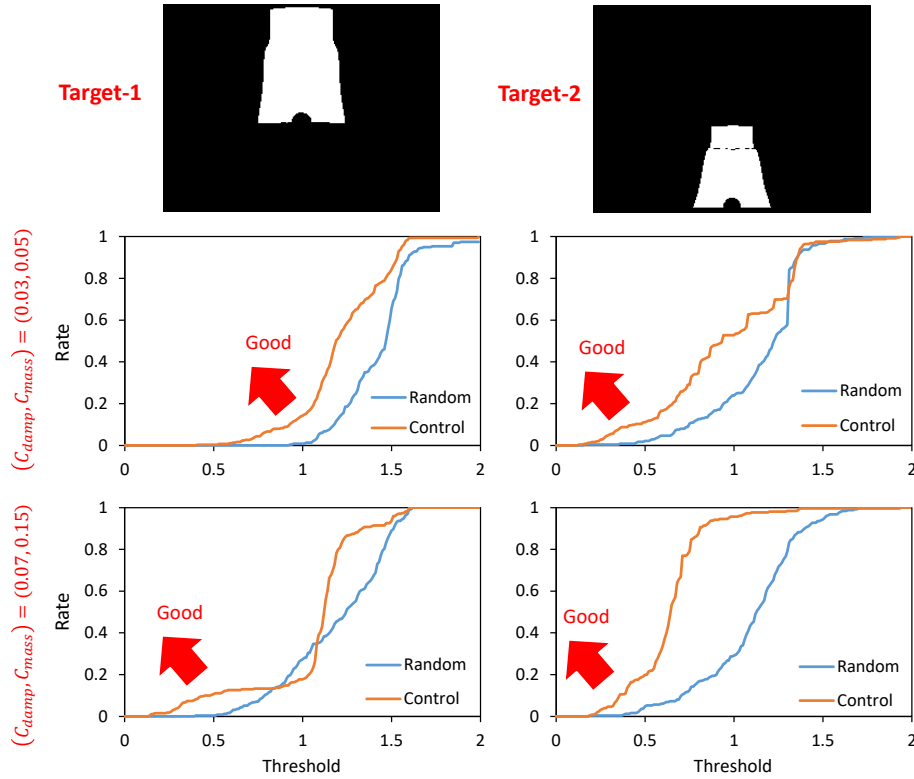


Fig. 8.41: Simulation experiment: the rate (y-axis) of $\|z^{ref} - z\|_2 < \text{threshold}$ (x-axis) when conducting experiments of dynamic manipulation of two cloths Target-1 and Target-2 when setting $(C_{damp}, C_{mass}) = \{(0.03, 0.05), (0.07, 0.15)\}$ regarding Control and Random.

約 50 秒間行いデータを取得した。計 4500 のデータを使い、LSTM の展開数を 30、バッチ数を 300、エポック数を 300 として DMNPB を学習した。その際に得られたそれぞれの布に関する Parametric Bias に Principle Component Analysis をかけ、2 次元平面内に配置した様子を Fig. 8.40 に示す。主に PC1 の軸において、 C_{damp} の大きさに従って PB が配置されていることがわかる。一方で、一部の PB は C_{mass} の大きさに沿って配置されておらず、布のダイナミクスに対する C_{mass} の影響は小さい、また、 C_{mass} によるダイナミクスの変化は複雑であることが読み取れる。

ここで、 $(C_{damp}, C_{mass}) = \{(0.05, 0.05), (0.07, 0.10), (0.03, 0.15)\}$ のそれぞれの布を持った状態で、Random の動作とオンライン素材推定を約 40 秒間走らせる。このときの Parametric Bias の軌跡 (traj) を Fig. 8.40 に示す。なお、PB の初期値は $\mathbf{0}$ とする (PCA をかけているため、PB の原点がグラフ上で原点にあるとは限らない)。現在の PB は、徐々に現在の布のダイナミクスにおいて訓練時に得られたそれぞれの PB の値へ近づいていくことがわかる。特に C_{damp} の軸についてはその精度は正確である。一方で、 C_{mass} の軸についてはある程度正しく素材のパラメータを認識できているものの、その精度は

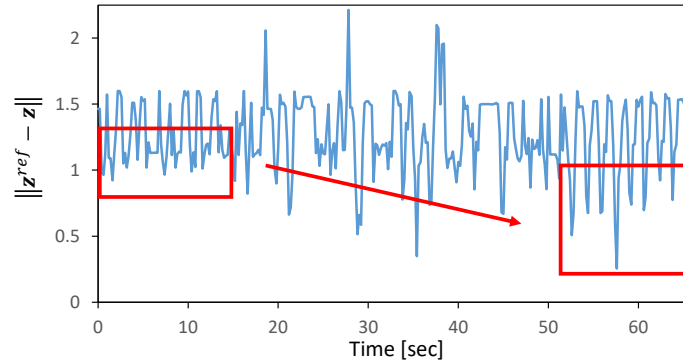


Fig. 8.42: Simulation experiment: the transition of $\|z^{ref} - z\|_2$ for the integrated experiment of online estimation and dynamic cloth manipulation.

C_{damp} の軸に比べると低いことがわかる。

次に、布の特性を $(C_{damp}, C_{mass}) = \{(0.03, 0.05), (0.07, 0.15)\}$ として、現在 PB の値を、操作する物体の訓練時に得られた PB に設定した状態で、DMNPB を使った制御を行う。このとき、布の指令画像として Fig. 8.41 の 2 つの画像 Target-1 と Target-2 を与える。また、本研究の制御を行った場合を Control、ランダムな試行を Random と表す。2 種類の布について、2 種類の画像を指令値として Control と Random のそれぞれを 50 秒間行う。その際に、 $\|z^{ref} - z\|_2$ がある閾値 (x 軸) よりも低い割合 (y 軸) を Fig. 8.41 に示す。Fig. 8.41 はグラフが左上方向に膨らむほど、より正確に指令画像状態が実現できていることを表す。どのケースについても、Control は Random を上回り、指令画像を実現することに成功している。Target-1 の方が Target-2 より難しいため、特に $(C_{damp}, C_{mass}) = (0.05, 0.05)$ のときはその違いが如実に現れている。また、 C_{damp} が大きい方が布の変形が遅くなり布が浮き上がりやすくなるため、 (C_{damp}, C_{mass}) が $(0.07, 0.15)$ のときの方が、 $(0.03, 0.05)$ のときに比べて比較的指令画像を正しく実現できている。

最後に、現在の布の特性を $(C_{damp}, C_{mass}) = (0.03, 0.10)$ 、現在の PB 値を $(C_{damp}, C_{mass}) = (0.07, 0.15)$ で訓練時に得られた値として、オンライン素材推定と DMNPB を使った制御を同時に実行する実験を行う。なお、指令画像は Fig. 8.41 の Target-1 としている。この際の PB の遷移 (integrated-traj) を Fig. 8.40 に、 $\|z^{ref} - z\|_2$ の遷移を Fig. 8.42 に示す。現在の PB は徐々に $(C_{damp}, C_{mass}) = (0.03, 0.10)$ で訓練時に得られた値へと近づいていくことがわかる。また、PB が正確になるにつれて、 $\|z^{ref} - z\|_2$ が周期的により小さな値を示すようになる。つまり、オンラインの素材推定と本学習制御は両立して実行することが可能であり、PB が正確になるにつれて制御も正確になる。

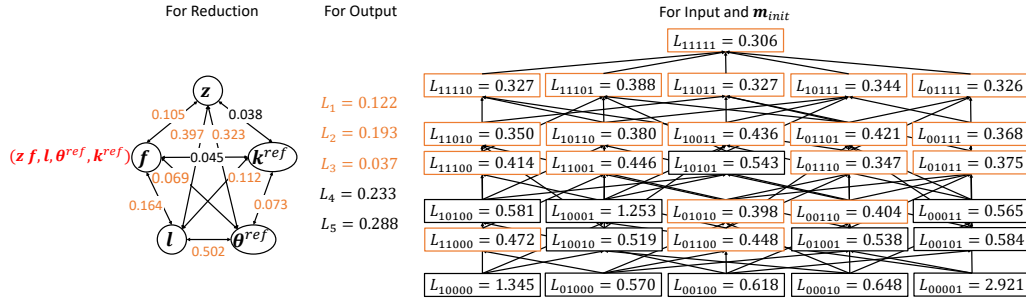


Fig. 8.43: The automatic network decision of DMNPB.

Musashi-W による実機実験

soft-1, soft-2, hard-1, hard-3 の布をそれぞれ持った状態で, θ^{ref} と k^{ref} をランダムに指令した場合 (Random) を約 80 秒間, GUI (関節角度を GUI で動かし, 関節剛性はランダム) で人間が動作させた場合を約 80 秒間行いデータを取得した. なお, ランダムに指令する k^{ref} は $[10, 70]$ [N] の間とした. 計 3000 のデータを使い, LSTM の展開数を 20, バッチ数を 1000, エポック数を 600 として DMNPB を学習した. このときのネットワーク構造の自動決定について Fig. 8.43 に示す. まず, それぞれのセンサ値について相互情報量を計算し, x に不要な値がないことを確認した. 次に, L_i を計算し, $C_{thre}^{out} = 0.2$ として $x_{out}^T = (x_1^T, x_2^T, x_3^T)$ として用いることに決定した. 最後に, それぞれのマスク m について L_m を計算し, $C_{thre}^{init} = 0.5$ と設定することで, 実行可能なマスク集合が決定され, これに必要な x_{in} として, x の全ての値が設定された. その際に得られたそれぞれの布に関する Parametric Bias に Principle Component Analysis をかけ, 2 次元平面内に配置した様子を Fig. 8.44 に示す. soft-1, soft-2, hard-1, hard-3 のそれぞれが四隅に配置されており, 布の厚さ・固さ方向が揃っていることがわかる.

ここで, hard-1, hard-2, hard-3 のそれぞれの布を持った状態で, Random の動作とオンライン素材推定を約 30 秒間走らせる. このときの Parametric Bias の軌跡 (traj) を Fig. 8.44 に示す. なお, PB の初期値は $\mathbf{0}$ とする. それぞれの PB は最終的に布の厚みに従って直線上に並んだ. hard-3 の PB は訓練時に得られた PB に近い値を示す一方, hard-1 の PB は訓練時に得られたものから少し外れ, hard-3 の PB に寄っていた. 訓練時に使っていないデータである hard-2 については, hard-1 と hard-3 の中間程度に位置している. よって, PB の空間が Cloth thickness や Cloth stiffness に従って自己組織化されていると考える.

次に, PB の値を, 操作する物体の訓練時に得られた PB に設定した状態で, DMNPB を使った制御を行った. Fig. 8.45 の左図のような布の初期状態から, 本制御により, Fig. 8.45 の右図にあるような, 布を空中で広げた状態へと近づけていく. hard-1 と soft-1 をそれぞれ操作したときの指令関節角度 θ^{ref} ,

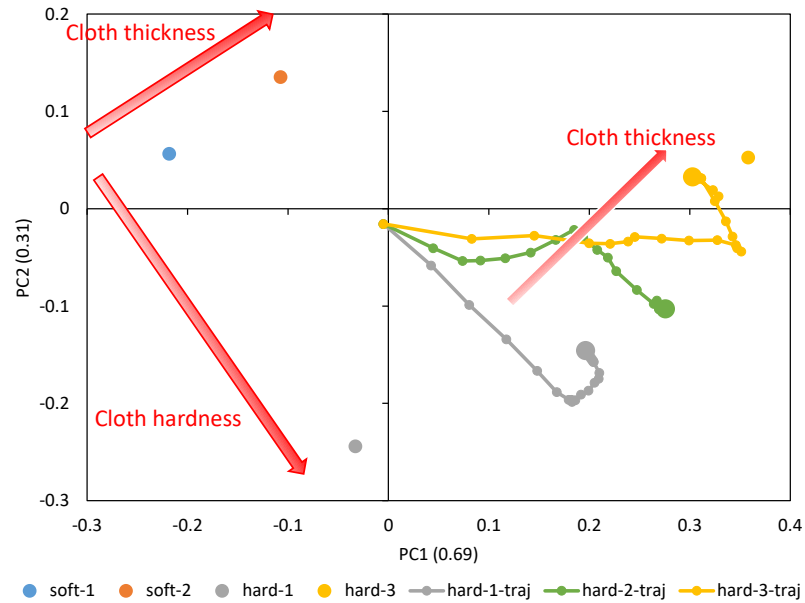


Fig. 8.44: Actual robot experiment: the trained parametric bias for soft-1, soft-2, hard-1, and hard-3, and the trajectory of online updated parametric bias for hard-1, hard-2, and hard-3.

測定された関節角度 θ , 指令剛性を表す筋張力 f^{const} , 指令画像状態と現在画像状態の誤差 $\|z^{ref} - z\|_2$ を Fig. 8.46, Fig. 8.47 に示す. なお, θ_{s-p} , θ_{e-p} はそれぞれ肩と肘のピッチ関節の角度を表す. 赤い枠で囲った部分は, 振り上げた肩と肘を大きく下に降ろしている主要な部分である (関節角度が大きいほど下に手を下ろしている状態である). このとき, f^{const} を見ると, 多くの場合で, 前半は剛性を高くし, 後半は剛性を低くしていることがわかる. これにより, 手先スピードが上がり, 布が広がって浮き上がることで Fig. 8.45 の右図の状態が実現され, $\|z^{ref} - z\|_2$ の値がより低くなっていることがわかる. 実際に, k^{ref} に対して Eq. 8.21 を行わず, f^{const} を固定 (10 N) として実験を行った場合 (w/o stiffness) と本研究の制御を行った場合 (w/ stiffness) の手先速度の絶対値を比較した図を Fig. 8.48 に示す. これは, Fig. 8.45 の左図の状態から 20 秒の実験を 5 回行い, その間の最大手先速度の平均と分散を表したものである. 剛性変化を使った場合のほうが約 12% 程度速度が大きくなっていることがわかる. また, Fig. 8.46, Fig. 8.47 の soft-1 と hard-1 の違いを見ると, hard-1 では θ_{e-p} と θ_{s-p} がほぼ同時に動き, また手を振り上げてから下ろすまでが素早い. 一方, soft-1 では, θ_{s-p} が θ_{e-p} よりも少し後に動き出ししており, また手を振り上げてから振り下ろすまでが緩やかな場合が多い. これは, 柔らかい布は最後はゆったりと手をおろしても滞空時間が確保できるのに対して, 固い布はすぐに手を降ろさないと滞空時間が確保できないことが理由であると考えられる. このように, PB の違いによって対象の操作の仕方が大きく変わっていることがわかる.

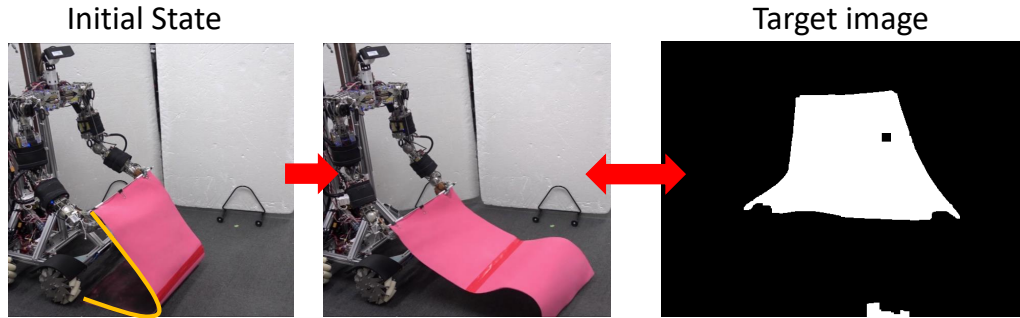


Fig. 8.45: Experimental setup for the actual robot experiment: the initial state of cloth and target image of spread-out cloth.

最後に、いくつかの条件で布操作を行い、それらを比較した。PB が正しい (soft-1 の操作に対して soft-1, hard-1 の操作に対して hard-1 の PB を用いている) 場合の試行を Correct, PB が間違っている (soft-1 の操作に対して hard-1, hard-1 の操作に対して soft-1 の PB を用いている) 場合の試行を Wrong, ランダムな試行を Random とする。また, PB は正しいが, \mathbf{k}^{ref} に対して Eq. 8.21 を行わず, f^{const} を固定 (10 N) として実験を行った場合を Correct w/o stiffness とする。この 4 つの条件において, hard-1 と soft-1 のそれぞれの布を Fig. 8.45 の左図の状態から 25 秒の実験を 5 回行う。その際に, $\|\mathbf{z}^{ref} - \mathbf{z}\|_2$ がある閾値 (x 軸) よりも低い割合 (y 軸) の平均を Fig. 8.49 に示す。また, Fig. 8.49 の一部を取り出した, $\|\mathbf{z}^{ref} - \mathbf{z}\|_2 < 0.5$ の割合の平均と分散を Fig. 8.50 に, $\|\mathbf{z}^{ref} - \mathbf{z}\|_2$ の最小値の平均と分散を Fig. 8.51 に, $\|\mathbf{z}^{ref} - \mathbf{z}\|_2$ の最大値の平均と分散を Fig. 8.52 に示す。まず, Fig. 8.49 はグラフが左上方向に膨らむほど, より指令画像状態が実現できていることを表す。hard-1, soft-1 のどちらにおいても Correct が最も良く, Random が最も悪い。そして, Fig. 8.50 を見ると, その特性が定量的に見て分かる。hard-1 の方がより難しいため全体的に実現率が低いが, Correct に対して Wrong, Correct w/o stiffness の実現率が低いという傾向は一致している。Fig. 8.51 を見ると, $\|\mathbf{z}^{ref} - \mathbf{z}\|_2$ の最小値の傾向も Fig. 8.50 と一致しており, 最小値が低いものほど実現率が高い割合も大きい。一方, Fig. 8.52 は Correct が最も高く, Random が最も低い。これは, 指令画像を実現するためには大きく動的に状態を変化させなければならないため, より指令画像を正しく実現できる方法ほど指令画像と離れた状態が起きやすいことを表していると考えられる。

テーブルセッティング統合実験

最後に, 動的布操作を含む一連の動作実験を行った。Musashi-W が hard-1 を手に取り, 本手法によりテーブルの上にテーブルクロスとしてこれを敷き, その上にお菓子の入ったかごを載せる。この際

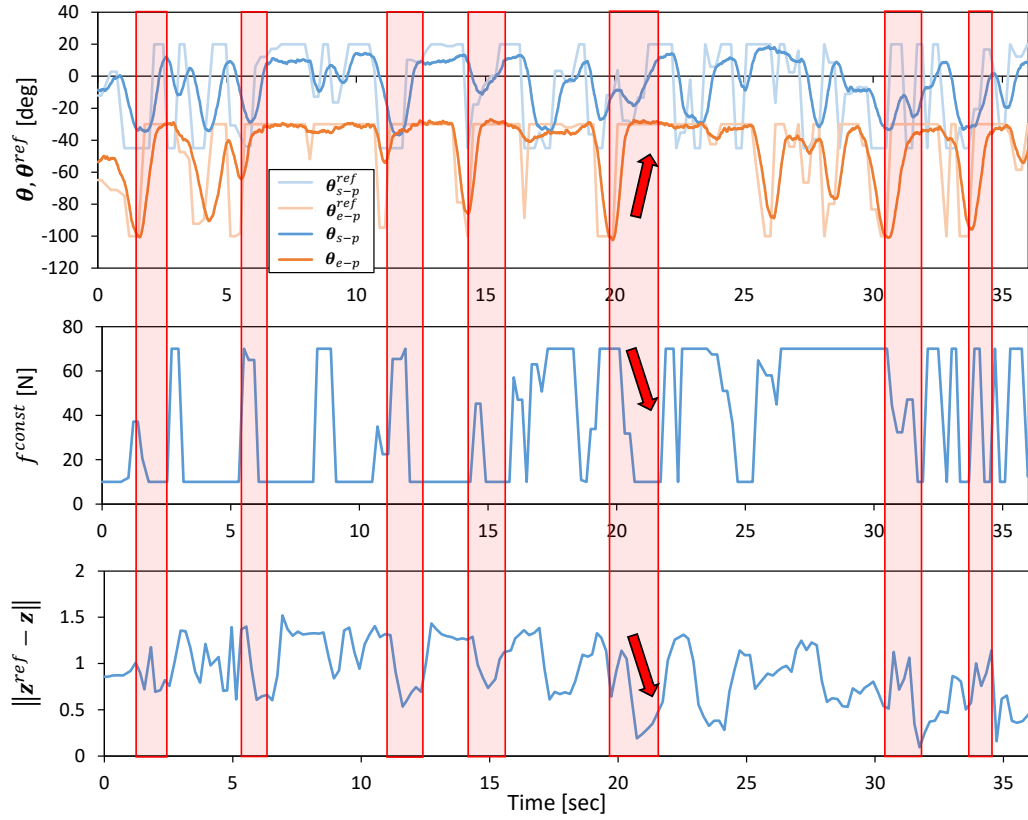


Fig. 8.46: Actual robot experiment: the transition of θ^{ref} , θ , f^{const} , and $\|z^{ref} - z\|_2$ when conducting an experiment of dynamic manipulation of hard-1.

の全体システムを Fig. 8.53 に示す。認識器として関節角度推定器や布素材認識器, 物体認識器が働いている。これらの情報を元に動作計画を行い, 通常の古典的動作制御, または DMNPB を使った動的布操作が行われる。計算された指令関節角度は 6.3 節の関節-筋空間マッピングに基づき指令筋長へと変換され, 指令車体速度は古典的制御により車輪モータへの指令入力へと変換される。筋骨格型の双腕部については反射である 5.3 節の筋弛緩制御が働くと同時に, 全てのモータが 5.2 節のモータ温度制御により最大出力を管理される。また, 前述の認識器である関節角度推定器や布素材推定器は関節-筋空間マッピングまたは DMNPB を用いて行われている。加えて, DPMPB や関節-筋空間マッピング, モータ温度制御モデルは実機データをもとにオンラインされる。

その様子を Fig. 8.54 に示す。まず, 布の Point Cloud を認識してビジュアルフィードバックにより親指と人差し指の間で布を挟み, これを把持する。次に, 予め同様の機等のセットアップで学習された動的布操作により, この布を机の上に広げる。最後に, カゴの Point Cloud を認識してビジュアルフィードバックによりかごを両手で挟んで把持し, これをテーブルの上に置くことに成功した。

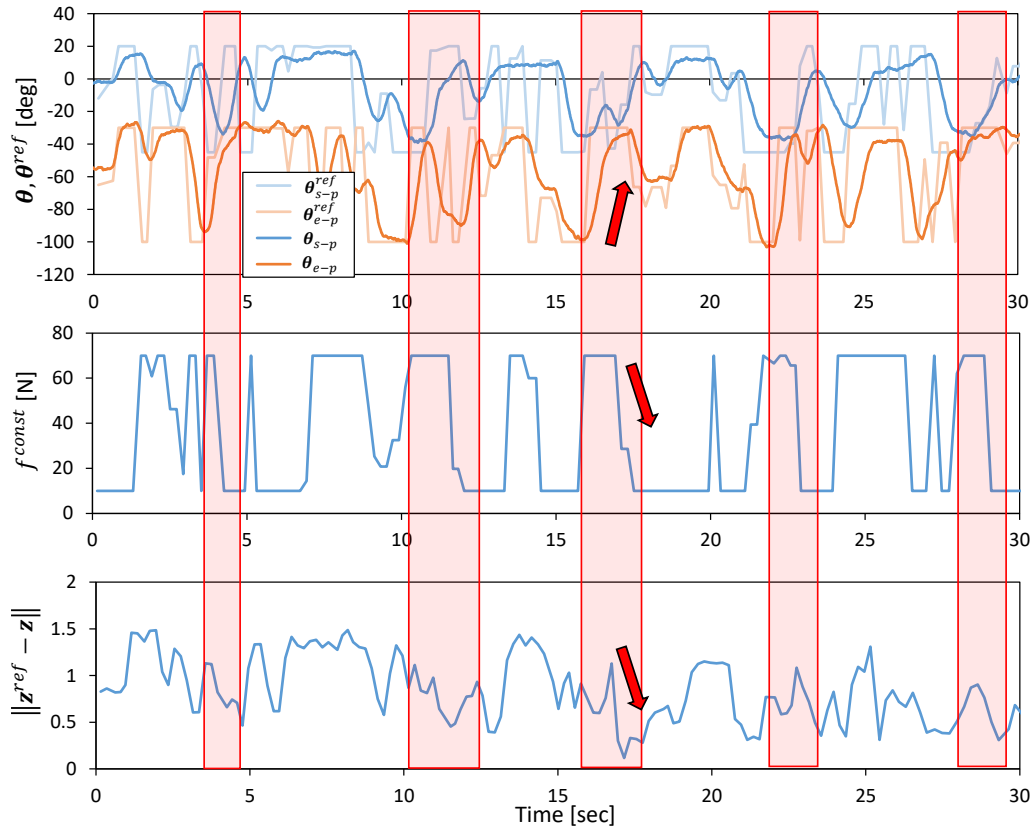


Fig. 8.47: Actual robot experiment: the transition of θ^{ref} , θ , f^{const} , and $\|z^{ref} - z\|$ when conducting an experiment of dynamic manipulation of soft-1.

8.5.4 議論

本研究の実験から得られた結果について議論する。まず、シミュレーション実験から、PB の値が布のパラメータに応じて自己組織化されることがわかった。布の操作動作データから、現在持っている布のダイナミクスをオンラインで推定することができる。一方で、ダイナミクスにあまり寄与しないパラメータについては、正確に推定することが難しいこともわかった。次に、布の特性、指令画像を変化させながら動的布操作実験を行うことで、本手法が正確に指令画像を実現できることがわかった。PB を変化させることで、様々な布の特性に対応でき、また、指令画像も任意に与えることが可能である。最後に、オンライン素材推定と動的布操作の統合実験から、これらは同時に実行できること、また、現在の PB が現在の布の特性に近くなるほどより正確に指令画像を実現できるようになることがわかった。

実機実験から、PB の値が布の固さ・枚数に応じて自己組織化されることがわかった。シミュレーション同様、布の操作動作データから、現在持っている布のダイナミクスをオンラインで推定するこ

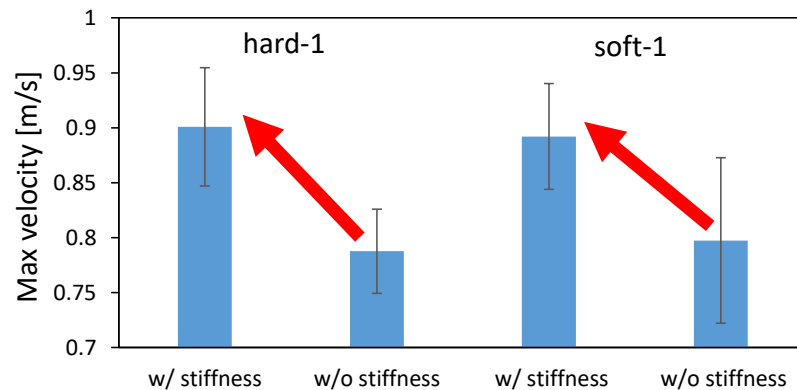


Fig. 8.48: Actual robot experiment: the average and standard deviation of the maximum velocity of end effector when conducting experiments of dynamic manipulation of hard-1 and soft-1.

とができる。また、訓練時には無いデータについても訓練時のデータを内分するような正しい点にダイナミクスが推定されることがわかった。次に、PB、つまり布のダイナミクスに応じて布操作の動作が大きく変化することがわかった。関節角度の連動のさせ方やスピードなどが、操作する布の素材に合わせて適切に変更されることがわかった。また、剛性値は手先の動作スピードに影響を及ぼし、これを最適化した場合は、最適化しない場合に比べて手先速度が12%程度上昇することがわかった。PBが現在の布に合致していない場合や剛性値を最適化しない場合は、PBが正しく、可変剛性を利用する場合に比べると大きく性能が低下した。また、指令画像を正しく実現するためには、指令画像から大きく離れた状態を経由する必要があることがわかった。これらから、Parametric Biasによって布素材のダイナミクスの埋め込み・オンラインの推定ができること、可変剛性制御を陽に利用することで手先速度を向上させることができること、これらを統合した深層予測モデル学習と制御入力への誤差逆伝播を用いることで、より正確に指令布状態を実現できることがわかった。

本手法はロボットの動的布操作の能力を大きく底上げすることができる一方、まだ課題も残る。まず、本研究では身体剛性をある一定値により代替しているが、実際の人間ではより細かく身体剛性を設定することができる。今後、身体剛性の自由度についても議論する必要がある。次に、本研究は手から布が離れたり、布を持ち替えたり等、大きく非線形な変化が起こる場合に対処することは難しいと考える。実機のみでの試行では、そのような高い非線形性・不連続性・高い次元の状態を扱うことができる手法はなく、今後の課題である。最後に、まだ人間の動的布操作への到達にはほど遠い。人間は、柔軟物体全体を万遍なく見るのではなく、細かいシワに着目しながらそれを伸ばしつつ広げることも可能である。また、現状人間ほど俊敏に柔らかく動けるハードウェアではないため、今後ハードウェア・ソフトウェアの両面からの開発が必要となる。

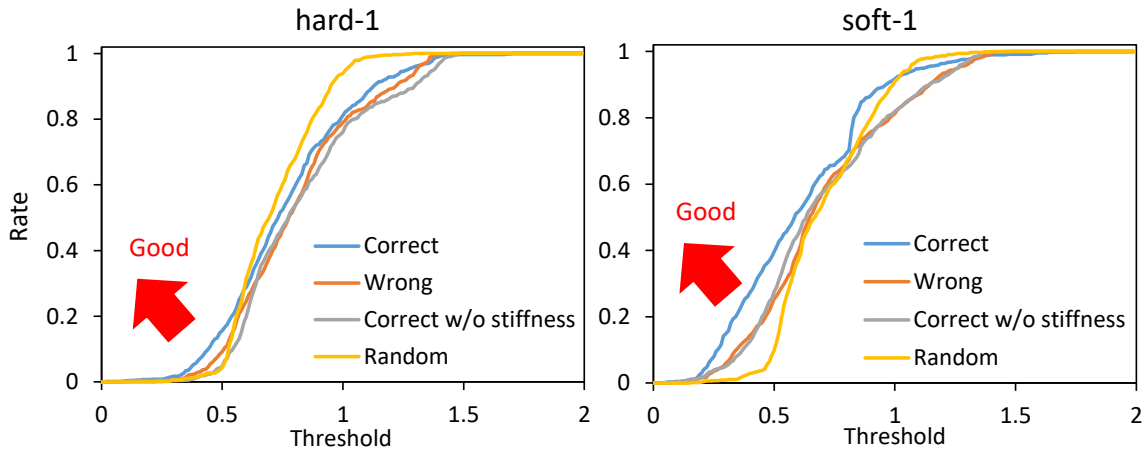


Fig. 8.49: Actual robot experiment: the rate (y-axis) of $\|z^{ref} - z\|_2 < \text{threshold}$ (x-axis) when conducting experiments of dynamic manipulation of hard-1 and soft-1 regarding Correct, Wrong, Correct without stiffness, and Random.

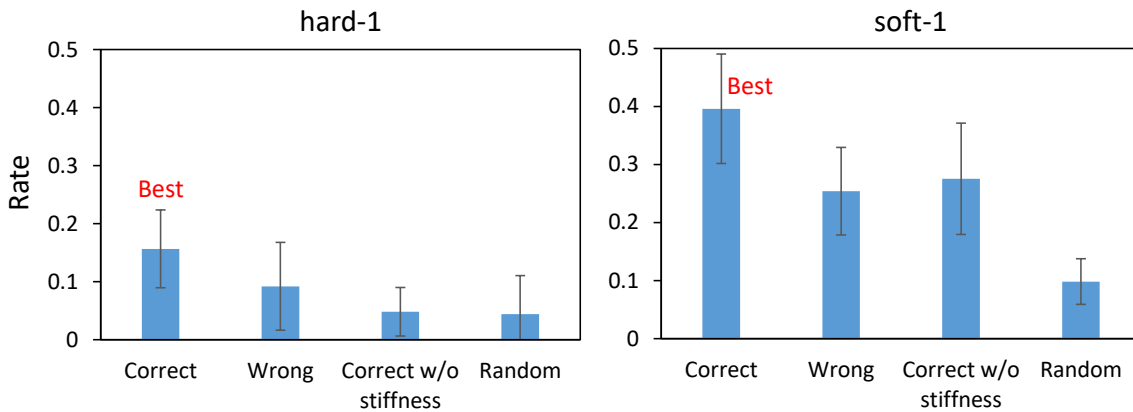


Fig. 8.50: Actual robot experiment: the rate of $\|z^{ref} - z\|_2 < 0.5$ when conducting experiments of dynamic manipulation of hard-1 and soft-1.

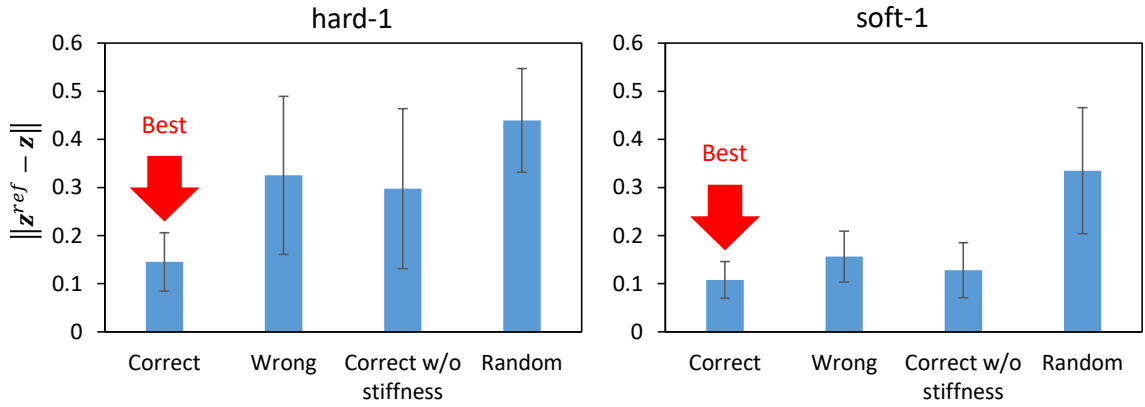


Fig. 8.51: Actual robot experiment: the minimum value of $\|z^{ref} - z\|_2$ when conducting experiments of dynamic manipulation of hard-1 and soft-1.

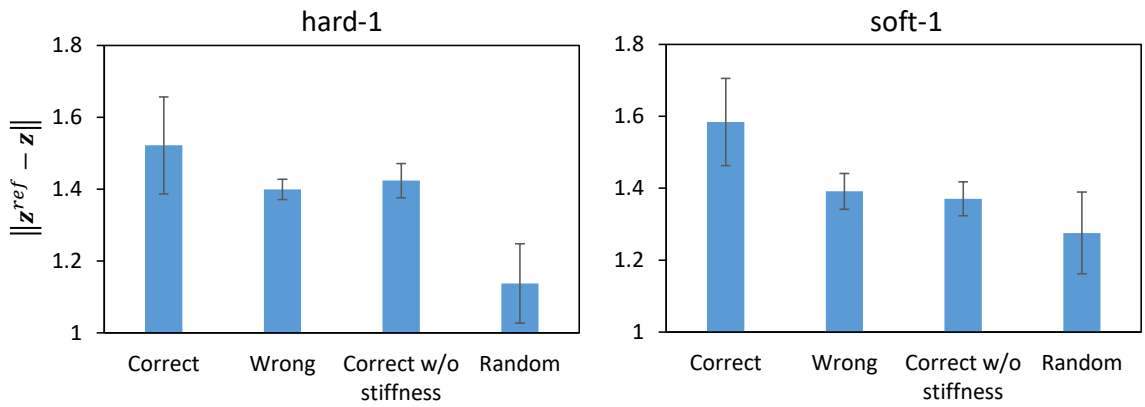


Fig. 8.52: Actual robot experiment: the maximum value of $\|z^{ref} - z\|_2$ when conducting experiments of dynamic manipulation of hard-1 and soft-1.

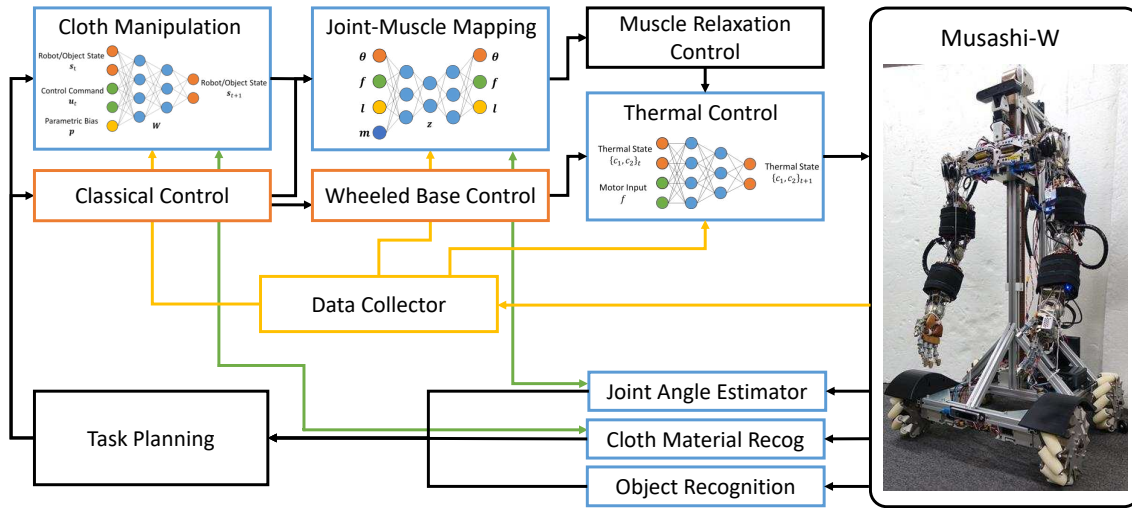


Fig. 8.53: System for the integrated table setting experiment.

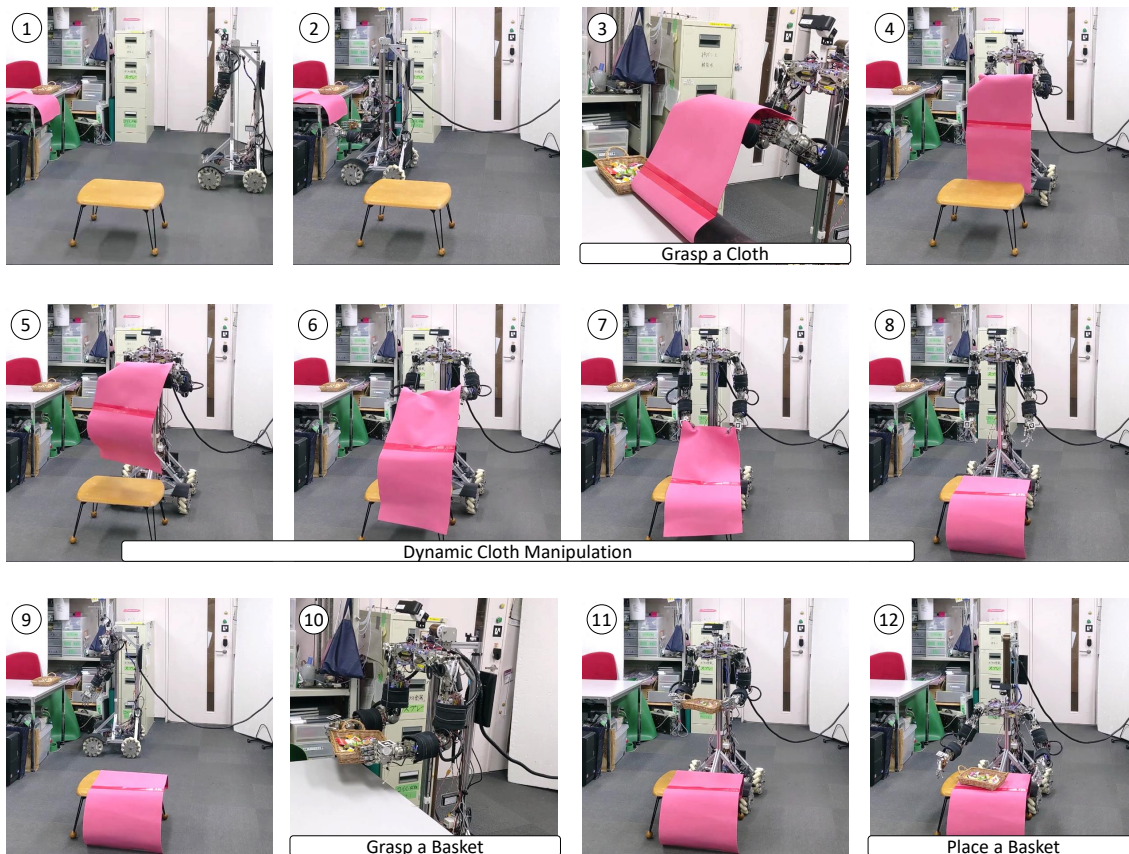


Fig. 8.54: The integrated table setting experiment of grasping a cloth, manipulating it dynamically, and putting a basket on it.

8.6 全身筋骨格ヒューマノイドのバランス制御実験

8.6.1 概要と先行研究

これまで様々な筋骨格ヒューマノイドが開発されてきたが、そのどれもが身体の柔軟性と冗長性から非常に制御が難しく、通常の軸駆動型ロボットと同じような制御を行うことは困難である。これらに対して様々な学習型制御手法が提案され [133, 129, 100, 146], 筋骨格型のロボットが自律的に自身を制御する能力を獲得できるようになってきた。一方、この中でも最も難しいのがバランス制御である。バランス制御の場合、常にバランスを保った状態でデータを取得しなければならないため、そもそもデータ取得自体が難しい。PID 等による簡単なバランス制御は行われているが [302], バランス性能が向上したとは言い難い。また、大抵ロボットや構造に応じた人間の強いパラメタライズが入っており、ロボットが自律的にバランス制御を獲得しているとは言い難い。筋骨格ヒューマノイドを除けば、これを解決する手法としては、Real2Sim [303] や Sim2Real [304] を使い、シミュレーション環境で強化学習を行う手法が開発されている。一方筋骨格ヒューマノイドのような複雑な身体はシミュレーションにおいてモデルを構築すること自体が非常に難しい。そのため、実機だけで様々なセンサ値の関係を学習することでバランス制御を獲得していくことが望ましい。そして、実機におけるデータ収集の問題点を解決する必要がある。

加えて、現在の身体状態変化に起因する直接モデル内に表現できないダイナミクス変化が多く存在するという問題がある。まず、全身のセンサ値に関するダイナミクスを学習するには膨大なデータが必要なため不可能であり、一部のダイナミクスのみを学習するために、問題が生まれる。つまり、足首の関節と筋、ZMP に関するダイナミクスを学習した場合、ここに含まれない上半身姿勢等の変化によるダイナミクス変化は考慮されない。また、筋骨格構造に特有な再現性のないキャリブレーションによる筋原点や関節原点の変化が考慮されていない。筋骨格ヒューマノイドは複雑な関節構造により基本的には関節角度を直接測定することができないため、筋長原点のキャリブレーションの再現性が低い。この他にも、履いている靴が変化したらダイナミクスは大きく変化し、これらも考慮する必要がある。これらダイナミクス変化は、外乱として扱う、または何らかの低次元パラメータとして埋め込み、これらに適応した制御を行う必要がある。本研究ではこの問題点に対して、動的な身体図式学習を利用し、Parametric Bias にこれまで説明した身体状態変化の情報を埋め込む。これをオンラインで学習・推定することで、現在の身体状態に適応しながらバランス制御を実行することが可能になる。

本研究ではこの筋骨格ヒューマノイドにおける Parametric Bias を含むバランス制御ネットワーク (Balance Controller Network with Parametric Bias, BCNPB) を使ったバランス制御を開発する。本研究

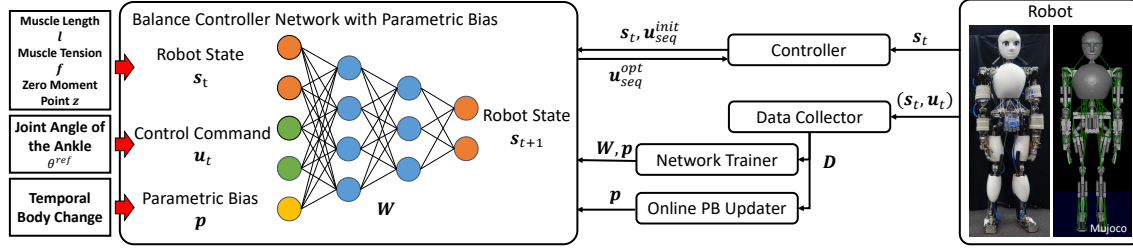


Fig. 8.55: The overall system of balance control for musculoskeletal humanoids using a deep predictive model with parametric bias.

の貢献を以下にまとめる。

- バランス制御のためのデータ収集
- Parametric Bias を使ったモデルに対する身体状態変化情報の埋め込み
- BCNPB を使った現在の身体状態への逐次適応とバランス制御

筋骨格ヒューマノイド Musashi [87] のシミュレーションと実機に本手法を適用し、その有効性を確認する。

8.6.2 動的な身体図式学習を用いた筋骨格ヒューマノイドのバランス制御

本研究における BCNPB を中心とした全体システムを Fig. 8.55 に示す。

ネットワーク構造

本研究におけるネットワーク構造を以下に示す。

$$s_{t+1} = h(s_t, u_t, p) \quad (8.23)$$

ここで、 t は現在のタイムステップ、 s はロボットの状態を表す変数、 u は制御入力、 p は Parametric Bias を表す。本研究では筋骨格ヒューマノイドにおけるバランス制御を扱う。その中でも、足首に関する関節や筋の状態を直接扱い、上半身の姿勢等は Parametric Bias によって暗黙的に扱う。そのため、 $s^T = (z^T \quad f^T \quad l^T)$ 、 $u = \theta^{ref}$ とする。ここで、 z は ZMP の値、 f は両脚の足首に関する筋張力、 l は両脚の足首に関する筋長、 θ^{ref} は足首の指令関節角度とする。なお、 z は 2 次元 (x 方向は z_x 、 y 方向は z_y とする)、 f は 12 次元、 l は 12 次元、 θ^{ref} は両脚についてロール・ピッチの角度が考えられるが、本研究では簡単のため両脚の角度を同一として、ピッチ軸のみの 1 次元とする。Parametric Bias p は暗黙的なダイナミクスの違いを埋め込むことができる変数であり、本研究では身体状態、特に上半身の姿

勢やキャリブレーション、靴等を変化させながらデータを取得することで、これらの情報が \mathbf{p} の中に自己組織化される。

本研究では BCNPB は 10 層としており、順に 4 つの全結合層、2 層の LSTM 層、4 層の全結合層からなる。ユニット数については、 $\{N_s + N_u + N_p, 200, 100, 30, 30$ (LSTM のユニット数), 30 (LSTM のユニット数), $30, 100, 200, N_s\}$ とした (なお、 $N_{\{s,u,p\}}$ は $\{s, u, p\}$ の次元数とする)。活性化関数は Tanh、更新則は Adam [46] とした。また、 \mathbf{p} は 2 次元、Eq. 8.23 の実行周期は 5 Hz としている。

データ収集

本研究はバランス制御の学習であるため、そのデータ収集の方法には工夫が必要である。単にランダムに足首を動かすだけではすぐに倒れてしまい、バランス制御に有用なデータを集めることが難しい。本研究では以下の操作を毎ステップ繰り返すことにより θ^{ref} を変化させていく。

$$c \leftarrow c + 1 \quad (8.24)$$

$$d \leftarrow d + C_{diff} \quad (8.25)$$

$$\theta^{ref} \leftarrow \theta^{ref} + \left| \sin\left(\pi \frac{c}{N_{cnt}}\right) \right| \text{Random}(-d, d) \quad (8.26)$$

$$\theta^{ref} \leftarrow \max(\theta_{min}, \min(\theta^{ref}, \theta_{max})) \quad (8.27)$$

ここで、 c はタイムカウント ($c = 0$ から始める)、 d は θ^{ref} の変位の最大値 ($d = C_{diff}^{init}$ から始める)、 $\text{Random}(a, b)$ は $[a, b]$ の範囲のランダムな値を表す。また、 $\theta_{\{min, max\}}$ は θ^{ref} の最小値最大値、 N_{cnt} 、 C_{diff} 、 C_{diff}^{init} は挙動を決める定数である。これは、 θ^{ref} の変位の最大値を徐々に増やしながらデータを収集する。初めから変位が大きいとすぐに倒れてしまい長くデータが取れないので、これは重要である。また、 θ^{ref} の変化を周期的に小さくしたり大きくしたりすることで、多様なデータを収集する。バランス制御にとって最も良い状態は安定して止まる状態なため、 θ^{ref} の変位が小さい止まった状態のデータも収集しない場合、振動的な動作が生成されてしまうことになる。

本研究では $N_{cnt} = 50$ 、 $C_{diff} = 0.002$ [rad]、 $C_{diff}^{init} = 0.1$ [rad]、 $\theta_{min} = -1.0$ [rad]、 $\theta_{max} = 1.0$ [rad] とする。

BCNPB の訓練

得られたデータ D を使い BCNPB を学習させる。この際、身体状態を変化させながらデータを収集することで、これらのデータを暗黙的に Parametric Bias に埋め込むことができる。ある同一

の身体状態 k において収集されたデータを $D_k = \{(s_1, u_1), (s_2, u_2), \dots, (s_{T_k}, u_{T_k})\}$ ($1 \leq k \leq K$, K は全試行回数, T_k はその身体状態 k における試行の動作ステップ数) として, 学習に用いるデータ $D_{train} = \{(D_1, p_1), (D_2, p_2), \dots, (D_K, p_K)\}$ を得る. ここで, p_k はその身体状態 k におけるダイナミクスを表現する Parametric Bias であり, その状態については共通の変数, 別の状態については別の変数となる. この D_{train} を使って BCNPB を学習させる. 通常の学習ではネットワークの重み W のみが更新されるが, ここでは W と各状態に関する p_k が同時に更新される. これにより, p_k にそれぞれの身体状態におけるダイナミクスの違いが埋め込まれることになる. 学習の際は損失関数として平均二乗誤差を使い, p_k は全て $\mathbf{0}$ を初期値として最適化される.

Parametric Bias のオンライン学習

現在の身体状態において得られたデータ D を使い, オンラインで Parametric Bias を更新する. ネットワークの重み W を更新してしまうと BCNPB がそのデータに過学習してしまう可能性があるが, 低次元の Parametric Bias p のみを更新するのであれば過学習は起こらない. このオンライン学習により, 常に現在の身体状態に適応したモデルを得ることができる.

得られたデータ数を N_{data}^{online} として, データ数が N_{thre}^{online} を超えたところからオンライン学習を始める. 新しいデータが入るたびにバッチ数を N_{batch}^{online} , エポック数を N_{epoch}^{online} , 更新則を MomentumSGD として学習を行う. N_{max}^{online} を超えたデータは古いものから削除していく.

本研究では, $N_{\{thre, max\}}^{online} = 50$, $N_{batch}^{online} = N_{max}^{online}$, $N_{epoch}^{online} = 1$ とした.

BCNPB によるバランス制御

BCNPB を使った制御手法について述べる. ここでは, s と u に関する損失関数から, u を最適化していくことを考える. まず, $u_{seq} = u_{[t:t+N_{step}-1]}$ の初期値 u_{seq}^{init} を与える. 最適化する u を u_{seq}^{opt} とおき, 時刻 t において以下の計算を繰り返すことで最適な u_t^{opt} を得る.

$$s_{seq}^{pred} = h_{expand}(s_t, u_{seq}^{opt}) \quad (8.28)$$

$$L = h_{loss}(s_{seq}^{pred}, u_{seq}^{opt}) \quad (8.29)$$

$$u_{seq}^{opt} \leftarrow u_{seq}^{opt} - \gamma \partial L / \partial u_{seq}^{opt} \quad (8.30)$$

ここで, s_{seq}^{pred} は予測された $s_{[t+1:t+N_{step}]}$, h_{expand} は h を N_{step} 回展開した関数, h_{loss} は損失関数, γ は学習率を表す. つまり, s_t から u_{seq}^{opt} により将来の s を予測し, これに対して設定した損失関数を最小化するように, u_{seq}^{opt} を誤差逆伝播法と勾配法により最適化する.

このとき \mathbf{u}_{seq}^{init} は、前ステップで最適化された \mathbf{u}_{seq} である $\mathbf{u}_{[t:t+N_{step}-1]}^{prev}$ を使い、時間を一つシフトし最後の項を複製した $\mathbf{u}_{\{t+1, \dots, t+N_{step}-1, t+N_{step}-1\}}^{prev}$ とする。前回の最適化結果を利用することでより速い収束が得られる。また、 γ については $[0, \gamma_{max}]$ を指数関数的に分割した $N_{batch}^{control}$ 個の γ を用意し、それぞれの γ で Eq. 8.30 を実行した後、Eq. 8.29 で損失を計算し最も損失の小さい \mathbf{u}_{seq}^{opt} を選択することを $N_{epoch}^{control}$ 回繰り返す。

ここで、損失関数 \mathbf{h}_{loss} について考える。本研究では \mathbf{h}_{loss} を以下のように設定する。

$$\begin{aligned} \mathbf{h}_{loss}(\mathbf{s}_{seq}^{pred}, \mathbf{u}_{seq}^{opt}) = & \|\mathbf{z}_{seq}^{pred} - \mathbf{z}_{seq}^{ref}\|_2 \\ & + C_f \|\mathbf{f}_{[3:N_{step}]}^{pred} - \mathbf{f}_{[2:N_{step}-1]}^{pred}\|_2 \\ & + C_l \|\mathbf{l}_{[3:N_{step}]}^{pred} - \mathbf{l}_{[2:N_{step}-1]}^{pred}\|_2 \\ & + C_u \|\mathbf{u}_{seq}^{opt}\|_2 \end{aligned} \quad (8.31)$$

ここで、 $\{\mathbf{z}, \mathbf{f}, \mathbf{l}\}_{seq}^{pred}$ は \mathbf{s}_{seq}^{pred} における $\{\mathbf{z}, \mathbf{f}, \mathbf{l}\}$ の値、 \mathbf{z}_{seq}^{ref} は \mathbf{z} の指令値を N_{step} 個並べた値、 $C_{\{f, l, u\}}$ はそれぞれの損失の重みの定数を表す。つまり、 \mathbf{z} に関する指令値の実現、 \mathbf{f} の変化の最小化、 \mathbf{l} の変化の最小化、 \mathbf{u} の最小化をまとめた損失となっている。なお、 $C_{\{f, l, u\}}$ は実験ごとに变化させる。

本研究では、 $N_{step} = 6$, $N_{batch}^{control} = 10$, $N_{epoch}^{control} = 3$, $\gamma_{max} = 0.1$ とする。

8.6.3 実験

実験セットアップ

本研究では筋骨格ヒューマノイド Musashi [87] を用いて実験を行う。本研究では基本的に足首のピッチ関節のみを動かし、一部上半身を動かす以外については、直立した姿勢でバランス制御を行う。ZMP については、足平に分布する 12 個のロードセルから計算している。また、筋骨格ヒューマノイドにおいては指令関節角度を筋長に変換する必要があるが、本研究では [125] を使い、指令筋張力を 100 [N] で一定として変換している。なお、[125] の学習は完璧に指令関節角度を実現できるとは限らず、筋の摩擦等によって誤差が生じる。シミュレーションについては Mujoco [257] を使う。また、 \mathbf{p} は 2 次元とした。

シミュレーション実験

シミュレーション実験を行う。ここでは身体状態変化として、腰関節のピッチ軸角度 θ_{s-p} とキャリブレーションのズレを表現した足首ピッチ関節角度のオフセット θ_{a-p}^{offset} を扱う。まずデータ収集

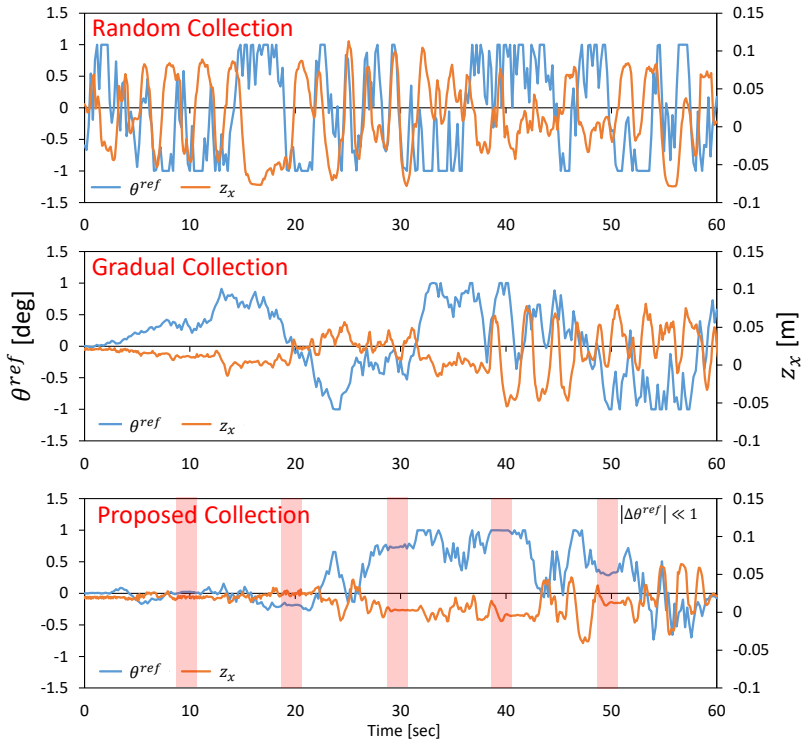


Fig. 8.56: Simulation experiment: the transition of θ^{ref} and z_x when conducting Random, Gradual, and Proposed Collection.

を行うが、この際身体状態を $\theta_{s-p} = \{-5.0, 0.0, 5.0\}$ [deg], $\theta_{a-p}^{offset} = \{-5.0, 0.0, 5.0\}$ [deg] によって 9 種類に変化させながらデータを取る。それぞれの身体状態について、300 ステップのデータを得る。ここで、Eq. 8.26 によるデータ収集を Proposed Collection, $|\sin(\pi \frac{c}{N_{cnt}})|$ の項を消した場合を Gradual Collection, Gradual Collection において $d = 1.0$ と固定した場合を Random Collection と呼ぶ。この際の z_x と θ^{ref} の遷移を Fig. 8.56 に示す。Random Collection では常に z_x と θ^{ref} が大きく変化する。Gradual Collection では z_x と θ^{ref} の変化幅は徐々に大きくなっていく。一方 Proposed Collection では Gradual Collection の特徴に加えて、 θ^{ref} が激しく動くところとゆったり動くところが交互に訪れており、多様なデータが収集されている。本データから動的な身体図式のネットワーク構造の自動決定を行った結果を Fig. 8.57 に示す。Eq. 8.23 のような運動方程式型のネットワークが自動で構築された。

この 2700 ステップのデータを使って BCNPB を学習させる。それぞれのデータ取得方法について、得られた Parametric Bias に Principle Component Analysis (PCA) をかけ、2 次元平面上にプロットしたものを Fig. 8.58 に示す。Proposed Collection を見ると、直接身体状態に関するパラメータはデータとして与えていないが、身体状態パラメータの大小に従って値が自己組織化していることがわかる。

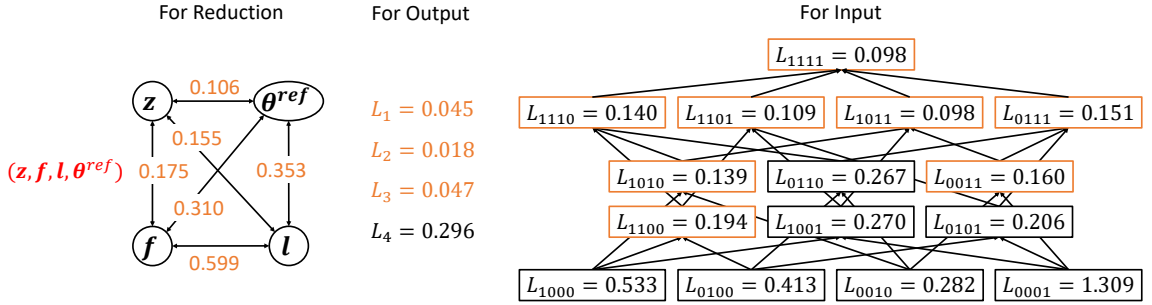


Fig. 8.57: The result of automatic network input output decision.

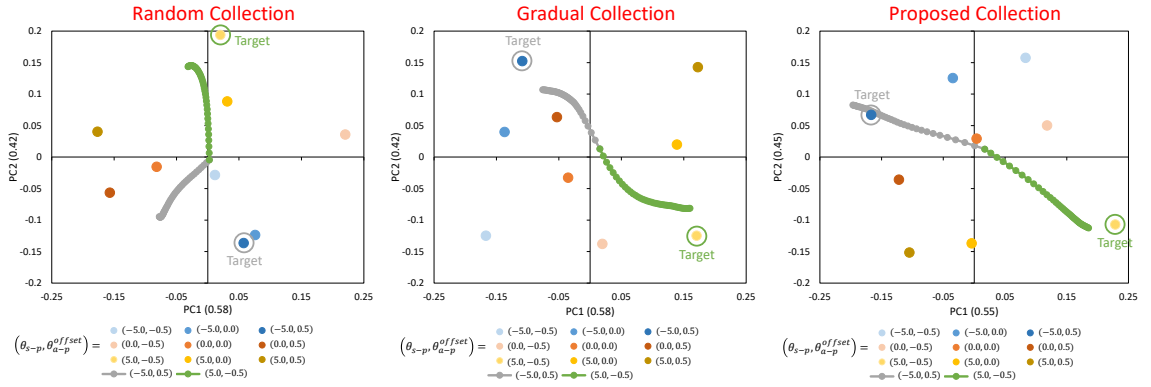


Fig. 8.58: Simulation experiment: the arrangement of parametric biases when training BCNPB using the data collected with Random, Gradual, and Proposed Collection, and the trajectories of parametric biases when running online learning by setting $(\theta_{s-p}, \theta_{a-p}^{offset}) = \{(-5.0, 0.5), (5.0, -0.5)\}$.

つまり、実機の再キャリブレーションのような直接身体状態のパラメータが得られないような場合にも、これらを Parametric Bias の空間に構造化することが可能である。一方で、Gradual Collection は Proposed Collection に比べて PB の空間が歪んでいることがわかる。また、Random Collection については Gradual Collection よりも更に PB の空間が歪んでいる。

次に、Parametric Bias のオンライン更新について実験を行う。 $p = \mathbf{0}$ の状態から始め、データ収集時と同様に体を動かすと同時に PB のオンライン更新を実行したときに、どのように p が遷移するかを調べる。 $(\theta_{s-p}, \theta_{a-p}^{offset}) = \{(-5.0, 0.5), (5.0, -0.5)\}$ としたときの p の軌跡を Fig. 8.58 に示す。なお、オンライン学習 45 ステップ分の軌跡を表示している。徐々に現在の p が、現在と同じ身体状態において学習された p へと近づいていくことがわかる。つまり、 p の空間を探索して身体状態を正しく認識するような適応が可能である。また、その精度は Random < Gradual < Proposed Collection の順で高くなる。

最後に、制御について実験を行う。本実験では全て $(\theta_{s-p}, \theta_{a-p}^{offset}) = (0.0, 0.0)$ とし、腰リンクに対

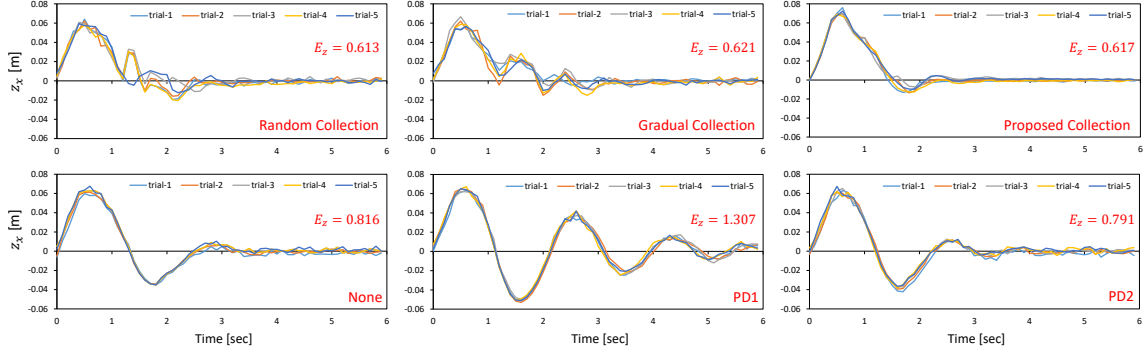


Fig. 8.59: Simulation experiment: the transitions of z_x after applying 30 N force to the chest link for 0.2 seconds, when the balance control using BCNPB trained with the data collected by Random, Gradual, or Proposed Collection is performed, when no balance control is performed (None), or when simple PD controls with different gains are performed (PD1, PD2).

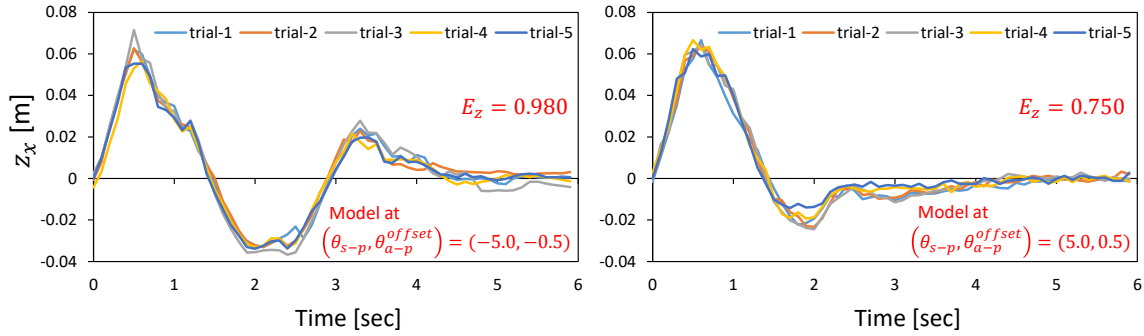


Fig. 8.60: Simulation experiment: the transitions of z_x after applying 30 N force to the chest link for 0.2 seconds, when running the balance control using BCNPB trained with the data collected by Proposed Collection and parametric bias trained at $(\theta_{s-p}, \theta_{a-p}^{offset}) = \{(-5.0, -0.5), (5.0, 0.5)\}$.

して 30N の外力を 0.2 秒間加えた後の z_x の遷移を 6 秒間、5 回ずつ検証する。 z_x についてはグラフを揃えるためにオフセットを取り除き、6 秒間 (30 ステップ分) の $|z_x|$ の合計値の平均を E_z として表示している。 また記載がない限り、制御については $(C_f, C_l, C_u) = (0, 30, 3)$ と設定し、PB については $(\theta_{s-p}, \theta_{a-p}^{offset}) = (0.0, 0.0)$ の際に得られた値とする。

まずは、Random, Gradual, Proposed Collection において得られたそれぞれのモデルを使ってバランス制御をした場合、何も制御しなかった場合 (None), PD 制御をした場合 (PD) の結果を Fig. 8.59 に示す。 PD についてはどのような設定をしても z_x の収束を妨げる方向に働いてしまったが、例として $(K_P, K_D) = (0.1, 0.1)$ (PD1), $(K_P, K_D) = (0.03, 0.1)$ (PD2) の場合を表示している。 Random Collection と Gradual Collection のデータを使ったモデルは Proposed Collection のデータを使った場合に比べて、 E_f

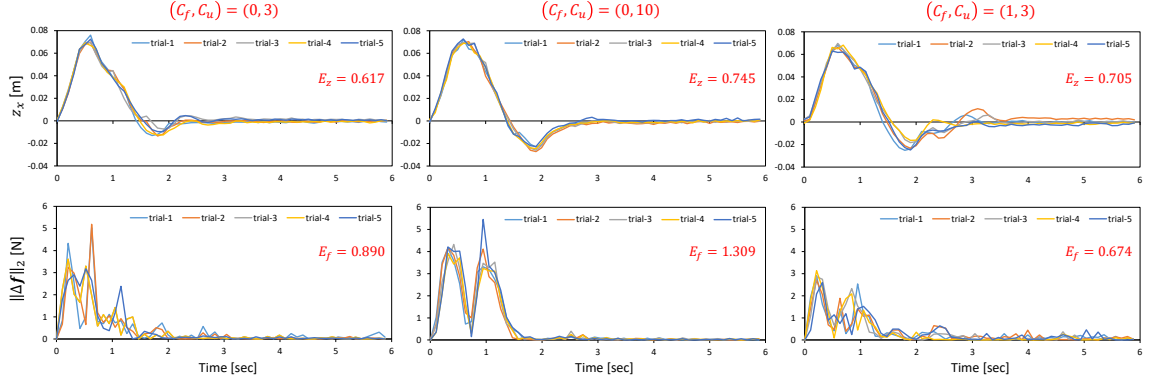


Fig. 8.61: Simulation experiment: the transitions of z_x and $\|\Delta f\|_2$ after applying 30 N force to the chest link for 0.2 seconds when running the proposed balance control with $(C_f, C_u) = \{(0, 3), (0, 10), (1, 3)\}$.

は大きく変わらないものの、 z_x が振動的になっていることがわかる。None については本研究の制御を使った場合に比べて E_z が大きく、 x_z が一度プラス方向に振れた後、大きくマイナス方向にもう一度振れている。一方で本研究の制御を使った場合はマイナス方向にはあまり振れず、そのまま収束することがわかる。PD の場合はゲインを変更しても None よりも悪い方向になる場合がほとんどであった。

次に、Proposed Collection において得られたモデルを使った場合について、PB を $(\theta_{s-p}, \theta_{a-p}^{offset}) = \{(-5.0, -0.5), (5.0, 0.5)\}$ の際に得られた値とした場合の結果を Fig. 8.60 に示す。どちらについても、 $(\theta_{s-p}, \theta_{a-p}^{offset}) = (0.0, 0.0)$ の際に得られた正しい PB を使った場合に比べて大きく誤差がのってしまうことがわかる。

最後に、Proposed Collection において得られたモデルを使った場合について、 $C_l = 30$ は固定し、 $(C_f, C_u) = \{(0, 3), (1, 3), (0, 10)\}$ のようにバランス制御のパラメータを変化させた際の結果を Fig. 8.61 に示す。なお、ここでは筋張力の前ステップからの時間変化のノルム $\|\Delta f\|_2$ の遷移についても表示しており、その値の二乗平均平方根を E_f とする。また、Fig. 8.61 の左上図は Fig. 8.59 の Proposed Collection と同じグラフである。 C_u を 3 から 10 に変更すると、 $\mathbf{u} = \theta^{ref}$ の動きが抑制されるため、 z_x の動きが Fig. 8.59 の None に近づくことがわかる。また、 C_f を 0 から 1 に変更すると、 $\|\Delta f\|_2$ のピークが収まり、 E_f も 0.674 と $C_f = 0$ の場合に比べると小さな値となっていることがわかる。

8.6.4 実機実験

筋骨格ヒューマノイド Musashi を使った実機実験を行う。ここでは身体状態変化として、Fig. 8.62 のうちのどの靴を使うか {Bare, White, Pink, Navy}、また、上半身の姿勢 {M, Z, P, U} を扱う (ここで、M は $\theta_{s-p} = -5$ 、Z は $\theta_{s-p} = 0$ 、P は $\theta_{s-p} = 5$ 、U は両腕の肘の関節角度 $\theta_{e-p} = -60$ [deg] を表す)。まずデー

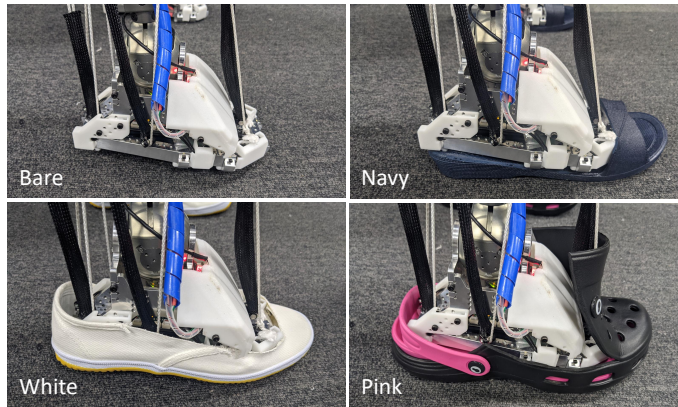


Fig. 8.62: Shoes as temporal body changes for the actual robot experiment.

タ収集を行うが、この際の身体状態について、靴を {Bare, Pink, Navy}, 上半身姿勢を {M, Z, P, U} として 12 種類に変化させながらデータを取る。それぞれの身体状態について、300 ステップのデータを得る。ここでは Eq. 8.26 によるデータ収集のみを行い、この方法で習得した制御を Proposed, 何も制御しない場合を None と表記する。これらのデータを使って BCNPB を学習した際に得られた Parametric Bias に PCA をかけ、2 次元平面上にプロットしたものを Fig. 8.63 に示す。シミュレーション実験ほど綺麗ではないが、Navy, Pink, Bare ごとに大まかに PB が構造化されていることがわかる。また、上半身姿勢の P と U は姿勢こそ違うものの前傾するという意味では似たダイナミクスを持っており、P と U は比較的近い位置に PB 同士が存在することもわかる。本モデルについては、シミュレーションで得られた BCNPB からの Fine Tuning ではモデルが大きく異なるため損失があまり下がらなかった。

次に、Parametric Bias のオンライン更新について実験を行う。 p を Bare-U の状態から始め、データ収集時と同様に体を動かすと同時に PB のオンライン更新を実行したときに、どのように p が遷移するかを調べる。現在の身体状態を Pink-U, Navy-Z, White-Z としたときの p の軌跡を Fig. 8.63 に示す。Pink-U と Navy-Z については、現在の p が、徐々に現在と同じ身体状態において学習された p へと近づいていくことがわかる。つまり、 p の空間を探索して身体状態を正しく認識するような適応が可能である。また、White については学習時のデータには入っていないが、これは学習結果として Bare の近く上方あたりに配置された。なお、靴には形や摩擦、柔らかさなどの様々なパラメータがあるが、White と Bare, Pink と Navy はそれぞれ靴底の硬さが似通っている。

最後に、本研究の制御 Proposed と何も制御しない None におけるバランス制御の結果を Fig. 8.64 に示す。本実験では全て上半身の姿勢は Z とし、腰リンクに対してある一定の力 (Bare については 15 N, Pink については 10 N) をかけてからこれを離れた後の z_x の遷移を 6 秒間、5 回ずつ検証する。 z_x

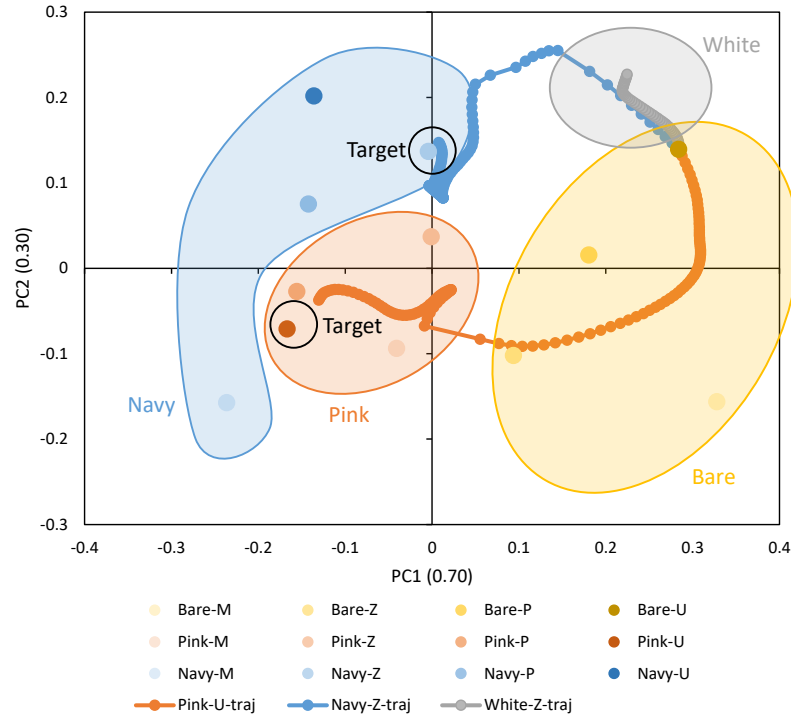


Fig. 8.63: Actual robot experiment: the arrangement of parametric biases when training BCNPB using the data collected with Proposed Collection, and the trajectories of parametric biases when running online learning by setting Pink-U, Navy-Z, or White-Z.

についてはグラフを揃えるためにオフセットを取り除き、6 秒間 (30 ステップ分) の $|z_x|$ の合計値の平均を E_z として表示している。PB についてはそれぞれの身体状態 (Bare-Z または Pink-Z) について学習時に得られた値を利用する。また、Bare-Z については $(C_f, C_l, C_u) = (0, 30, 3)$, Pink-Z については $(C_f, C_l, C_u) = (0, 3, 1)$ とした。シミュレーションほど効果は大きくないが、外力が加わった後の収束は、None に比べて Proposed の方が速いことが見て取れる。実際、Bare については、None では $E_z = 0.349$, Proposed では $E_z = 0.296$, また、Pink については、None では $E_z = 0.322$, Proposed では $E_z = 0.246$ となり、Proposed の方が誤差が少ない。

8.6.5 議論

本研究の実験結果について考察する。まずシミュレーション結果から、BCNPB の学習により明示的に値として与えていないダイナミクスのパラメータが Parametric Bias に埋め込まれることがわかった。この PB の配置はデータが多様な時系列変化を持つほど綺麗に自己組織化し、オンラインで現在のダイナミクスを学習し PB を正確に更新することができるようになる。また、多様な時系列変化を学

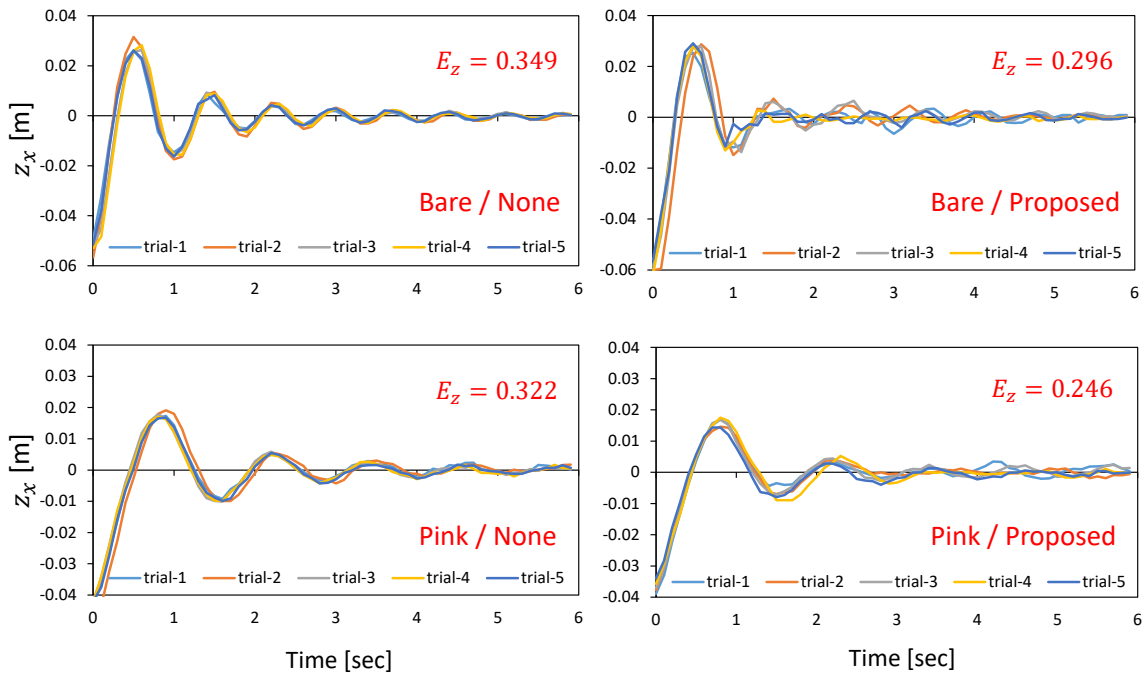


Fig. 8.64: Actual robot experiment: the transitions of z_x when 15 N (for Bare) or 10 N (for Pink) of external force is applied to the chest link and released, when the balance control using BCNPB is performed (Proposed), or when no control is performed (None).

習することでよりバランス制御が正確になり、外力に対する応答の収束が速くなることがわかる。制御をしない場合や PID 制御をする場合には、収束が遅かったり逆に発散したりしてしまう場合があるが、本手法により外力をいなし即座に安定して直立することが可能となった。一方で、PB が現在の身体状態に適合したものではない場合、現実のダイナミクスと予測したダイナミクスが異なるため正確にバランス制御が働かない場合がある。本バランス制御は損失関数における重みを変化させることで、制御入力を小さくしたり、筋長や筋張力の変化を抑えたりと、別の目的を同時に実行することが可能である。

次に実機実験では、靴という人間が明示的に与えることの難しいダイナミクスの違いを扱った。学習された PB は靴の種類ごとにまとまりを示し、また現在の動きから現在履いている靴の種類を当て、そのダイナミクスを理解することができる。学習時に用いていない靴にも汎化するようなダイナミクスの近さや遠さが反映された PB の空間が構築されていた。また、肘や腰の角度のような上半身の姿勢についても、PB の中では下半身のダイナミクス変化という形で同じ変数で統一的に扱うことが可能であった。バランス制御についてはシミュレーションほどではないもののある程度の性能を示し、制御を入れない場合よりも速い誤差の収束が可能であった。

本研究の限界について述べる。まず、現状は誤差逆伝播の速度が律速となり、速い周期で制御を実行できないという問題がある。15 Hz 程度が限界であるが、本研究ではその程度では 5 Hz と結果は大差なかった。今後 100 Hz 程度まで周期を上げることができれば、より適応範囲が拡大されと考える。

次に、データ収集の問題が挙げられる。本研究では徐々に大きく体を揺らすことで多様なデータを収集したが、よりダイナミックなデータを得るためにはカリキュラム学習のように学習とデータ収集を交互に行うような、さらなるデータ収集の工夫が必要である。今後は、手すりを使いながら足踏みを学習し、最終的に手を離して歩くような学習方法の開発が望まれる。

8.7 本章のまとめ

まず、本研究における身体図式の逐次学習機能を有する智能ロボットシステムの全貌についてまとめた。また、身体-道具-動作環境の相関複雑性と時間的変容の攻略についてまとめ、身体図式学習による環境適応能力の向上について考察した。

次に、筋骨格ヒューマノイドのモデル化困難ではあるが柔軟な身体の特性を活かし、人間の教示によって動作を変化させる手法について考察した。軸駆動型とは異なる、関節角度が測定できず、筋張力の影響がアクチュエータでなく筋のハードウェア弾性側に大きく伝播される特性を考慮した、筋補償制御による動作再生手法を開発した。静的身体図式である Musculoskeletal AutoEncoder から得られる情報を使い、ハードウェア弾性を推定し、正確に修正された動作を再生することができることを実験を通して確認した。

次に、筋骨格ヒューマノイドによる自動運転に焦点を当て、そのハードウェアとソフトウェアの特性を生かした様々な運転操作を行った。柔軟かつ、その柔軟さを変化させることができる点、センシングに優れる点を活かし、両手による運転やサイドミラー認識等も成功させている。また、静的・動的な身体図式学習を導入し、それをもとに認識を含むペダル操作やハンドル操作を可能とした。

次に、低剛性な樹脂製ロボットにおける道具利用について、その道具先端位置や重心、関節角度に関する相互関係の学習、道具の把持による学習した相互作用の変化を考慮可能な身体図式学習によるモデル化手法を開発した。ネットワークの入出力で表現可能な損失関数を設定し、これにより意図した通りの動作制御が可能となる。また、現在の動作データから道具の長さや重さを認識し、道具の逐次的な変化にも対応することが可能となった。

次に、より人間らしい動的布操作を目指し、布の素材変化への対応・可変剛性を使った素早いマニピュレーションを取り入れた動的な身体図式学習手法を開発した。布の素材変化に対しては Parametric Bias を、可変剛性については関節角度と剛性値を制御入力として誤差逆伝播を用いて指令値を計算する手法を取り入れた。訓練データにはない布素材についてもそのダイナミクスが推定でき、また、素材によって動的布操作の特性が大きく変わることを示した。また、動作フェーズに応じて適切に剛性値が設定されることで、可変剛性制御を行わない場合に比べて速度を 12% 程度向上させ、より正確に指令状態を実現することが可能であった。

最後に、柔軟性と冗長性を備える複雑な筋骨格ヒューマノイドのバランス制御に向けた動的な身体図式学習の提案を行った。バランス制御というデータ収集の困難なタスクに対して、ランダムな制御入力の変位を徐々に増やし、かつそのランダム幅を周期的に変化させながらデータを収集することで、安定

した有用なバランス制御を構築することができる. また, 学習した足首に関するダイナミクスに含まれないような, 上半身の姿勢や履いている靴等については暗黙的なダイナミクス変化として **Parametric Bias** に埋め込む. 提案した身体図式モデルにより, 身体状態の変化に適応しつつ, 様々な損失関数に従って筋骨格ヒューマノイドがバランス制御を行うことに成功した.

第9章

結論

9.1 本研究の総括

本研究ではロボットの身体-道具-動作環境間の相関複雑性と時間的変容に対応する、身体図式の逐次学習機能を有する知能ロボットシステムの開発を行った。全てのロボットはその柔軟性と冗長性から、モデル化困難性と逐次的モデル変化を持つ。これに対応するために、ロボットの身体図式をモデル化し、モデル化困難性と逐次的モデル変化を考慮可能な一般化多感覚相関モデルを構築した。また、この身体図式の逐次学習に向けて、柔軟性と冗長性を備えるロボットのハードウェア設計、柔軟性と冗長性の欠点を支える反射制御が必要となる。これらにより、身体-道具-動作環境の相関複雑性と時間的変容を攻略した環境適応能力の高い知能ロボットシステムの開発に成功した。

柔軟性と冗長性を持つロボットの知能システム開発

本研究では知能ロボットシステムを、人間のように自律的に経験から学習を行うシステムと定義し、この自律性により環境適応能力の向上が可能なシステム開発を目指した。全てのロボットは柔軟性と冗長性を持ち、これによりモデル化困難性と逐次的モデル変化が生まれる。この柔軟性と冗長性は、時間的な柔軟性と空間的な柔軟性、制御的な冗長性と感覚的な冗長性という性質によりまとめることが可能である。それぞれの特徴に利点と欠点が存在する。時間的柔軟性は、成長や身体変化を扱うことができる一方、経年劣化や逐次的な状態変化への対応が必要になる。空間的柔軟性は、環境との柔軟接触や力の蓄積が可能である一方、再現性の低下やモデル化困難性等の問題を持つ。制御的冗長性は、可変剛性制御や冗長駆動を可能にする一方、制御的競合や内力上昇の原因となる。感覚的冗長性は、多数の感覚による学習性能向上や異常検知を可能にする一方、モデル化は困難である。そこで本研究では、これらの利点を増強するハードウェア構成、欠点を補完する反射制御、利点を増強し欠点を補う身体図式学習手法を開発した。

本研究で提案する知能ロボットシステムはこれまでの知能ロボットシステムの要素と競合するものではない。主に認識・動作計画・タスクごとにおける身体制御・反射が存在し、このうち認識やタスクごとにおける身体制御を、身体図式学習により一般化したものと捉えることができる。既存のコンポーネントを併用しつつ、モデル化困難性と逐次的モデル変化によりこれまで困難だったコンポーネントを生成し置き換えることができる。

身体図式学習に向けたハードウェア構成

軸駆動型や台車型のロボットは既存の身体構成法が確立しているため、主に筋骨格型ロボットに関してハードウェア構成法を述べた。時間的柔軟性の利点を活かすためには、容易に身体構造を変化可能な身体のモジュール化が重要である。そのため、関節モジュール・筋モジュール・汎用骨格・筋経由点ユニットからなる容易に身体を構成・再構成可能な設計開発を行った。空間的柔軟性の利点を活かすためには、環境接触が可能な柔軟身体構造が必要であり、ユニット自体が柔軟な非線形弾性ユニットの開発を行った。制御的冗長性の利点を活かすためには、より多くの筋をコンパクトに身体に配置することが重要である。そのため、回路やアクチュエータ、筋張力測定ユニットの統合によるコンパクトな筋モジュール作成や、骨と筋を統合した骨構造一体小型筋モジュールの開発を行った。感覚的冗長性の利点を活かすためには、より多くのセンサを搭載した身体が重要である。そこで、関節角度を測定可能な擬似球関節モジュール、冗長な接触センサを有する柔軟ハンド、輻輳による物体距離認識機能と高解像度カメラを有した可動眼球ユニット、足全周の触覚を持ったコア-シェル型足ユニット、人体模倣構造により三次元音源定位が可能な両耳ユニット、臀部と環境の柔軟な接触力を測定可能な臀部触覚センサの設計開発を行った。この時間的柔軟性と感覚的冗長性の利点を増強することは、学習制御プラットフォームの要件である、多種感覚を使った学習評価機構、信頼性のあるハードウェア、身体構造変化の考察機構の実現にも繋がる。これらのコンポーネントの組み合わせにより、筋骨格ヒューマノイド MusashiLarm, Musashi, MusashiOLegs, TWIMP, Musashi-W, Kengoro を開発した。また、筋破断時にも動き続けることが可能な冗長駆動の利点を最大限活用可能な筋配置の最適化アルゴリズムも開発した。

身体図式学習を支える反射制御

空間的柔軟性と制御的冗長性によって発生するモデル化困難性とこれに伴う制御的競合・内力上昇・ヒステリシスという欠点にいち早く対処するためには、高周期で実行される反射制御が重要である。このために、モータ温度制御、筋弛緩制御、速度最大化反射戦略、拮抗筋抑制制御、伸長反射制御を開発した。このうち、モータ温度制御は様々なロボットに適用できるものの、それ以外については制御的競合により問題を起こしやすい筋骨格型に向けた開発された。モータ温度制御はモータハウジング温度・モータコア温度・入力電流の関係からモータコア温度を推定し、これを定格内に収める逐次学習型の反射制御である。避けられない内力上昇によるモータ温度上昇を抑え、長期的で信頼性のある動作を可能にする。筋弛緩制御は動作が停止しているときに有効であり、関節角度を変化させな

い中で、無駄な拮抗筋を徐々に弛緩させる制御である。モデル化誤差による内力上昇を抑え、長期的な動作を可能にする。速度最大化反射戦略は動作時に拮抗筋の電流をゼロとする拮抗筋抑制反射、動作前に予めモーメントアームの大きな筋を伸長させる拮抗筋予見伸長反射の2つに大別される。最も速度の遅い筋に最大関節速度が規定されるという制御的競合を解決し、より素早い動きを可能にする。拮抗筋抑制制御は拮抗筋の筋追従を抑制する反射であり、人間の相反性神経支配という反射を模倣した制御である。モデル化誤差による拮抗筋の動作制限を抑制し、広可動域動作を実現可能にする。伸長反射制御は筋の突発的な伸長に対して筋を収縮させる反射であり、人間の伸長反射を模倣した制御である。重量物把持等の際の、ヒステリシスによる再現性のない関節動作を抑制するとともに、危険回避や姿勢安定化に寄与する。

身体図式の逐次学習法とその応用

柔軟性と冗長性の利点を増強し、欠点を補う身体図式学習法を開発し、多種多様な感覚間の相関関係を潜在変数を通して統合的に記述する一般化多感覚相関モデルを提案した。このモデルにより、モデル化困難な身体図式を実機データのみから、ネットワークの入出力等も含めて自律的に獲得することができる。また、オンライン学習と Parametric Bias の更新に基づく、経年劣化や身体状態変化等による逐次的モデル変化への適応が可能になる。順伝播と逆伝播の繰り返し計算により、多様な指令状態実現の制御、身体状態推定、異常検知、シミュレーションが可能になる。また、入出力への異常検知マスク導入によるネットワーク入力への減少対応、重みコピーと再学習を利用したネットワーク入力への増加対応、グラフ分割アルゴリズムによる身体図式モデルの身体部位間自動分割、ネットワークへの確率表現と平均分散表現の導入、時系列展開型のネットワークによる身体図式学習の高速動作対応、明示的パラメータによる学習安定化等の応用を行った。また、最終実験として、筋骨格ヒューマノイドによる運転動作、低剛性樹脂製ヒューマノイドによる道具操作、台車型筋骨格ヒューマノイドによる柔軟物体操作を実現した。

環境適応能力の向上

開発した身体図式学習は、ロボットの身体-道具-動作環境の相関複雑性と時間的変容に対応することができる。ロボット身体にはモータのようなモデル化容易で小さなシステムから、軸駆動型、台車型、低剛性樹脂製ロボット、筋骨格型、柔軟ハンド等、モデル化困難性の異なる様々な形態が存在する。入出力を一般化した一般化多感覚相関モデルにより、これらの身体図式を実機データのみから学習し、

制御や状態推定, 異常検知等に利用することが可能になる. 同様に, 身体と環境の相互作用を考えた制御や, 柔軟な道具・柔軟な対象物体の操作も可能になることがわかった. また, 身体-道具-動作環境の間には様々な時間的変容がつきまとう. 身体には経年劣化や成長, 再初期化やアクチュエータの特性変化等様々な状態変化が, 道具にはそのプロパティの変化が起こりうる. 環境には対象物の位置変化や障害物変化, 床摩擦変化等が起こり, 身体と道具の間の位置関係変化も避けられない. これらの時間的変容はその時間的性質から, 突発的变化, 短時間変化, 中時間変化, 長時間変化に分けることができる. このうち, 突発的变化にはネットワークの構造変化対応とネットワーク重みの再学習により対応することができる. また, 短時間変化には Parametric Bias の更新, 中時間変化にはネットワーク重みのオンライン学習, 長時間変化には明示的モデル化による安定したパラメータの逐次更新が重要となる. これらから, 筋経路変化適応, 把持物体変化適応, 身体-道具位置関係変化適応, 対象物体位置変化適応, 筋破断・筋増加による身体変化適応等が可能となった.

9.2 結論

全てのロボットは時間的・空間的柔軟性と, 制御的・感覚的冗長性を持ち, ゆえにその身体-道具-動作環境間に相関複雑性と時間的変容という特徴を持つ. この問題を解決するには, ハードウェアにおいて, 小型化・モジュラー化・非線形弾性要素の追加により, 身体の柔軟性と冗長性の利点を増強する必要がある. 反射制御において, 人間の反射機構, 特にその主動筋拮抗筋関係に基づき, 短い周期で身体の内力増加やヒステリシス, 温度上昇を解消することで, 柔軟性と冗長性の欠点を補完する必要がある. 身体図式学習において, 様々なロボットやタスクにおける多様な感覚間の相関をマスク変数により表現する (多感覚性) 予測モデル型の身体図式を, 入出力構造も含めて自律的に学習し (自律獲得性), 誤差逆伝播と勾配法をもとに状態推定や制御, 異常検知やシミュレーション等を行い (汎用性), そして, この身体図式をモデルの逐次的変化に応じて常に更新することで (変化適応性), 長い周期で柔軟性と冗長性の利点を増強, 欠点を補完する必要がある. このハードウェア・反射制御・身体図式の逐次学習機能を有する知能ロボットシステムにより, 軸駆動型や台車型, 低剛性樹脂製や筋骨格型ヒューマノイドの環境適応能力が飛躍的に向上することが明らかになった.

9.3 今後の展望

本研究の今後の展望として, 身体-道具-環境の最適設計, 身体図式のタスク間自己組織化, 今後のロボットの在り方について述べる.

まず、本研究は身体-道具-環境の相互作用から、これらを自在に操り、その変化に適応するシステム構築を行った。一方、現在の状態に順応するだけでなく、身体や道具、環境そのものを目的に従って最適化することも考えられる。目的に適した道具や身体的设计、現状の身体に適した環境の作成等、身体-道具-環境を目的に対して微分可能な表現とすることで、これらを成し遂げる新しいシステム構築が可能なはずである。本研究との併用により、身体-道具-環境を自身で変化させつつ、その変化に順応するような、道具や環境を身体の一部かのように扱う知能ロボットシステム構築は興味深い。またここには、シミュレーションと実機の接続も重要になると考える。ハードウェアのパラメータは簡単にランダム化できるものではないため、Real2Sim や Sim2Real によりシミュレーションと実機を相互作用させながら最適化を行必要がある。

次に、本研究でタスクごとに構築した身体図式を、単一のネットワークにより表現し、自己組織化させていく新しい知能ロボットシステムの構築が挙げられる。一つのネットワーク内でそれぞれのタスクが必要な感覚や制御入力を使いながら自己組織化することで、共通表現を効率的に利用することができ、タスク間のスイッチングや管理コストにおいてもメリットがある。そして、タスク間遷移を表現する計画器も同一のネットワーク内に包含されることが望ましい。また、様々なタスクや身体に共通な身体図式の共通特徴量を取り出し、これを使って様々なタスクや身体に転移させていく方法も考えられる。最終的には、一つのアルゴリズムで動作計画から認識と状態推定、制御や反射等が自律的に獲得されていく手法を開発していきたい。

最後に、本研究を受けての今後のロボットの在り方について考える。筋骨格型や低剛性樹脂製のヒューマノイドのような柔軟で冗長な複雑構造を扱えるようになった今、これらを積極的に取り入れたロボット構成を考えていくべきである。人体の利点をよりロボットに取り込んだハイブリッドなヒューマノイド開発により、ロボットがより人間らしく、賢く振る舞うことができると考える。また、固定された身体や道具、動作環境を飛び出し、身の回りの金属棒を身体や道具として取り入れていくような新しいロボットの形態も非常に興味深い。ハードウェアや制御アルゴリズムが進化した今、次に必要となるのは身体構造や制御システムのより高度な環境適応性である。決まった環境や決まった身体だけではなく、その外側でも自律的な学習により動き続けられるロボットの開発に力を注いでいきたい。

謝辭

本論文は筆者が東京大学大学院情報理工学系研究科知能機械情報学専攻に在学中、稲葉雅幸教授、岡田慧教授の御指導の下に執筆したものです。

稲葉先生は多角的な視点から自身の研究を発散させていただきました。元々筋骨格ヒューマノイドに関する研究であった本論文を知能ロボットという広い枠組みで捉え直し、広く様々なロボットで実験・検証を行う良い機会となり、最後まで走り抜けることができました。

岡田先生は自分とは全く異なる観点から本研究を見てくださいました。本論文の構成を見つめ直す良い機会となると同時に、論文の主張の作り方についても様々勉強させていただきました。

國吉康夫教授、深尾隆則教授、竹内昌治教授には、本論文の審査において、ご多忙の中貴重な時間を割いていただき、多くの有意義な指摘を頂きましたこと、深く感謝しております。

また、これまで関わってきた多くの先生・先輩・後輩には感謝してもし切れません。卒論・修論を共に駆け抜けた同期の牧野、ロボットの修理や実験を手伝ってくれた永松さん、片山さん、川村さん、鬼塚、都築、真壁、新城、西浦くん、大村くん、古賀くん。自分の博士最終年度に実験や修理、議論を手伝ってくれた中島さん、利光、楠山、三木、鈴木くん、李林くん。研究の環境を整えてくださった小島先生、浅野先生、垣内先生、川崎さん。全員を挙げることはできませんが、皆様のおかげで楽しく実りある6年間を過ごすことができたと思っています。これからの知能ロボット研究を引っ張っていけるよう、今後も研究・教育に精進しようと思います。

最後に、遠く暑い熊谷の地から応援してくれる家族、常に自分を励まし応援してくれた優香に感謝し、謝辞を締めたいと思います。ありがとうございました。

2021 年 12 月 3 日 河原塚 健人

発表文献

筆頭著者論文

学術論文誌

1. Kento Kawaharazuka, Kei Okada, Masayuki Inaba. Adaptive Robotic Tool-Tip Control Learning Considering Online Changes in Grasping State. IEEE Robotics and Automation Letters, Vol. 6 No.3 pp. 5992-5999, 2021.
2. Kento Kawaharazuka, Yoichiro Kawamura, Kei Okada, Masayuki Inaba. Imitation Learning with Additional Constraints on Motion Style using Parametric Bias. IEEE Robotics and Automation Letters, Vol. 6 No. 3 pp. 5897-5904, 2021.
3. Kento Kawaharazuka, Manabu Nishiura, Yuya Koga, Yusuke Omura, Yasunori Toshimitsu, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Automatic Grouping of Redundant Sensors and Actuators Using Functional and Spatial Connections: Application to Muscle Grouping for Musculoskeletal Humanoids. IEEE Robotics and Automation Letters, Vol. 6 No. 2 pp. 1981-1988, 2021.
4. Kento Kawaharazuka, Kei Tsuzuki, Yuya Koga, Yusuke Omura, Tasuku Makabe, Koki Shinjo, Moritaka Onitsuka, Yuya Nagamatsu, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Toward Autonomous Driving by Musculoskeletal Humanoids: Study of Developed Hardware and Learning-Based Software. IEEE Robotics and Automation Magazine, Vol. 27 No. 3 pp. 84-96, 2020.
5. Kento Kawaharazuka, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Object Recognition, Dynamic Contact Simulation, Detection, and Control of the Flexible Musculoskeletal Hand Using a Recurrent Neural Network With Parametric Bias. IEEE Robotics and Automation Letters, Vol. 5 No. 3 pp. 4580-4587, 2020.
6. Kento Kawaharazuka, Naoki Hiraoka, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Estimation and Control of Motor Core Temperature with Online Learning of Thermal Model Parameters: Application to Musculoskeletal Humanoids. IEEE Robotics and Automation Letters, Vol. 5 No. 3 pp. 4273-4280, 2020.
7. Kento Kawaharazuka, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Musculoskeletal AutoEncoder: A Unified Online Acquisition Method of Intersens-

- sory Networks for State Estimation, Control, and Simulation of Musculoskeletal Humanoids. IEEE Robotics and Automation Letters, Vol. 5 No. 2 pp. 2411-2418, 2020.
8. Kento Kawaharazuka, Shogo Makino, Masaya Kawamura, Shinsuke Nakashima, Yuki Asano, Kei Okada, Masayuki Inaba. Human Mimetic Forearm and Hand Design with a Radioulnar Joint and Flexible Machined Spring Finger for Human Skillful Motions. Journal of Robotics and Mechatronics, Vol. 32 No. 2 pp. 2411-2418, 2020
 9. Kento Kawaharazuka, Kei Tsuzuki, Shogo Makino, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Long-time Self-body Image Acquisition and its Application to the Control of Musculoskeletal Structures. IEEE Robotics and Automation Letters, Vol. 4 No. 3 pp. 2965-2972, 2019.
 10. Kento Kawaharazuka, Shogo Makino, Masaya Kawamura, Yuki Asano, Kei Okada, Masayuki Inaba. Online Learning of Joint-Muscle Mapping using Vision in Tendon-driven Musculoskeletal Humanoids. IEEE Robotics and Automation Letters, Vol. 3 No. 2 pp. 772-779, 2018.
 11. Kento Kawaharazuka, Masaya Kawamura, Shogo Makino, Yuki Asano, Kei Okada, Masayuki Inaba. Antagonist Inhibition Control in Redundant Tendon-driven Structures Based on Human Reciprocal Innervation for Wide Range Limb Motion of Musculoskeletal Humanoids. IEEE Robotics and Automation Letters, Vol. 2 No. 4 pp. 2119-2126, 2017.

国際会議

12. Kento Kawaharazuka, Koki Shinjo, Yoichiro Kawamura, Kei Okada, Masayuki Inaba. Environmentally Adaptive Control Including Variance Minimization Using Stochastic Predictive Network with Parametric Bias: Application to Mobile Robots. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2021), pp. 8381-8387, 2021.
13. Kento Kawaharazuka, Yasunori Toshimitsu, Manabu Nishiura, Yuya Koga, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Design Optimization of Musculoskeletal Humanoids with Maximization of Redundancy to Compensate for Muscle Rupture. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2021), pp. 3204-3210, 2021.

14. Kento Kawaharazuka, Naoki Hiraoka, Yuya Koga, Manabu Nishiura, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Online Learning of Danger Avoidance for Complex Structures of Musculoskeletal Humanoids and Its Applications. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2020)*, pp. 349-355, 2021.
15. Kento Kawaharazuka, Yuya Koga, Manabu Nishiura, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Motion Modification Method of Musculoskeletal Humanoids by Human Teaching Using Muscle-Based Compensation Control. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2020)*, pp. 83-89, 2021.
16. Kento Kawaharazuka, Manabu Nishiura, Shinsuke Nakashima, Yasunori Toshimitsu, Yusuke Omura, Yuya Koga, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Stability Recognition with Active Vibration for Bracing Behaviors and Motion Extensions Using Environment in Musculoskeletal Humanoids. In *Proceedings of the 2021 IEEE International Conference on Soft Robotics (ROBOSOFT2021)*, pp. 126-133, 2021.
17. Kento Kawaharazuka, Yuya Koga, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Exceeding the Maximum Speed Limit of the Joint Angle for the Redundant Tendon-driven Structures of Musculoskeletal Humanoids. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2020)*, pp. 3585-3590, 2020.
18. Kento Kawaharazuka, Yuya Koga, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Applications of Stretch Reflex for the Upper Limb of Musculoskeletal Humanoids: Protective Behavior, Postural Stability, and Active Induction. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2020)*, pp. 3598-3603, 2020.
19. Kento Kawaharazuka, Kei Tsuzuki, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Stable Tool-Use with Flexible Musculoskeletal Hands by Learning the Predictive Model of Sensor State Transition. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA2020)*, pp. 4572-4578, 2020.
20. Kento Kawaharazuka, Shogo Makino, Kei Tsuzuki, Moritaka Onitsuka, Y. Nagamatsu, Koki Shinjo,

- Tasuku Makabe, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Component Modularized Design of Musculoskeletal Humanoid Platform Musashi to Investigate Learning Control Systems. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2019)*, pp. 7294-7301, 2019.
21. Kento Kawaharazuka, Kei Tsuzuki, Shogo Makino, Moritaka Onitsuka, Koki Shinjo, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Task-specific Self-body Controller Acquisition by Musculoskeletal Humanoids: Application to Pedal Control in Autonomous Driving. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2019)*, pp. 813-818, 2019.
22. Kento Kawaharazuka, Kei Tsuzuki, Moritaka Onitsuka, Yuya Koga, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Reflex-based Motion Strategy of Musculoskeletal Humanoids under Environmental Contact Using Muscle Relaxation Control. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2019)*, pp. 114-119, 2019.
23. Kento Kawaharazuka, Kei Tsuzuki, Shogo Makino, Yuki Asano, Kei Okada, Masayuki Inaba. Modeling and Online Learning of Musculoskeletal Intersensory Networks for Static Controls of Tendon-driven Humanoids. In *Proceedings of 9th International Symposium on Adaptive Motion of Animals and Machines (AMAM2019)*, 2019.
24. Kento Kawaharazuka, Tasuku Makabe, Shogo Makino, Kei Tsuzuki, Yuya Nagamatsu, Yuki Asano, T. Shirai, F. Sugai, Kei Okada, Koji Kawasaki, Masayuki Inaba. TWIMP: Two-Wheel Inverted Musculoskeletal Pendulum as a Learning Control Platform in the Real World with Environmental Physical Contact. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2018)*, pp. 784-790, 2018.
25. Kento Kawaharazuka, Shogo Makino, Masaya Kawamura, Yuki Asano, Kei Okada, Masayuki Inaba. A Method of Joint Angle Estimation Using Only Relative Changes in Muscle Lengths for Tendon-driven Humanoids with Complex Musculoskeletal Structures. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2018)*, pp. 1128-1135, 2018.
26. Kento Kawaharazuka, Shogo Makino, Masaya Kawamura, Ayaka Fujii, Yuki Asano, Kei Okada,

Masayuki Inaba. Online Self-body Image Acquisition Considering Changes in Muscle Routes Caused by Softness of Body Tissue for Tendon-driven Musculoskeletal Humanoids. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2018)*, pp. 1711-1717, 2018.

27. Kento Kawaharazuka, Shogo Makino, Masaya Kawamura, Yuki Asano, Yohei Kakiuchi, Kei Okada, Masayuki Inaba. Human Mimetic Forearm Design with Radioulnar Joint using Miniature Bone-muscle Modules and its Applications. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*, pp. 4956-4962, 2017.

国内学術講演会

28. 河原塚 健人, 河村 洋一郎, 岡田 慧, 稲葉 雅幸. Parametric Bias を用いた動作スタイルを制約可能な模倣学習. 第 39 回日本ロボット学会学術講演会講演論文集, 1I3-01, 2021.
29. 河原塚 健人, 西浦 学, 大村 柚介, 古賀 悠矢, 利光 泰徳, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 機能的・空間的接続を利用した冗長なセンサ・アクチュエータの自動分割: 筋骨格ヒューマノイドの筋分割への適用. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 1A1-D06, 2021.
30. 河原塚 健人, 利光 泰徳, 西浦 学, 古賀 悠矢, 大村 柚介, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋破断を補償する冗長性を最大限活用した筋骨格ヒューマノイドの設計最適化. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 2P3-H04, 2021.
31. 河原塚 健人, 平岡 直樹, 都築 敬, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 温度モデルパラメータのオンライン学習を用いたモータコア温度推定と制御: 筋骨格ヒューマノイドへの適用. 第 21 回 SICE システムインテグレーション部門講演会講演概要集, 2F3-14, 2020.
32. 河原塚 健人, 古賀 悠矢, 都築 敬, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 拮抗筋抑制制御と拮抗筋予見伸長制御による冗長な筋を有する筋骨格ヒューマノイドの最大関節速度を突破する動作戦略. 第 21 回 SICE システムインテグレーション部門講演会講演概要集, 2D2-08, 2020.
33. 河原塚 健人, 都築 敬, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. Parametric Bias を含む再帰型ニューラルネットワークを用いた柔軟ハンドの物体認識・動的接触制御/検知/シミュレーション. 第 38 回日本ロボット学会学術講演会講演論文集, 2A1-05, 2020.

34. 河原塚 健人, 都築 敬, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸 Musculoskeletal AutoEncoder: 筋骨格ヒューマノイドの状態推定・制御・シミュレーションを統一的に扱う筋骨格センサ間ネットワークのオンライン獲得手法. 第37回日本ロボット学会学術講演会講演論文集, 3B3-06, 2019.
35. 河原塚 健人, 牧野 将吾, 都築 敬, 鬼塚 盛宇, 永松 祐弥, 新城 光樹, 真壁 佑, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸 学習制御模索のためのモジュラー型筋骨格プラットフォームの設計開発. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 2P1-C06, 2019.
36. 河原塚 健人, 都築 敬, 牧野 将吾, 鬼塚 盛宇, 新城 光樹, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸 筋骨格ヒューマノイドにおけるタスク特化した動的自己身体制御の獲得 - 自動運転におけるペダル操作への応用 -. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-L08, 2019.
37. 河原塚 健人, 都築 敬, 牧野 将吾, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 稲葉 雅幸 筋骨格構造における長期的自己身体像獲得と可変剛性制御の実現. 第36回日本ロボット学会学術講演会講演論文集, 1J2-02, 2018.
38. 河原塚 健人, 真壁 佑, 牧野 将吾, 都築 敬, 永松 祐弥, 浅野 悠紀, 白井 拓磨, 菅井 文仁, 岡田 慧, 稲葉 雅幸 環境接触を伴う学習型制御研究のための筋骨格型倒立二輪ロボットの開発. 第36回日本ロボット学会学術講演会講演論文集, 1P2-02, 2018.
39. 河原塚 健人, 牧野 将吾, 陳 相羽, 藤井 綺香, 川村 将矢, 真壁 佑, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸 擬似球関節モジュールにより冗長な非線形弾性要素を制御可能な筋骨格ヒューマノイドの上肢設計. 日本機械学会ロボティクス・メカトロニクス講演会'18 講演論文集, 2A2-G09, 2018.
40. 河原塚 健人, 牧野 将吾, 川村 将矢, 藤井 綺香, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドにおける身体組織の柔軟性による筋経路変化を考慮した逐次的自己身体像の獲得. 第23回ロボティクスシンポジウム予稿集, pp. 306-312, 2018.
41. 河原塚 健人, 牧野 将吾, 川村 将矢, 浅野 悠紀, 岡田 慧, 稲葉 雅幸 筋骨格ヒューマノイドにおける視覚を利用した関節-筋空間マップの逐次的再学習. 第35回日本ロボット学会学術講演会講演論文集, 2L1-01, 2017.

42. 河原塚 健人, 牧野 将吾, 川村 将矢, 浅野 悠紀, 岡田 慧, 稲葉 雅幸 骨構造一体小型筋モジュールにより構成された橈骨尺骨構造を有する前腕部の設計. 日本機械学会ロボティクス・メカトロニクス講演会'17 講演論文集, 1A1-O11, 2017.

受賞

43. Moritaka Onitsuka, Manabu Nishiura, Kento Kawaharazuka, Kei Tsuzuki, Yasunori Toshimitsu, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Development of Musculoskeletal Legs with Planar Interskeletal Structures to Realize Human Comparable Moving Function. *Best Oral Paper Award & Mike Stilman Award Finalist, The 2020 IEEE-RAS International Conference on Humanoid Robots*, 2021.7.21.
44. 河原塚健人. 筋骨格ヒューマノイドの状態推定・制御・シミュレーションを統一的に扱う筋骨格センサ間ネットワークのオンライン獲得手法. 日本ロボット学会第 35 回研究奨励賞, 第 37 回日本ロボット学会学術講演会, 2019.9.8.
45. Shogo Makino, Kento Kawaharazuka, Masaya Kawamura, Ayaka Fujii, Tasuku Makabe, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Five-Fingered Hand with Wide Range of Thumb Using Combination of Machined Springs and Variable Stiffness Joints. *IROS ICROS Best Application Paper Award 2018 Finalists, The 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.6.2.
46. Kento Kawaharazuka. Human Mimetic Forearm Design with Radioulnar Joint using Miniature Bone-muscle Modules and its Applications. *IEEE Robotics and Automation Society Japan Chapter Young Award, The 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.6.2.
47. Yuki Asano, Toyotaka Kozuki, Soichiro Ookubo, Masaya Kawamura, Shinsuke Nakashima, Takeshi Katayama, Yanokura Iori, Hirose Toshinori, Kento Kawaharazuka, Shogo Makino, Yohei Kakiuchi, Kei Okada, Masayuki Inaba. Human Mimetic Musculoskeletal Humanoid Kengoro toward Real World Physically Interactive Actions. *Best Interactive Paper Award Finalist, The 2016 IEEE-RAS International Conference on Humanoid Robots*, 2016.6.4.

招待講演・解説

48. 河原塚健人. 深層予測モデル学習によるロボットの時間的・空間的柔軟性攻略. キーノート講演 (OS: 確率ロボティクスとデータ工学ロボティクス`認識・行動学習・記号創発`), 第39回日本ロボット学会学術講演会, 2021.9.6.

共著論文

学術論文誌

49. Yuya Koga, Kento Kawaharazuka, Yasunori Toshimitsu, Manabu Nishiura, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Self-Body Image Acquisition and Posture Generation with Redundancy using Musculoskeletal Humanoid Shoulder Complex for Object Manipulation. *IEEE Robotics and Automation Letters*, Vol. 6 No. 4 pp. 6686-6692, 2021.

国際会議

50. Moritaka Onitsuka, Manabu Nishiura, Kento Kawaharazuka, Kei Tsuzuki, Yasunori Toshimitsu, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Development of Musculoskeletal Legs with Planar Interskeletal Structures to Realize Human Comparable Moving Function. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2020)*, pp. 17-24, 2021.
51. Yasunori Toshimitsu, Kento Kawaharazuka, Manabu Nishiura, Yuya Koga, Yusuke Omura, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Biomimetic Operational Space Control for Musculoskeletal Humanoid Optimizing across Muscle Activation and Joint Nullspace. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA2021)*, pp. 1184-1190, 2021.
52. Shinsuke Nakashima, Kento Kawaharazuka, Manabu Nishiura, Yuki Asano, Yohei Kakiuchi, Kei Okada, Koji Kawasaki, Masayuki Inaba. Restoring Force Design of Active Self-Healing Tension Transmission System and Application to Tendon-Driven Legged Robot. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA2021)*, pp. 7033-7038, 2021.

53. Yasunori Toshimitsu, Kento Kawaharazuka, Kei Tsuzuki, Moritaka Onitsuka Manabu Nishiura, Yuya Koga, Yusuke Omura, Motoki Tomita, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Biomimetic Control Scheme for Musculoskeletal Humanoids Based on Motor Directional Tuning in the Brain. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2020)*, pp. 7784-7791, 2020.
54. Takuzumi Nishio, Moju Zhao, Fan Shi, Tomoki Anzai, Kento Kawaharazuka, Kei Okada, Masayuki Inaba. Stable Control in Climbing and Descending Flight under Upper Walls using Ceiling Effect Model based on Aerodynamics. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA2020)*, pp. 172-178, 2020.
55. Koki Shinjo, Kento Kawaharazuka, Yuki Asano, Shinsuke Nakashima, Shogo Makino, Moritaka Onitsuka, Kei Tsuzuki, Kei Okada, Koji Kawasaki, Masayuki Inaba. Foot with a Core-shell Structural Six-axis Force Sensor for Pedal Depressing and Recovering from Foot Slipping during Pedal Pushing Toward Autonomous Driving by Humanoids. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2019)*, pp. 3049-3054, 2019.
56. Shinsuke Nakashima, Takuma Shirai, Kento Kawaharazuka, Yuki Asano Yohei Kakiuchi, Kei Okada, Masayuki Inaba. An Approach of Facilitated Investigation of Active Self-healing Tension Transmission System Oriented for Legged Robots. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2019)*, pp. 2567-2572, 2019.
57. Tasuku Makabe, Takuma Shirai, Yuya Nagamatsu, Kento Kawaharazuka, Sugai Fumihito, Kei Okada, Masayuki Inaba. Development of Joint Module with Two-Speed Gear Transmission and Joint Lock Mechanism during Driving for Task Adaptable Robot. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2019)*, pp. 5123-5130, 2019.
58. Yuya Koga, Kento Kawaharazuka, Moritaka Onitsuka, Tasuku Makabe, Kei Tsuzuki, Yusuke Omura, Yuki Asano, Kei Okada, Masayuki Inaba. Modification of Muscle Antagonistic Relations and Hand Trajectory on the Dynamic Motion of Musculoskeletal Humanoid. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2019)*, pp. 632-637, 2019.
59. Yuki Asano, Shinsuke Nakashima, Iori Yanokura, Moritaka Onitsuka, Kento Kawaharazuka, Kei Tsuzuki, Yuya Koga, Yusuke Omura, Kei Okada, Masayuki Inaba. Ankle-Hip-Stepping Stabilizer

- on Tendon-Driven Humanoid Kengoro by Integration of Muscle-Joint-Work Space Controllers for Knee-Stretched Humanoid Balance. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2019)*, pp. 397-402, 2019.
60. Tasuku Makabe, Kento Kawaharazuka, Kei Tsuzuki, Kentaro Wada, Shogo Makino, Masaya Kawamura, Ayaka Fujii, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Development of Movable Binocular High-Resolution Eye-Camera Unit for Humanoid and the Evaluation of Looking Around Fixation Control and Object Recognition. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2018)*, pp. 840-845, 2018.
61. Shogo Makino, Kento Kawaharazuka, Masaya Kawamura, Ayaka Fujii, Tasuku Makabe, Moritaka Onitsuka, Yuki Asano, Kei Okada, Koji Kawasaki, Masayuki Inaba. Five-Fingered Hand with Wide Range of Thumb Using Combination of Machined Springs and Variable Stiffness Joints. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2018)*, pp. 4562-4567, 2018.
62. Ayaka Fujii, Shinsuke Nakashima, Masaya Kawamura, Kento Kawaharazuka, Shogo Makino, Yuki Asano, Kei Okada, Masayuki Inaba. Development and Functional Evaluation of a Deformable Membrane Capsule for an Open Ball Glenohumeral Joint. In *Proceedings of The 2018 IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics (BIOROB2018)*, pp. 853-858, 2018.
63. Shogo Makino, Kento Kawaharazuka, Masaya Kawamura, Yuki Asano, Kei Okada, Masayuki Inaba. High-power, flexible, robust hand: Development of musculoskeletal hand using machined springs and realization of self-weight supporting motion with humanoid. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*, pp. 1187-1192, 2017.
64. Yuki Asano, Toyotaka Kozuki, Soichiro Ookubo, Masaya Kawamura, Shinsuke Nakashima, Takeshi Katayama, Yanokura Iori, Hirose Toshinori, Kento Kawaharazuka, Shogo Makino, Yohei Kakiuchi, Kei Okada, Masayuki Inaba. Human Mimetic Musculoskeletal Humanoid Kengoro toward Real World Physically Interactive Actions. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS2016)*, pp. 876-883, 2016.

65. 古賀 悠矢, 河原塚 健人, 利光 泰徳, 西浦 学, 大村 柚介, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋骨格ヒューマノイドの肩複合体における冗長性を活かした姿勢生成と物体操作を目的とした自己身体像の実機学習. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 1A1-D07, 2021.
66. 若林 隼平, 北川 晋吾, 河原塚 健人, 室岡 貴之, 岡田 慧, 稲葉 雅幸. 視覚情報に基づく食器類の把持の冗長性を考慮した自己教師あり把持学習. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 1A1-F09, 2021.
67. 大村 柚介, 河原塚 健人, 永松 祐弥, 古賀 悠矢, 西浦 学, 利光 泰徳, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋骨格ヒューマノイドによる人体模倣両耳聴を用いた視野外環境認識行動. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 2A1-I15, 2021.
68. 西浦 学, 河原塚 健人, 利光 泰徳, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 接触状態を含む身体モデルと強化学習を用いた筋骨格ヒューマノイドによる環境接触行動. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 2A1-I16, 2021.
69. 利光 泰徳, 河原塚 健人, 西浦 学, 古賀 悠矢, 大村 柚介, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋骨格ヒューマノイド腕部の筋・関節冗長性を活用したタスク空間における制御. 日本機械学会ロボティクス・メカトロニクス講演会'21 講演論文集, 2P2-G15, 2021.
70. 大村 柚介, 河原塚 健人, 永松 祐弥, 古賀 悠矢, 西浦 学, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 人体外耳機構を模したヒューマノイドの両耳間スペクトル差学習に基づく空間音源方向推定システム. 第 21 回 SICE システムインテグレーション部門講演会講演概要集, 1C3-17, 2020.
71. 鬼塚 盛宇, 西浦 学, 河原塚 健人, 都築 敬, 利光 泰徳, 大村 柚介, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 面状骨格間構造を利用し広い可動域においてモーメントアームを確保し高出力での環境接触動作が可能な筋骨格脚の開発. 第 38 回日本ロボット学会学術講演会講演論文集, 2G2-08, 2020.
72. 利光 泰徳, 河原塚 健人, 都築 敬, 鬼塚 盛宇, 西浦 学, 古賀 悠矢, 大村 柚介, 富田 幹, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. Motor Directional Tuning 現象に基づく筋張力制御による筋骨格ヒューマノイドの上肢動作. 日本機械学会ロボティクス・メカトロニクス講演会'20 講演論文集, 1P1-G05, 2020.

73. 大村 柚介, 河原塚 健人, 永松 祐弥, 都築 敬, 鬼塚 盛宇, 古賀 悠矢, 西浦 学, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 外耳構造を有し音響処理を行う人体模倣ヒューマノイドの耳機構の設計開発. 日本機械学会ロボティクス・メカトロニクス講演会'20 講演論文集, 1A1-E12, 2020.
74. 中島 慎介, 河原塚 健人, 浅野 悠紀, 垣内 洋平, 岡田 慧, 稲葉 雅幸. 自己修復張力伝達モジュールを備える腱駆動脚ロボットの開発. 第 37 回日本ロボット学会学術講演会講演論文集, 1K3-01, 2019.
75. 西浦 学, 河原塚 健人, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドにおける環境物体に応じた適応的剛性レンジ選択とその可変剛性制御戦略の獲得. 第 37 回日本ロボット学会学術講演会講演論文集, 1K3-06, 2019.
76. 浅野 悠紀, 都築 敬, 河原塚 健人, 鬼塚 盛宇, 古賀 悠矢, 大村 柚介, 永松 祐弥, 真壁 佑, 藤井 綺香, 新城 光樹, 中島 慎介, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 腱駆動ヒューマノイドにおける認識判断操作統合に基づく自動車運転の実証実験. 第 37 回日本ロボット学会学術講演会講演論文集, 3L2-06, 2019.
77. 真壁 佑, 白井 拓磨, 永松 裕弥, 河原塚 健人, 菅井 文仁, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 用途適応型ロボットののための, 駆動時二段可変減速非駆動時ロック機構を持つ関節モジュールの設計開発. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 2A2-F08, 2019.
78. 都築 敬, 河原塚 健人, 真壁 佑, 鬼塚 盛宇, 牧野 将吾, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 可動眼球と自己身体を用いた距離認識機能の獲得. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-M10, 2019.
79. 大村 柚介, 河原塚 健人, 牧野 将吾, 鬼塚 盛宇, 新城 光樹, 都築 敬, 古賀 悠矢, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドにおける時系列聴覚情報を用いた打音認識に基づく動作獲得. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-K03, 2019.
80. 古賀 悠矢, 河原塚 健人, 牧野 将吾, 鬼塚 盛宇, 真壁 佑, 都築 敬, 大村 柚介, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドのダイナミック動作における筋の拮抗関係と手先軌道の修正. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-K02, 2019.
81. 新城 光樹, 河原塚 健人, 浅野 悠紀, 中島 慎介, 牧野 将吾, 鬼塚 盛宇, 都築 敬, 岡田 慧, 川崎 宏治, 稲葉 雅幸. コア・シェル構造を有する 6 軸力計測モジュールをつま先・踵に持つ足部ユニット

- を用いた等身大筋骨格腿駆動ヒューマノイドによるペダル踏み・復帰動作の実現. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-K01, 2019.
82. 鬼塚 盛宇, 河原塚 健人, 牧野 将吾, 新城 光樹, 都築 敬, 中島 慎介, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋骨格ヒューマノイドにおける面状牽引構造を有する関節の開発. 日本機械学会ロボティクス・メカトロニクス講演会'19 講演論文集, 1A1-J02, 2019.
83. 都築 敬, 河原塚 健人, 鬼塚 盛宇, 真壁 佑, 牧野 将吾, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドによる自動車運転動作の実現に向けたペダル操作戦略. 第 36 回日本ロボット学会学術講演会講演論文集, 2P1-05, 2018.
84. 鬼塚 盛宇, 真壁 佑, 河原塚 健人, 牧野 将吾, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドにおける脚全体の筋に基づく筋張力 ZMP を用いた平衡動作. 第 36 回日本ロボット学会学術講演会講演論文集, 1J2-05, 2018.
85. 牧野 将吾, 河原塚 健人, 藤井 綺香, 川村 将矢, 真壁 佑, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 組み合わせ切削ばねによる広可動域関節母指関節と可変剛性指関節をもつ人体模倣型五指ハンドの開発. 日本機械学会ロボティクス・メカトロニクス講演会'18 講演論文集, 1P1-H16, 2018.
86. 真壁 佑, 河原塚 健人, 牧野 将吾, 川村 将矢, 藤井 綺香, 鬼塚 盛宇, 浅野 悠紀, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 筋骨格ヒューマノイドにおける可動眼球の開発と車両見回し発進動作の実現. 日本機械学会ロボティクス・メカトロニクス講演会'18 講演論文集, 2A2-G11, 2018.
87. 浅野 悠紀, 川村 将矢, 河原塚 健人, 牧野 将吾, 藤井 綺香, 真壁 佑, 鬼塚 盛宇, 岡田 慧, 川崎 宏治, 稲葉 雅幸. 人体模倣筋骨格ヒューマノイドにおける筋張力を用いた関節空間コントローラによる車両ペダル操作の実現. 日本機械学会ロボティクス・メカトロニクス講演会'18 講演論文集, 2A2-G07, 2018.
88. 藤井綺香, 中島慎介, 川村将矢, 河原塚健人, 牧野将吾, 浅野悠紀, 岡田慧, 稲葉雅幸. 人体の関節包構造に示唆を得た柔軟で伸縮変形可能な膜構造を備えた開放型球関節の開発. 第 18 回 SICE システムインテグレーション部門講演会講演概要集, 3B4-02, 2017.
89. 牧野 将吾, 河原塚 健人, 川村 将矢, 浅野 悠紀, 岡田 慧, 稲葉 雅幸. 筋骨格ヒューマノイドのための切削ばねによる柔軟関節を備えた五指ハンドの開発と自己身体負荷保持動作の実現. 日本機械学会ロボティクス・メカトロニクス講演会'17 講演論文集, 2P1-B08, 2017.

参考文献

- [1] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba. Humanoid Motion Generation System on HRP2-JSK for Daily Life Environment. In *Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation*, pp. 1772–1777, 2005.
- [2] T. Tanioka. Nursing and Rehabilitative Care of the Elderly Using Humanoid Robots. *The Journal of Medical Investigation*, Vol. 66, No. 1.2, pp. 19–23, 2019.
- [3] K. Kaneko, M. Morisawa, S. Kajita, S. Nakaoka, T. Sakaguchi, R. Cisneros, and F. Kanehiro. Humanoid robot HRP-2Kai - Improvement of HRP-2 towards disaster response tasks. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 132–139, 2015.
- [4] DARPA Robotics Challenge. <http://archive.darpa.mil/roboticschallenge/>. Accessed: 2021-06-30.
- [5] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. Boer, T. Koolen, P. Neuhaus, and J. Pratt. Team IHMC’s Lessons Learned from the DARPA Robotics Challenge Trials. *Journal of Field Robotics*, Vol. 32, No. 2, pp. 192–208, 2015.
- [6] J. Lim, I. Lee, I. Shim, H. Jung, H. J. Min, H. Bae, O. Sim, J. Oh, T. Jung, S. Shin, K. Joo, M. Kim, K. Lee, Y. Bok, D.-G. Choi, B. Cho, S. Kim, J. Heo, I. Kim, J. Lee, I. S. Kwon, and J.-H. Oh. Robot System of DRC-HUBO+ and Control Strategy of Team KAIST in DARPA Robotics Challenge Finals. *Journal of Field Robotics*, Vol. 34, No. 4, pp. 802–829, 2016.
- [7] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padiar, F. Polido, G. G. Tighe, and X. Xinjilefu. No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 623–630, 2015.
- [8] K. Kojima, T. Karasawa, T. Kozuki, E. Kuroiwa, S. Yukizaki, S. Iwaishi, T. Ishikawa, R. Koyama, S. Noda, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Development of life-sized high-power humanoid robot JAXON for real-world use. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 838–843, 2015.

- [9] S. Kim, C. Laschi, and B. Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in Biotechnology*, Vol. 31, No. 5, pp. 287–294, 2013.
- [10] C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. J. Kim, and S. Kim. Soft robot review. *International Journal of Control, Automation and Systems*, Vol. 15, No. 1, pp. 3–15, 2017.
- [11] 井上博允. ロボット言語の研究課題. 日本ロボット学会誌, Vol. 2, No. 2, pp. 87–90, 1984.
- [12] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, Vol. 2, No. 3, pp. 189–208, 1971.
- [13] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - The Planning Domain Definition Language. Technical Report TR-98-003, Yale Center for Computational Vision and Control, 1998.
- [14] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, Vol. 2, No. 1, pp. 14–23, 1986.
- [15] D. Chapman. *Vision, Instruction and Action*. MIT Press, 1991.
- [16] Y. Kuniyoshi and S. Suzuki. Dynamic emergence and adaptation of behavior through embodiment as coupled chaotic field. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2042–2049, 2004.
- [17] A. Crespi, D. Lachat, A. Pasquier, and A. J. Ijspeert. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*, Vol. 25, No. 1, pp. 3–13, 2008.
- [18] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, 1949.
- [19] Y. Kuniyoshi and S. Shinji. Early motor development from partially ordered neural-body dynamics: experiments with a cortico-spinal-musculo-skeletal model. *Biological Cybernetics*, Vol. 95, No. 6, pp. 589–605, 2006.
- [20] C. J. Watkins and P. Dayan. Technical Note: Q-Learning. *Machine Learning*, Vol. 8, pp. 279–292, 1992.
- [21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889–1897, 2015.

- [22] Y. Yamashita and J. Tani. Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLOS Computational Biology*, Vol. 4, No. 11, pp. 1–18, 2008.
- [23] P. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata. Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning. *IEEE Robotics and Automation Letters*, Vol. 2, No. 2, pp. 397–403, 2017.
- [24] I. Lenz, R. Knepper, and A. Saxena. DeepMPC: Learning deep latent features for model predictive control. In *Proceedings of the 2015 Robotics: Science and Systems*, 2015.
- [25] D. Tanaka, S. Arnold, and K. Yamazaki. EMD Net: An Encode-Manipulate-Decode Network for Cloth Manipulation. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1771–1778, 2018.
- [26] K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, Vol. 12, No. 7, pp. 961–974, 1999.
- [27] A. Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, Vol. 36, No. 3, pp. 181–204, 2013.
- [28] K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, Vol. 11, No. 2, pp. 127–138, 2013.
- [29] H. Head and G. Holmes. Sensory disturbances from cerebral lesions. *Brain*, Vol. 34, No. 2–3, pp. 102–254, 1911.
- [30] S. Gallagher. Body Image and Body Schema: A Conceptual Clarification. *The Journal of Mind and Behavior*, Vol. 7, No. 4, pp. 541–554, 1986.
- [31] P. Haggard and D. M. Wolpert. Disorders of Body Scheme. In *In Higher-Order Motor Disorders*, Ed. Freund, Jeannerod, Hallett, and Leiguarda. Oxford University Press, 2005.
- [32] P. K. Khosla. and T. Kanade. Parameter identification of robot dynamics. In *Proceedings of the 1985 IEEE Conference on Decision and Control*, pp. 1754–1760, 1985.
- [33] N. Saito, T. Ogata, S. Funabashi, H. Mori, and S. Sugano. How to Select and Use Tools? : Active Perception of Target Objects Using Multimodal Deep Learning. *IEEE Robotics and Automation Letters*, Vol. 6, No. 2, pp. 2517–2524, 2021.

- [34] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 2012 Neural Information Processing Systems*, pp. 1097–1105, 2012.
- [35] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, Vol. 34, No. 4-5, pp. 705–724, 2015.
- [36] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and Q. J. e. al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, Vol. 37, No. 4-5, pp. 421–436, 2018.
- [37] J. Ho and S. Ermon. Generative Adversarial Imitation Learning. In *Proceedings of the 2016 Neural Information Processing Systems*, pp. 4565–4573, 2016.
- [38] Y. Hu and S. X. Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 4350–4355, 2004.
- [39] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, Vol. 11, No. 1, pp. 1–18, 2003.
- [40] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Vol. 43, pp. 59–69, 1982.
- [41] J. A. Hartigan and M. A. Wong. A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, Vol. 28, No. 1, pp. 100–108, 1979.
- [42] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, Vol. 79, No. 8, pp. 2554–2558, 1982.
- [43] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, pp. 1–14, 2014.
- [44] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proceedings of the 2014 Neural Information Processing Systems*, pp. 2672–2680, 2014.

- [45] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.
- [46] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, pp. 1–15, 2015.
- [47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 2017 Neural Information Processing Systems*, pp. 6000–6010, 2017.
- [48] D. Verstraeten, B. Schrauwen, M. D’Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, Vol. 20, No. 3, pp. 391–403, 2007.
- [49] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Proceedings of the 2013 Neural Information Processing Systems*, pp. 2616–2624, 2013.
- [50] F. Zhang, J. Leitner, M. Jürgen, M. Milford, B. Upcroft, and C. Peter. Towards vision-based deep reinforcement learning for robotic motion control. arXiv preprint arXiv:1511.03791, 2015.
- [51] S. Lathuilière, B. Massé, P. Mesejo, and R. Horaud. Deep Reinforcement Learning for Audio-Visual Gaze Control. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1555–1562, 2018.
- [52] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation. *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, pp. 3838–3845, 2020.
- [53] P. T. Kotnik, S. Yurkovich, and Ü. Özgüner. Acceleration feedback for control of a flexible manipulator arm. *Journal of Robotic Systems*, Vol. 5, No. 3, pp. 181–196, 1988.
- [54] W. J. Book. Recursive Lagrangian Dynamics of Flexible Manipulator Arms. *The International Journal of Robotics Research*, Vol. 3, No. 3, pp. 87–101, 1984.
- [55] V. G. Moudgal, W. A. Kwong, K. M. Passino, and S. Yurkovich. Fuzzy learning control for a flexible-link robot. *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp. 199–210, 1995.

- [56] C. Sun, W. He, and J. Hong. Neural Network Control of a Flexible Robotic Manipulator Using the Lumped Spring-Mass Model. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 47, No. 8, pp. 1863–1874, 2017.
- [57] S. K. Pradhan and B. Subudhi. Real-Time Adaptive Control of a Flexible Manipulator Using Reinforcement Learning. *IEEE Transactions on Automation Science and Engineering*, Vol. 9, No. 2, pp. 237–249, 2012.
- [58] Z. Su and K. Khorasani. A neural-network-based controller for a single-link flexible manipulator using the inverse dynamics approach. *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 6, pp. 1074–1086, 2001.
- [59] Y. Yamakawa, A. Namiki, and M. Ishikawa. Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 49–54, 2010.
- [60] M. Hofer, L. Spannagl, and R. D’Andrea. Iterative Learning Control for Fast and Accurate Position Tracking with an Articulated Soft Robotic Arm. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6602–6607, 2019.
- [61] Y. Wu, K. Takahashi, H. Yamada, K. KIM, S. Murata, S. Sugano, and T. Ogata. Dynamic Motion Generation by Flexible-Joint Robot based on Deep Learning using Images. In *Proceeding of the 8th Joint IEEE International Conference on Development and Learning on Epigenetic Robotics*, 2018.
- [62] H. V. Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 121–127, 2015.
- [63] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, Vol. 19, No. 6, pp. 716–723, 1974.
- [64] N. Murata, S. Yoshizawa, and S. Amari. Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, pp. 865–872, 1994.

- [65] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations*, pp. 1–16, 2017.
- [66] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient Neural Architecture Search via Parameters Sharing. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4095–4104, 2018.
- [67] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1–13, 2018.
- [68] Y. Kobayashi, K. Harada, and K. Takagi. Automatic controller generation based on dependency network of multi-modal sensor variables for musculoskeletal robotic arm. *Robotics and Autonomous Systems*, Vol. 118, pp. 55–65, 2019.
- [69] G. J. Bowden, G. C. Dandy, and H. R. Maier. Input determination for neural network models in water resources applications. Part 1—background and methodology. *Journal of Hydrology*, Vol. 301, No. 1, pp. 75–92, 2005.
- [70] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pp. 1620–1626, 2003.
- [71] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, Vol. 5, No. 47, 2020.
- [72] J. Tani. Self-organization of behavioral primitives as multiple attractor dynamics: a robot experiment. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 489–494, 2002.
- [73] I. Kumagai, S. Noda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Whole body joint load reduction control for high-load tasks of humanoid robot through adapting joint torque limitation based on online joint temperature estimation. In *Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 463–468, 2014.
- [74] A. A. Feldbaum. *Optimal control systems*. Academic Press, 1965.
- [75] M. Kawato and H. Gomi. A computational model of four regions of the cerebellum based on feedback-error learning. *Biological Cybernetics*, Vol. 68, No. 2, pp. 95–103, 1992.

- [76] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, Vol. 11, No. 7, pp. 1317–1329, 1998.
- [77] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot HRP-2. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 1083–1090, 2004.
- [78] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The Development of Honda Humanoid Robot. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 1321–1326, 1998.
- [79] M. Hirose and K. Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 365, No. 1850, pp. 11–19, 2007.
- [80] Y. Nagamatsu, T. Shirai, H. Suzuki, Y. Kakiuchi, K. Okada, and M. Inaba. Distributed torque estimation toward low-latency variable stiffness control for gear-driven torque sensorless humanoid. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5239–5244, 2017.
- [81] G. Cheng, E. Dean-Leon, F. Bergner, J. R. G. Olvera, Q. Leboutet, and P. Mittendorfer. A Comprehensive Realization of Robot Skin: Sensors, Sensing, Control, and Applications. *Proceedings of the IEEE*, Vol. 107, No. 10, pp. 2034–2051, 2019.
- [82] R. Niiyama, S. Nishikawa, and Y. Kuniyoshi. Athlete Robot with applied human muscle activation patterns for bipedal running. In *Proceedings of the 2010 IEEE-RAS International Conference on Humanoid Robots*, pp. 498–503, 2010.
- [83] K. Ogawa, K. Narioka, and K. Hosoda. Development of whole-body humanoid “pneumat-BS” with pneumatic musculoskeletal system. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4838–4843, 2011.
- [84] I. Mizuuchi, M. Kawamura, T. Asaoka, and S. Kumakura. Design and development of a compressor-embedded pneumatic-driven musculoskeletal humanoid. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 811–816, 2012.

- [85] H. G. Marques, M. Jäntsh, S. Wittmeier, O. Holland, C. Alessandro, A. Diamond, M. Lungarella, and R. Knight. ECCE1: the first of a series of anthropomorphic musculoskeletal upper torsos. In *Proceedings of the 2010 IEEE-RAS International Conference on Humanoid Robots*, pp. 391–396, 2010.
- [86] Y. Asano, T. Kozuki, S. Ookubo, M. Kawamura, S. Nakashima, T. Katayama, Y. Iori, H. Toshinori, K. Kawaharazuka, S. Makino, Y. Kakiuchi, K. Okada, and M. Inaba. Human Mimetic Musculoskeletal Humanoid Kengoro toward Real World Physically Interactive Actions. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 876–883, 2016.
- [87] K. Kawaharazuka, S. Makino, K. Tsuzuki, M. Onitsuka, Y. Nagamatsu, K. Shinjo, T. Makabe, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Component Modularized Design of Musculoskeletal Humanoid Platform Musashi to Investigate Learning Control Systems. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7294–7301, 2019.
- [88] H. Kobayashi, K. Hyodo, and D. Ogane. On Tendon-Driven Robotic Mechanisms with Redundant Tendons. *The International Journal of Robotics Research*, Vol. 17, No. 5, pp. 561–571, 1998.
- [89] G. Endo, A. Horigome, and A. Takata. Super Dragon: A 10-m-Long-Coupled Tendon-Driven Articulated Manipulator. *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, pp. 934–941, 2019.
- [90] G. Bledt, M. J. Powell, B. Katz, J. D. Carlo, P. M. Wensing, and S. Kim. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2245–2252, 2018.
- [91] M. Chignoli, D. Kim, E. S. Jones, and S. Kim. The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors. arXiv preprint arXiv:2104.09025, 2021.
- [92] N. G. Tsagarakis, S. Morfey, G. M. Cerda, L. Zhibin, and D. G. Caldwell. COMpliant huMANoid COMAN: Optimal joint stiffness tuning for modal frequency control. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, pp. 673–678, 2013.
- [93] C. Fitzgerald. Developing baxter. In *Proceedings of the 2013 IEEE Conference on Technologies for Practical Robot Applications*, pp. 1–6, 2013.

- [94] S. Wolf, O. Eiberger, and G. Hirzinger. The DLR FSJ: Energy based design of a variable stiffness joint. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pp. 5082–5089, 2011.
- [95] S. Kuindersma, R. Deits, M. Fallon, V. Andr s, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, Vol. 40, pp. 429–455, 2016.
- [96] H. Kaminaga, T. Ko, R. Masumura, M. Komagata, S. Sato, S. Yorita, and Y. Nakamura. Mechanism and Control of Whole-Body Electro-Hydrostatic Actuator Driven Humanoid Robot Hydra. In *Proceedings of the 2016 International Symposium on Experimental Robotics*, pp. 656–665, 2017.
- [97] T. Makabe, T. Anzai, Y. Kakiuchi, K. Okada, and M. Inaba. Development of Amphibious Humanoid for Behavior Acquisition on Land and Underwater. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots*, pp. 104–111, 2020.
- [98] K. Kawaharazuka, K. Shinjo, Y. Kawamura, K. Okada, and M. Inaba. Environmentally Adaptive Control Including Variance Minimization Using Stochastic Predictive Network with Parametric Bias: Application to Mobile Robots. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8381–8387, 2021.
- [99] J. Urata, Y. Nakanishi, A. Miyadera, I. Mizuuchi, T. Yoshikai, and M. Inaba. A Three-Dimensional Angle Sensor for a Spherical Joint Using a Micro Camera. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 4428–4430, 2006.
- [100] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, K. Okada, and M. Inaba. Online Learning of Joint-Muscle Mapping using Vision in Tendon-driven Musculoskeletal Humanoids. *IEEE Robotics and Automation Letters*, Vol. 3, No. 2, pp. 772–779, 2018.
- [101] I. Mizuuchi, R. Tajima, T. Yoshikai, D. Sato, K. Nagashima, M. Inaba, Y. Kuniyoshi, and H. Inoue. The Design and Control of the Flexible Spine of a Fully Tendon-Driven Humanoid “Kenta”. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1192–1197, 2004.
- [102] I. Mizuuchi, T. Yoshikai, Y. Sodeyama, Y. Nakanishi, A. Miyadera, T. Yamamoto, T. Niemela, M. Hayashi, J. Urata, Y. Namiki, T. Nishino, and M. Inaba. Development of musculoskeletal

- humanoid Kotaro. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 82–87, 2006.
- [103] I. Mizuuchi, Y. Nakanishi, Y. Sodeyama, Y. Namiki, T. Nishino, N. Muramatsu, J. Urata, K. Hongo, T. Yoshikai, and M. Inaba. An advanced musculoskeletal humanoid kojiro. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 294–299, 2007.
- [104] Y. Sodeyama, T. Yoshikai, T. Nishino, I. Mizuuchi, and M. Inaba. The Designs and Motions of a Shoulder Structure with a Wide Range of Movement Using Bladebone-Collarbone Structures. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3629–3634, 2007.
- [105] I. Mizuuchi, T. Yoshikai, Y. Nakanishi, and M. Inaba. A Reinforceable-Muscle Flexible-Spine Humanoid “Kenji”. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4143–4148, 2005.
- [106] R. Niiyama, S. Nishikawa, and Y. Kuniyoshi. Athlete Robot with applied human muscle activation patterns for bipedal running. In *Proceedings of the 2010 IEEE-RAS International Conference on Humanoid Robots*, pp. 498–503, 2010.
- [107] M. Jäntschi, S. Wittmeier, K. Dalamagkidis, A. Panos, F. Volkart, and A. Knoll. Anthrob - A Printed Anthropomimetic Robot. In *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, pp. 342–347, 2013.
- [108] Y. Nakanishi, T. Izawa, M. Osada, N. Ito, S. Ohta, J. Urata, and M. Inaba. Development of Musculoskeletal Humanoid Kenzoh with Mechanical Compliance Changeable Tendons by Nonlinear Spring Unit. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2384–2389, 2011.
- [109] Y. Nakanishi, Y. Asano, T. Kozuki, H. Mizoguchi, Y. Motegi, M. Osada, T. Shirai, J. Urata, K. Okada, and M. Inaba. Design concept of detail musculoskeletal humanoid “Kenshiro” - Toward a real human body musculoskeletal simulator. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 1–6, 2012.
- [110] S. Kurumaya, K. Suzumori, H. Nabae, and S. Wakimoto. Musculoskeletal lower-limb robot driven by multifilament muscles. *Robomech Journal*, Vol. 3, No. 18, pp. 1–15, 2016.

- [111] 大村柚介, 河原塚健人, 永松祐弥, 都築敬, 鬼塚盛宇, 古賀悠矢, 西浦学, 浅野悠紀, 岡田慧, 川崎宏治, 稲葉雅幸. 外耳構造を有し音響処理を行う人体模倣ヒューマノイドの耳機構の設計開発. 日本機械学会ロボティクス・メカトロニクス講演会'20 講演論文集, pp. 1A1-E12, 2020.
- [112] T. Makabe, K. Kawaharazuka, K. Tsuzuki, K. Wada, S. Makino, M. Kawamura, A. Fujii, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Development of Movable Binocular High-Resolution Eye-Camera Unit for Humanoid and the Evaluation of Looking Around Fixation Control and Object Recognition. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots*, pp. 840–845, 2018.
- [113] T. Kozuki, H. Mizoguchi, Y. Asano, M. Osada, T. Shirai, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Design methodology for the thorax and shoulder of human mimetic musculoskeletal humanoid Kenshiro - a thorax structure with rib like surface -. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3687–3692, 2012.
- [114] T. Kozuki, H. Toshinori, T. Shirai, S. Nakashima, Y. Asano, Y. Kakiuchi, K. Okada, and M. Inaba. Skeletal structure with artificial perspiration for cooling by latent heat for musculoskeletal humanoid Kengoro. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2135–2140, 2016.
- [115] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, Y. Kakiuchi, K. Okada, and M. Inaba. Human Mimetic Forearm Design with Radioulnar Joint using Miniature Bone-muscle Modules and its Applications. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4956–4962, 2017.
- [116] S. Makino, K. Kawaharazuka, M. Kawamura, A. Fujii, T. Makabe, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Five-Fingered Hand with Wide Range of Thumb Using Combination of Machined Springs and Variable Stiffness Joints. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4562–4567, 2018.
- [117] H. Mizoguchi, Y. Asano, T. Izawa, M. Osada, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Biomimetic design and implementation of muscle arrangement around hip joint for musculoskeletal humanoid. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, pp. 1819–1824, 2011.

- [118] M. Onitsuka, M. Nishiura, K. Kawaharazuka, K. Tsuzuki, Y. Toshimitsu, Y. Omura, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Development of Musculoskeletal Legs with Planar Interskeletal Structures to Realize Human Comparable Moving Function. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots*, pp. 17–24, 2021.
- [119] Y. Asano, H. Mizoguchi, T. Kozuki, Y. Motegi, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Achievement of twist squat by musculoskeletal humanoid with screw-home mechanism. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4649–4654, 2013.
- [120] Y. Asano, S. Nakashima, T. Kozuki, S. Ookubo, I. Yanokura, Y. Kakiuchi, K. Okada, and M. Inaba. Human mimetic foot structure with multi-DOFs and multi-sensors for musculoskeletal humanoid Kengoro. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2419–2424, 2016.
- [121] T. Shirai, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Whole body adapting behavior with muscle level stiffness control of tendon-driven multijoint robot. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2229–2234, 2011.
- [122] S. Ookubo, Y. Asano, T. Kozuki, T. Shirai, K. Okada, and M. Inaba. Learning Nonlinear Muscle-Joint State Mapping Toward Geometric Model-Free Tendon Driven Musculoskeletal Robots. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 765–770, 2015.
- [123] M. Kawamura, S. Ookubo, Y. Asano, T. Kozuki, K. Okada, and M. Inaba. A Joint-Space Controller Based on Redundant Muscle Tension for Multiple DOF Joints in Musculoskeletal Humanoids. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 814–819, 2016.
- [124] K. Kawaharazuka, S. Makino, M. Kawamura, A. Fujii, Y. Asano, K. Okada, and M. Inaba. Online Self-body Image Acquisition Considering Changes in Muscle Routes Caused by Softness of Body Tissue for Tendon-driven Musculoskeletal Humanoids. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1711–1717, 2018.

- [125] K. Kawaharazuka, K. Tsuzuki, S. Makino, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Long-time Self-body Image Acquisition and its Application to the Control of Musculoskeletal Structures. *IEEE Robotics and Automation Letters*, Vol. 4, No. 3, pp. 2965–2972, 2019.
- [126] Y. Nakanishi, K. Hongo, I. Mizuuchi, and M. Inaba. Joint proprioception acquisition strategy based on joints-muscles topological maps for musculoskeletal humanoids. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pp. 1727–1732, 2010.
- [127] Y. Motegi, T. Shirai, T. Izawa, T. Kurotobi, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Motion control based on modification of the Jacobian map between the muscle space and work space with musculoskeletal humanoid. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 835–840, 2012.
- [128] I. Mizuuchi, Y. Nakanishi, T. Yoshikai, M. Inaba, H. Inoue, and O. Khatib. Body Information Acquisition System of Redundant Musculo-Skeletal Humanoid. In *Experimental Robotics IX*, pp. 249–258, 2006.
- [129] A. Diamond and O. E. Holland. Reaching control of a full-torso, modelled musculoskeletal robot using muscle synergies emergent under reinforcement learning. *Bioinspiration & Biomimetics*, Vol. 9, No. 1, pp. 1–16, 2014.
- [130] D. Driess, H. Zimmermann, S. Wolfen, D. Suissa, D. Haeufle, D. Hennes, M. Toussaint, and S. Schmitt. Learning to Control Redundant Musculoskeletal Systems with Neural Networks and SQP: Exploiting Muscle Properties. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pp. 6461–6468, 2018.
- [131] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters. Control of Musculoskeletal Systems Using Learned Dynamics Models. *IEEE Robotics and Automation Letters*, Vol. 3, No. 4, pp. 3161–3168, 2018.
- [132] I. Mizuuchi, S. Yoshida, T. Yoshikai, M. Inaba, D. Sato, and H. Inoue. Behavior developing environment for the large-DOF muscle-driven humanoid equipped with numerous sensors. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pp. 1940–1945, 2003.

- [133] Y. Nakanishi, I. Mizuuchi, T. Yoshikai, T. Inamura, and M. Inaba. Pedaling by a redundant musculoskeletal humanoid robot. In *Proceedings of the 2005 IEEE-RAS International Conference on Humanoid Robots*, pp. 68–73, 2005.
- [134] S. Wittmeier, M. Jäntschi, K. Dalamagkidis, M. Rickert, H. G. Marques, and A. Knoll. CALIPER: A universal robot simulation framework for tendon-driven robots. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1063–1068, 2011.
- [135] D. Lau, J. Eden, Y. Tan, and D. Oetomo. CASPR: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3004–3011, 2016.
- [136] S. Trendel, Y. P. Chan, A. Kharchenko, R. Hostettler, A. Knoll, and D. Lau. CARDSFlow: An End-to-End Open-Source Physics Environment for the Design, Simulation and Control of Musculoskeletal Robots. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots*, pp. 245–250, 2018.
- [137] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems in The 29th Annual Conference on Neural Information Processing Systems*, 2015.
- [138] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.
- [139] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, Vol. 12, No. 1, pp. 145–151, 1999.
- [140] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, Vol. 323, No. 6088, pp. 533–536, 1986.
- [141] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pp. 318–362. MIT Press, Cambridge, MA, USA, 1986.

- [142] C. S. I. J. Goodfellow, J. Shlens. Explaining and Harnessing Adversarial Examples. In *Proceedings of the 3rd International Conference on Learning Representations*, pp. 1–11, 2015.
- [143] J. Tani, M. Ito, and Y. Sugita. Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB. *Neural Networks*, Vol. 17, No. 8, pp. 1273–1289, 2004.
- [144] T. Ogata, H. Ohba, J. Tani, K. Komatani, and H. G. Okuno. Extracting multi-modal dynamics of objects using RNNPB. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 966–971, 2005.
- [145] R. Yokoya, T. Ogata, J. Tani, K. Komatani, and H. G. Okuno. Experience Based Imitation Using RNNPB. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3669–3674, 2006.
- [146] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Object Recognition, Dynamic Contact Simulation, Detection, and Control of the Flexible Musculoskeletal Hand Using a Recurrent Neural Network With Parametric Bias. *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, pp. 4580–4587, 2020.
- [147] S. Nishide, T. Nakagawa, T. Ogata, J. Tani, T. Takahashi, and H. G. Okuno. Modeling tool-body assimilation using second-order Recurrent Neural Network. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5376–5381, 2009.
- [148] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical Review E*, Vol. 69, pp. 1–16, 2004.
- [149] PR2 (Willow Garage). <http://www.willowgarage.com/pages/pr2/overview>.
- [150] Fetch (Fetch Robotics). <https://fetchrobotics.com/>.
- [151] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich. Fetch & Freight: Standard Platforms for Service Robot Applications. In *Proceedings of International Joint Conference on Artificial Intelligence - Workshop on Autonomous Mobile Service Robots*, 2016.

- [152] Y. Asano, T. Kozuki, S. Ookubo, K. Kawasaki, T. Shirai, K. Kimura, K. Okada, and M. Inaba. A Sensor-driver Integrated Muscle Module with High-tension Measurability and Flexibility for Tendon-driven Robots. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5960–5965, 2015.
- [153] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics platform for HRP. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2431–2436, 2002.
- [154] M. Osada, N. Ito, Y. Nakanishi, and M. Inaba. Realization of flexible motion by musculoskeletal humanoid “Kojiro” with add-on nonlinear spring units. In *Proceedings of the 2010 IEEE-RAS International Conference on Humanoid Robots*, pp. 174–179, 2010.
- [155] S. Makino, K. Kawaharazuka, M. Kawamura, Y. Asano, K. Okada, and M. Inaba. High-power, flexible, robust hand: Development of musculoskeletal hand using machined springs and realization of self-weight supporting motion with humanoid. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1187–1192, 2017.
- [156] K. Kawaharazuka, S. Makino, M. Kawamura, S. Nakashima, Y. Asano, K. Okada, and M. Inaba. Human Mimetic Forearm and Hand Design with a Radioulnar Joint and Flexible Machined Spring Finger for Human Skillful Motions. *Journal of Robotics and Mechatronics*, Vol. 32, No. 2, pp. 445–458, 2020.
- [157] K. Shinjo, K. Kawaharazuka, Y. Asano, S. Nakashima, S. Makino, M. Onitsuka, K. Tsuzuki, K. Okada, K. Kawasaki, and M. Inaba. Foot with a Core-shell Structural Six-axis Force Sensor for Pedal Depressing and Recovering from Foot Slipping during Pedal Pushing Toward Autonomous Driving by Humanoids. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3049–3054, 2019.
- [158] 大村柚介, 河原塚健人, 永松祐弥, 古賀悠矢, 西浦学, 利光泰徳, 浅野悠紀, 岡田慧, 川崎宏治, 稲葉雅幸. 筋骨格ヒューマノイドによる人体模倣両耳聴を用いた視野外環境認識行動. 日本機械学会ロボティクス・メカトロニクス講演会’21 講演論文集, pp. 2A1–I15, 2021.
- [159] K. Kawaharazuka, T. Makabe, S. Makino, K. Tsuzuki, Y. Nagamatsu, Y. Asano, T. Shirai, F. Sugai, K. Okada, K. Kawasaki, and M. Inaba. TWIMP: Two-Wheel Inverted Musculoskeletal Pendulum as

- a Learning Control Platform in the Real World with Environmental Physical Contact. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots*, pp. 784–790, 2018.
- [160] K. Y. Kin, S. H. Kim, and Y. K. Kwak. Dynamic analysis of a nonholonomic two-wheeled inverted pendulum robot. *Journal of Intelligent and Robotic Systems*, Vol. 44, No. 1, pp. 25–46, 2005.
- [161] S. Jeong and T. Takahashi. Wheeled inverted pendulum type assistant robot: inverted mobile, standing, and sitting motions. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1932–1937, 2007.
- [162] Handle (Boston Dynamics). <https://www.bostondynamics.com/handle>.
- [163] Y. Hosoda, S. Egawa, J. Tamamoto, K. Yamamoto, R. Nakamura, and M. Togami. Basic design of human-symbiotic robot EMIEW. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5079–5084, 2006.
- [164] S. R. Kuindersma, R. A. Grupen, and A. G. Barto. Variable risk control via stochastic optimization. *The International Journal of Robotics Research*, Vol. 32, No. 7, pp. 806–825, 2013.
- [165] K. Kawaharazuka, Y. Toshimitsu, M. Nishiura, Y. Koga, Y. Omura, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Design Optimization of Musculoskeletal Humanoids with Maximization of Redundancy to Compensate for Muscle Rupture. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3204–3210, 2021.
- [166] M. A. Sharbafi, C. Rode, S. Kurowski, D. Scholz, R. Möckel, K. Radkhah, G. Zhao, A. M. Rashty, O. V. Stryk, and A. Seyfarth. A new biarticular actuator design facilitates control of leg function in BioBiped3. *Bioinspiration & Biomimetics*, Vol. 11, No. 4, p. 046003, 2016.
- [167] K. Kawaharazuka, M. Kawamura, S. Makino, Y. Asano, K. Okada, and M. Inaba. Antagonist Inhibition Control in Redundant Tendon-driven Structures Based on Human Reciprocal Innervation for Wide Range Limb Motion of Musculoskeletal Humanoids. *IEEE Robotics and Automation Letters*, Vol. 2, No. 4, pp. 2119–2126, 2017.
- [168] Y. Koga, K. Kawaharazuka, M. Onitsuka, T. Makabe, K. Tsuzuki, Y. Omura, Y. Asano, K. Okada, and M. Inaba. Modification of Muscle Antagonistic Relations and Hand Trajectory on the Dynamic Motion

- of Musculoskeletal Humanoid. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots*, pp. 632–637, 2019.
- [169] K. Kawaharazuka, Y. Koga, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Exceeding the Maximum Speed Limit of the Joint Angle for the Redundant Tendon-driven Structures of Musculoskeletal Humanoids. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3585–3590, 2020.
- [170] L. Kuxhaus, S. S. Roach, and F. J. Valero-Cuevas. Quantifying deficits in the 3D force capabilities of a digit caused by selective paralysis: application to the thumb with simulated low ulnar nerve palsy. *Journal of Biomechanics*, Vol. 38, No. 4, pp. 725–736, 2005.
- [171] Y. Nakanishi, S. Ohta, T. Shirai, Y. Asano, T. Kozuki, Y. Takehashi, H. Mizoguchi, T. Kurotobi, Y. Motegi, K. Sasabuchi, J. Urata, K. Okada, I. Mizuuchi, and M. Inaba. Design Approach of Biologically-Inspired Musculoskeletal Humanoids. *International Journal of Advanced Robotic Systems*, Vol. 10, No. 4, pp. 216–228, 2013.
- [172] R. Niiyama and Y. Kuniyoshi. Design principle based on maximum output force profile for a musculoskeletal robot. *Industrial Robot: An International Journal*, Vol. 37, No. 3, pp. 1–6, 2010.
- [173] Y. Almubarak and Y. Tadesse. Twisted and coiled polymer (TCP) muscles embedded in silicone elastomer for use in soft robot. *International Journal of Intelligent Robotics and Applications*, Vol. 1, No. 3, pp. 352–368, 2017.
- [174] R. Finotello, T. Grasso, G. Rossi, and A. Terribile. Computation of kinetostatic performances of robot manipulators with polytopes. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 3241–3246, 1998.
- [175] J. M. Inouye and F. J. Valero-Cuevas. Anthropomorphic tendon-driven robotic hands can exceed human grasping capabilities following optimization. *The International Journal of Robotics Research*, Vol. 33, No. 5, pp. 694–705, 2014.
- [176] F. J. Valero-Cuevas. *Fundamentals of Neuromechanics*, Vol. 8 of *Biosystems & Biorobotics*. Springer-Verlag, London, 2015.

- [177] F. Fortin, F. D. Rainville, M. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, Vol. 13, pp. 2171–2175, 2012.
- [178] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Musculoskeletal AutoEncoder: A Unified Online Acquisition Method of Intersensory Networks for State Estimation, Control, and Simulation of Musculoskeletal Humanoids. *IEEE Robotics and Automation Letters*, Vol. 5, No. 2, pp. 2411–2418, 2020.
- [179] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, K. Akachi, T. Kawasaki, S. Ota, K. Yokoyama, H. Handa, Y. Fukase, J. i. Maeda, Y. Nakamura, S. Tachi, and H. Inoue. Humanoid robotics platforms developed in HRP. *Robotics and Autonomous Systems*, Vol. 48, No. 4, pp. 165–175, 2004.
- [180] J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Design of high torque and high speed leg module for high power humanoid. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4497–4502, 2010.
- [181] E. Sevinchan, I. Dincer, and H. Lang. A review on thermal management methods for robots. *Applied Thermal Engineering*, Vol. 140, pp. 799–813, 2018.
- [182] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Koley, and E. Todorov. An integrated system for real-time model predictive control of humanoid robots. In *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, pp. 292–299, 2013.
- [183] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber. *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*, pp. 65–93. Springer Berlin Heidelberg, 2006.
- [184] S. Noda, M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Online maintaining behavior of high-load and unstable postures based on whole-body load balancing strategy with thermal prediction. In *Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering*, pp. 1166–1171, 2014.
- [185] J. Urata, T. Hirose, Y. Namiki, Y. Nakanishi, I. Mizuuchi, and M. Inaba. Thermal control of electrical motors for high-power humanoid robots. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2047–2052, 2008.

- [186] K. Kawaharazuka, N. Hiraoka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Estimation and Control of Motor Core Temperature with Online Learning of Thermal Model Parameters: Application to Musculoskeletal Humanoids. *IEEE Robotics and Automation Letters*, Vol. 5, No. 3, pp. 4273–4280, 2020.
- [187] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1310–1318, 2013.
- [188] M. Jäntschi, S. Wittmeier, K. Dalamagkidis, and A. Knoll. Computed muscle control for an anthropomorphic elbow joint. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2192–2197, 2012.
- [189] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Koga, Y. Omura, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Reflex-based Motion Strategy of Musculoskeletal Humanoids under Environmental Contact Using Muscle Relaxation Control. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots*, pp. 114–119, 2019.
- [190] Y. Asano, T. Shirai, T. Kozuki, Y. Motegi, Y. Nakanishi, K. Okada, and M. Inaba. Motion Generation of Redundant Musculoskeletal Humanoid Based on Robot-Model Error Compensation by Muscle Load Sharing and Interactive Control Device. In *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, pp. 336–341, 2013.
- [191] Y. Nakanishi, T. Izawa, T. Kurotobi, J. Urata, K. Okada, and M. Inaba. Achievement of complex contact motion with environments by musculoskeletal humanoid using humanlike shock absorption strategy. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1815–1820, 2012.
- [192] R. Terasawa, S. Noda, K. Kojima, R. Koyama, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Achievement of Dynamic Tennis Swing Motion by Offline Motion Planning and Online Trajectory Modification Based on Optimization with a Humanoid Robot. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 1094–1100, 2016.
- [193] S. Haddadin, M. Weis, S. Wolf, and A. Albu-Schäffer. Optimal Control for Maximizing Link Velocity of Robotic Variable Stiffness Joints. *IFAC Proceedings Volumes*, Vol. 44, No. 1, pp. 6863–6871, 2011.

- [194] L. Chen, M. Garabini, M. Laffranchi, N. Kashiri, N. G. Tsagarakis, A. Bicchi, and D. G. Caldwell. Optimal control for maximizing velocity of the CompAct compliant actuator. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, pp. 516–522, 2013.
- [195] V. Potkonjak, B. Svetozarevic, K. Jovanovic, and O. Holland. The Puller-Follower Control of Compliant and Noncompliant Antagonistic Tendon Drives in Robotic Systems. *International Journal of Advanced Robotic Systems*, Vol. 8, No. 5, pp. 143–155, 2011.
- [196] X. Liu, A. Rosendo, S. Ikemoto, M. Shimizu, and K. Hosoda. Robotic investigation on effect of stretch reflex and crossed inhibitory response on bipedal hopping. *Journal of The Royal Society Interface*, Vol. 15, No. 140, p. 20180024, 2018.
- [197] M. Shimizu, K. Suzuki, K. Narioka, and K. Hosoda. Roll motion control by stretch reflex in a continuously jumping musculoskeletal biped robot. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1264–1269, 2012.
- [198] H. Geyer and H. Herr. A Muscle-Reflex Model That Encodes Principles of Legged Mechanics Produces Human Walking Dynamics and Muscle Activities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 18, No. 3, pp. 263–273, 2010.
- [199] M. Folgheraiter and G. Gini. Human-like reflex control for an artificial hand. *Biosystems*, Vol. 76, No. 1, pp. 65–74, 2004.
- [200] H. G. Marques, F. Imtiaz, F. Iida, and R. Pfeifer. Self-organization of reflexive behavior from spontaneous motor activity. *Biological Cybernetics*, Vol. 107, No. 1, pp. 25–37, 2013.
- [201] K. Kawaharazuka, Y. Koga, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Applications of Stretch Reflex for the Upper Limb of Musculoskeletal Humanoids: Protective Behavior, Postural Stability, and Active Induction. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3598–3603, 2020.
- [202] F. Doemges and P. M. Rack. Task-dependent changes in the response of human wrist joints to mechanical disturbance. *The Journal of Physiology*, Vol. 447, No. 1, pp. 575–585, 1992.

- [203] E. Huber and K. Baker. Using a hybrid of silhouette and range templates for real-time pose estimation. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 1652–1657, 2004.
- [204] Y. Zhu, Y. Zhao, and S. Zhu. Understanding tools: Task-oriented object modeling, learning and recognition. In *Proceedings of the 2015 IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 2855–2864, 2015.
- [205] K. P. Tee, J. Li, L. T. P. Chen, K. W. Wan, and G. Ganesh. Towards Emergence of Tool Use in Robots: Automatic Tool Recognition and Use Without Prior Tool Learning. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pp. 6439–6446, 2018.
- [206] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Proceedings of the 2018 Robotics: Science and Systems*, 2018.
- [207] K. Kawaharazuka, T. Ogawa, and C. Nabeshima. Tool Shape Optimization through Backpropagation of Neural Network. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8387–8393, 2020.
- [208] A. T. Miller and P. K. Allen. Graspit! A versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, Vol. 11, No. 4, pp. 110–122, 2004.
- [209] Y. Xue and Y. B. Jia. Gripping a Kitchen Knife on the Cutting Board. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 9226–9231, 2020.
- [210] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, Vol. 4, No. 26, 2019.
- [211] H. Hoffmann, Z. Chen, D. Earl, D. Mitchell, B. Salemi, and J. Sinapov. Adaptive robotic tool use under variable grasps. *Robotics and Autonomous Systems*, Vol. 62, No. 6, pp. 833–846, 2014.
- [212] C. Nabeshima, Y. Kuniyoshi, and M. Lungarella. Adaptive body schema for robotic tool-use. *Advanced Robotics*, Vol. 20, No. 10, pp. 1105–1126, 2006.

- [213] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning Task-Oriented Grasping for Tool Manipulation with Simulated Self-Supervision. In *Proceedings of the 2018 Robotics: Science and Systems*, 2018.
- [214] A. Xie, F. Ebert, S. Levine, and C. Finn. Improvisation through Physical Understanding: Using Novel Objects as Tools with Visual Foresight. arXiv preprint arXiv:1904.05538, 2019.
- [215] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots*, pp. 7–12, 2006.
- [216] K. Takahashi, K. Kim, T. Ogata, and S. Sugano. Tool-body assimilation model considering grasping motion through deep learning. *Robotics and Autonomous Systems*, Vol. 91, pp. 115–127, 2017.
- [217] M. Eppe, P. D. H. Nguyen, and S. Wermter. From Semantics to Execution: Integrating Action Planning With Reinforcement Learning for Robotic Causal Problem-Solving. *Frontiers in Robotics and AI*, Vol. 6, p. 123, 2019.
- [218] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, Vol. 39, No. 2-3, pp. 202–216, 2020.
- [219] T. Mar, V. Tikhonoff, and L. Natale. What Can I Do With This Tool? Self-Supervised Learning of Tool Affordances From Their 3-D Geometry. *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 10, No. 3, pp. 595–610, 2018.
- [220] K. Kawaharazuka, K. Okada, and M. Inaba. Adaptive Robotic Tool-Tip Control Learning Considering Online Changes in Grasping State. *IEEE Robotics and Automation Letters*, Vol. 6, No. 3, pp. 5992–5999, 2021.
- [221] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntschi, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland. Toward Anthropomorphic Robotics: Development, Simulation, and Control of a Musculoskeletal Torso. *Artificial Life*, Vol. 19, No. 1, pp. 171–193, 2013.

- [222] S. Nishide, J. Tani, T. Takahashi, H. G. Okuno, and T. Ogata. Tool-Body Assimilation of Humanoid Robot Using a Neurodynamical System. *IEEE Transactions on Autonomous Mental Development*, Vol. 4, No. 2, pp. 139–149, 2012.
- [223] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards Associative Skill Memories. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 309–315, 2012.
- [224] H. Girgin and E. Ugur. Associative Skill Memory Models. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6043–6048, 2018.
- [225] A. Sasagawa, S. Sakaino, and T. Tsuji. Motion Generation Using Bilateral Control-Based Imitation Learning With Autoregressive Learning. *IEEE Access*, Vol. 9, pp. 20508–20520, 2021.
- [226] M. Jäntschi, C. Schmalzer, S. Wittmeier, K. Dalamagkidis, and A. Knoll. A scalable joint-space controller for musculoskeletal robots with spherical joints. In *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2211–2216, 2011.
- [227] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.
- [228] C. Galindo, J. Fernández-Madriral, J. González, and A. Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, Vol. 56, No. 11, pp. 955–966, 2008.
- [229] K. Kawaharazuka, K. Tsuzuki, M. Onitsuka, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Stable Tool-Use with Flexible Musculoskeletal Hands by Learning the Predictive Model of Sensor State Transition. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation*, pp. 4572–4578, 2020.
- [230] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, Vol. 7, No. 1-2, pp. 1–179, 2018.
- [231] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pp. 5628–5635, 2018.

- [232] A. Sasagawa, K. Fujimoto, S. Sakaino, and T. Tsuji. Imitation Learning Based on Bilateral Control for Human–Robot Cooperation. *IEEE Robotics and Automation Letters*, Vol. 5, No. 4, pp. 6169–6176, 2020.
- [233] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-Shot Visual Imitation Learning via Meta-Learning. In *Proceedings of the 2017 Conference on Robot Learning*, pp. 357–368, 2017.
- [234] K. Kawaharazuka, Y. Kawamura, K. Okada, and M. Inaba. Imitation Learning with Additional Constraints on Motion Style using Parametric Bias. *IEEE Robotics and Automation Letters*, Vol. 6, No. 3, pp. 5897–5904, 2021.
- [235] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp. 3968–3974, 2009.
- [236] C. A. V. Perico, J. D. Schutter, and E. Aertbeliën. Combining Imitation Learning With Constraint-Based Task Specification and Control. *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, pp. 1892–1899, 2019.
- [237] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Proceedings of the 2017 Neural Information Processing Systems*, pp. 3812–3822, 2017.
- [238] P. Henderson, W. Chang, P. L. Bacon, D. Meger, J. Pineau, and D. Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.
- [239] L. Chen, R. Paleja, M. Ghuy, and M. Gombolay. Joint Goal and Strategy Inference across Heterogeneous Demonstrators via Reward Network Distillation. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 659–668, 2020.
- [240] K. Aberman, Y. Weng, D. Lischinski, D. Cohen-Or, and B. Chen. Unpaired Motion Style Transfer from Video to Animation. *ACM Transactions on Graphics*, Vol. 39, No. 4, p. 64, 2020.
- [241] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

- [242] A. Kochan. Shadow delivers first hand. *Industrial Robot*, Vol. 32, No. 1, pp. 15–16, 2005.
- [243] Y. Kim, Y. Lee, J. Kim, J. Lee, K. Park, K. Roh, and J. Choi. RoboRay hand: A highly backdrivable robotic hand with sensorless contact force measurements. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, pp. 6712–6718, 2014.
- [244] R. Deimel and O. Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *The International Journal of Robotics Research*, Vol. 35, No. 1–3, pp. 161–185, 2016.
- [245] Z. Xu and E. Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation*, pp. 3485–3492, 2016.
- [246] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal. Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1960–1966, 2016.
- [247] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov. Learning Deep Visuomotor Policies for Dexterous Hand Manipulation. In *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 3636–3643, 2019.
- [248] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus. Robust proprioceptive grasping with a soft robot hand. *Autonomous Robots*, Vol. 43, No. 3, pp. 681–696, 2019.
- [249] S. Zhong, J. Chen, X. Niu, H. Fu, and H. Qiao. Reducing Redundancy of Musculoskeletal Robot with Convex Hull Vertexes Selection. *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2019.
- [250] C. Alessandro, I. Delis, F. Nori, S. Panzeri, and B. Berret. Muscle synergies in neuroscience and robotics: from input-space to task-space perspectives. *Frontiers in Computational Neuroscience*, Vol. 7, No. 43, pp. 1–16, 2013.
- [251] Z. Xie, L. Jin, X. Luo, S. Li, and X. Xiao. A Data-Driven Cyclic-Motion Generation Scheme for Kinematic Control of Redundant Manipulators. *IEEE Transactions on Control Systems Technology*, Vol. 29, No. 1, pp. 53–63, 2021.

- [252] Y. Huang, D. Büchler, O. Koç, B. Schölkopf, and J. Peters. Jointly learning trajectory generation and hitting point prediction in robot table tennis. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 650–655, 2016.
- [253] D. Park, Y. Hoshi, and C. C. Kemp. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 1544–1551, 2018.
- [254] E. Principi, D. Rossetti, S. Squartini, and F. Piazza. Unsupervised electric motor fault detection by using deep autoencoders. *IEEE/CAA Journal of Automatica Sinica*, Vol. 6, No. 2, pp. 441–451, 2019.
- [255] S. Nakashima, T. Shirai, K. Kawaharazuka, Y. A. Y. Kakiuchi, K. Okada, and M. Inaba. An Approach of Facilitated Investigation of Active Self-healing Tension Transmission System Oriented for Legged Robots. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2567–2572, 2019.
- [256] T. Kozuki, Y. Motegi, T. Shirai, Y. Asano, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Design of upper limb by adhesion of muscles and bones - Detail human mimetic musculoskeletal humanoid kenshiro. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 935–940, 2013.
- [257] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.
- [258] I. Mizuuchi, Y. Nakanishi, Y. Namiki, T. Nishino, J. Urata, M. Inaba, T. Yoshikai, and Y. Sodeyama. Realization of Standing of the Musculoskeletal Humanoid Kotaro by Reinforcing Muscles. In *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots*, pp. 176–181, 2006.
- [259] Z. Li and D. Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 12, pp. 2935–2947, 2018.
- [260] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. v. d. Weijer. Class-incremental learning: survey and performance evaluation, 2020.

- [261] K. Kawaharazuka, N. Hiraoka, Y. Koga, M. Nishiura, Y. Omura, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Online Learning of Danger Avoidance for Complex Structures of Musculoskeletal Humanoids and Its Applications. In *Proceedings of the 2020 IEEE-RAS International Conference on Humanoid Robots*, pp. 349–355, 2021.
- [262] O. Kanoun, F. Lamiroux, P. Wieber, F. Kanehiro, E. Yoshida, and J. Laumond. Prioritizing linear equality and inequality systems: Application to local motion planning for redundant robots. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp. 2939–2944, 2009.
- [263] B. Sofman, E. Lin, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, Vol. 23, No. 11-12, pp. 1059–1075, 2006.
- [264] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2849–2854, 2004.
- [265] S. Murata, J. Namikawa, H. Arie, S. Sugano, and J. Tani. Learning to Reproduce Fluctuating Time Series by Inferring Their Time-Dependent Stochastic Properties: Application in Robot Learning Via Tutoring. *IEEE Transactions on Autonomous Mental Development*, Vol. 5, No. 4, pp. 298–310, 2013.
- [266] Y. Duan, X. Chen, R. Houthoofd, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1329–1338, 2016.
- [267] K. Kawaharazuka, M. Nishiura, Y. Koga, Y. Omura, Y. Toshimitsu, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Automatic Grouping of Redundant Sensors and Actuators Using Functional and Spatial Connections: Application to Muscle Grouping for Musculoskeletal Humanoids. *IEEE Robotics and Automation Letters*, Vol. 6, No. 2, pp. 1981–1988, 2021.
- [268] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, K. Okada, and M. Inaba. A Method of Joint Angle Estimation Using Only Relative Changes in Muscle Lengths for Tendon-driven Humanoids with Complex Musculoskeletal Structures. In *Proceedings of the 2018 IEEE-RAS International Conference on Humanoid Robots*, pp. 1128–1135, 2018.

- [269] H. Nagamochi and T. Ibaraki. Graph connectivity and its augmentation: applications of MA orderings. *Discrete Applied Mathematics*, Vol. 123, No. 1, pp. 447–472, 2002.
- [270] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, Vol. 7, No. 1, pp. 48–50, 1956.
- [271] K. Kawaharazuka, K. Tsuzuki, S. Makino, M. Onitsuka, K. Shinjo, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba. Task-specific Self-body Controller Acquisition by Musculoskeletal Humanoids: Application to Pedal Control in Autonomous Driving. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 813–818, 2019.
- [272] A. Byravan, F. Leeb, F. Meier, and D. Fox. SE3-Pose-Nets: Structured Deep Dynamics Models for Visuomotor Planning and Control. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pp. 3339–3346, 2018.
- [273] T. v. d. Sande, I. Besselink, and H. Nijmeijer. Steer-by-wire: a study into the bandwidth and force requirements. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, pp. 1–6, 2012.
- [274] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Höppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wüsthoff, and G. Hirzinger. The DLR hand arm system. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pp. 3175–3182, 2011.
- [275] T. Wiste and M. Goldfarb. Design of a simplified compliant anthropomorphic robot hand. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation*, pp. 3433–3438, 2017.
- [276] G. P. Kontoudis, M. V. Liarokapis, A. G. Zisimatos, C. I. Mavrogiannis, and K. J. Kyriakopoulos. Open-source, anthropomorphic, underactuated robot hands with a selectively lockable differential mechanism: Towards affordable prostheses. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5857–5862, 2015.
- [277] P. K. Allen, A. T. Miller, P. Y. Oh, and B. S. Leibowitz. Using tactile and visual sensing with a robotic hand. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 676–681, 1997.

- [278] A. Bicchi, J. K. Salisbury, and P. Dario. Augmentation of grasp robustness using intrinsic tactile sensing. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 302–307, 1989.
- [279] M. Regoli, U. Pattacini, G. Metta, and L. Natale. Hierarchical grasp controller using tactile feedback. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 387–394, 2016.
- [280] A. J. Schmid, N. Gorges, D. Goger, and H. Worn. Opening a door with a humanoid robot using multi-sensory tactile feedback. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pp. 285–291, 2008.
- [281] F. R. Hogan, M. Bauza, O. Canal, E. Donlon, and A. Rodriguez. Tactile Regrasp: Grasp Adjustments via Simulated Tactile Transformations. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2963–2970, 2018.
- [282] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine. More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch. *IEEE Robotics and Automation Letters*, Vol. 3, No. 4, pp. 3300–3307, 2018.
- [283] Y. Li, D. Xu, Y. Yue, Y. Wang, S. Chang, E. Grinspun, and P. K. Allen. Regrasping and unfolding of garments using predictive thin shell modeling. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, pp. 1382–1388, 2015.
- [284] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima. Dynamic Manipulation of Flexible Objects with Torque Sequence Using a Deep Neural Network. In *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 2139–2145, 2019.
- [285] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 2008.
- [286] T. Kurotobi, T. Shirai, Y. Motegi, Y. Nakanishi, K. Okada, and M. Inaba. Controlling tendon driven humanoids with a wearable device with Direct-Mapping Method. In *Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication*, pp. 437–442, 2012.

- [287] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards fully autonomous driving: Systems and algorithms. In *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–168, 2011.
- [288] M. R. Endsley. Autonomous Driving Systems: A Preliminary Naturalistic Study of the Tesla Model S. *Journal of Cognitive Engineering and Decision Making*, Vol. 11, No. 3, pp. 225–238, 2017.
- [289] C. Rasmussen, K. Sohn, Q. Wang, and P. Oh. Perception and control strategies for driving utility vehicles with a humanoid robot. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 973–980, 2014.
- [290] A. Paolillo, P. Gergondet, A. Cherubini, M. Vendittelli, and A. Kheddar. Autonomous car driving by a humanoid robot. *Journal of Field Robotics*, Vol. 35, No. 2, pp. 169–186, 2018.
- [291] E. Haug, H. Choi, S. Robin, and M. Beaugonin. Human Models for Crash and Impact Simulation. In *Computational Models for the Human Body*, Vol. 12 of *Handbook of Numerical Analysis*, pp. 231–452. Elsevier, 2004.
- [292] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, 2018.
- [293] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke. Child-sized 3D printed igus humanoid open platform. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 33–40, 2015.
- [294] C. T. Kiang, A. Spowage, and C. K. Yoong. Review of Control and Sensor System of Flexible Manipulator. *Journal of Intelligent & Robotic Systems*, Vol. 77, No. 1, pp. 187–213, 2015.
- [295] M. Inaba and H. Inoue. Rope handling by a robot with visual feedback. *Advanced Robotics*, Vol. 2, No. 1, pp. 39–54, 1987.
- [296] M. Saha and P. Isto. Manipulation Planning for Deformable Linear Objects. *IEEE Transactions on Robotics*, Vol. 23, No. 6, pp. 1141–1150, 2007.

- [297] C. Elbrechter, R. Haschke, and H. Ritter. Folding paper with anthropomorphic robot hands using real-time physics-based modeling. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 210–215, 2012.
- [298] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, pp. 177–184, 2015.
- [299] Y. Yamakawa, A. Namiki, and M. Ishikawa. Motion planning for dynamic folding of a cloth with two high-speed robot hands and two high-speed sliders. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pp. 5486–5491, 2011.
- [300] R. Jangir, G. Alenyä, and C. Torras. Dynamic Cloth Manipulation with Deep Reinforcement Learning. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation*, pp. 4630–4636, 2020.
- [301] T. Ogata, H. Ohba, J. Tani, K. Komatani, and H. G. Okuno. Extracting multi-modal dynamics of objects using RNNPB. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 966–971, 2005.
- [302] Y. Asano, S. Nakashima, I. Yanokura, M. Onitsuka, K. Kawaharazuka, K. Tsuzuki, Y. Koga, Y. Omura, K. Okada, and M. Inaba. Ankle-Hip-Stepping Stabilizer on Tendon-Driven Humanoid Kengoro by Integration of Muscle-Joint-Work Space Controllers for Knee-Stretched Humanoid Balance. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots*, pp. 397–402, 2019.
- [303] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, Vol. 4, No. 26, 2019.
- [304] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik’s Cube with a Robot Hand. arXiv preprint arXiv:1910.07113, 2019.

以上

1p~ 562p 完

博士論文

令和3年12月3日提出

東京大学大学院 情報理工学系研究科
知能機械情報学専攻
48-197506 河原塚 健人