東京大学
情報理工学系研究科 創造情報学専攻
博士論文

認識計画実行機能の評価制御機構を備えた等身大ヒューマノイド統合システムの研究

東京大学大学院 情報理工学系研究科
創造情報学専攻 博士課程
植田 亮平

指導教員　稲葉 雅幸 教授

2015 年 12 月

概要

ロボットシステムには要求される時間や安全性, 利用可能な計算機資源に従って自動的に品質を調整する機能が必要不可欠である. というのも, ロボットはタスク遂行に時間をかけられる場合は, 精度が高く成功率の高い認識や動作計画, 動作実行を慎重に行うことが可能である. 逆にタスク遂行に時間をかけられない場合は精度が粗く成功率の低い認識や動作計画を利用することが求められる. ロボットシステムの研究開発では, このような品質と時間の関係はシステム実装者によって注意深く調整されてきた. 本論文はこのようなタスク遂行に必要とされる要求時間や成功率にしたがってシステムが自動的にパラメータやアルゴリズムをスケジューリング可能なシステムの構築を目指し, 認識計画実行機能の評価制御が可能なロボットシステムの構成法について論じる.

本論文ではこれらの問題を解決するためのベースとなるシステムとして, ヒューマノイドの遠隔操縦システムの構成法について示し, 災害対応ロボット競技会の結果を通じて求められる要求時間に対してシステムの振る舞いをカスタマイズ可能な統合システムの必要性を示す. 認識器と動作計画器の協調的サブスケジューリングによる足配置計画法を提案し, 計画前の認識によるモデル化コストを削減した高速化手法について述べる. 継続的認識機能としてパーティクルフィルタを用いた 3 次元点群追跡法, およびそれを用いた環境記述の監視機能によって環境の動的変化への即時性の向上を示す. これらの要素技術を利用することによって, 移動を伴うマニピュレーションタスクである災害対応ロボット競技会のタスクを題材に実験を行い, タスク必要時間に従ってパラメータ可変な認識計画実行を行うシステムが構成可能であることを示す.

ロボットの認識計画実行機能を評価制御する機構を備えるというシステム構成法により, 求められる要求度緊急度が異なるタスクに対応する基盤となるシステムを構成可能であることを示し, 人間がその機構を利用して要求に応じたシステムをカスタマイズできる構成法とするだけでなく, システム自体がシステムの運用履歴を通してそのカスタマイズの方法を導入することで適応的・学習的に要求タスクへ対応するロボットシステムの可能性を示したものとなっている.

目次

第 1 章

序論

1.1　研究の背景と目的

ロボットはハードウェアとして計算機に環境を認識するためのセンサ, 環境と相互作用するためのアクチュエータを付与した実世界システムである. 人が行なっている作業をロボットによって代替することで, 人が危険な環境で作業する必要がなくなったり, より知的な生産活動に時間を割くことができると期待されている. 自動車の溶接や塗装, 宇宙空間での船外活動といった人が行うには大きな危険を含む単純な作業に関してはすでに実用化され, 産業の一端を担うまでになっている.

ヒューマノイドロボットはロボットの中でも特殊であり, 人に類似した関節自由度とリンク構成, 頭部に搭載するカメラなどの人と類似したセンサ構成を持つロボットである. ヒューマノイドロボットに期待される作業は他のロボット同様に人が行なっている作業の代替であるが, より高度な作業を行うことが期待されている. ヒューマノイドロボットは人に類似した関節自由度, リンク構成を持つことから, 普段人が生活している生活環境にヒューマノイドロボットが入り, 家事などの作業を人に代わって行う家事支援行動はヒューマノイドロボットに期待されるアプリケーションとしてその特徴を生かしたものである. さらに近年では, 災害対応ロボット競技会 [1] が開かれたことからも顕著なように, 災害現場という人の侵入が難しい・危険が伴うような環境であるが人が活動することを前提として構築された環境でのヒューマノイドロボットの活躍も期待されている. このような環境においてタスクを遂行するには高度な認識機能・動作計画機能が必要とされる. というのも, ロボットに有利なように設計された特殊な環境であったり, 既知の変化しない環境ではなく, ヒューマノイドロボットには様々な環境での活動が期待されるからである.

しかしながら, 人が普段行なっているタスクをヒューマノイドロボットに行わせると, 人と

ヒューマノイドロボットではタスク実現のために必要となる時間が大きく異なる．人は環境に応じて即時に動作を修正しつつ最低限の動作で目的を達成したり，環境の認識や複数の動作を同時に特に意識することなく実現できている．その一方でこのような挙動を可能とさせるロボットシステムの構築は難しい．これはロボットの認識・動作計画・動作実行といった基本的な行動が停止時間少なく並列にシステムが駆動していくことが難しく，さらに人が認識や動作を計画するのに必要な時間と同程度の性能を実現する計算機性能および計算アルゴリズムの実現が難しいからである．

このような課題を解決するためには，システム実装者が注意深く環境やタスクに最適なアルゴリズムおよびパラメータを設計する必要がある．しかしながら，そのようなアプローチで構築されたロボットシステムでは，様々な環境に適応するための環境変化に応じて経験からパラメータを学習できるような柔軟性を持たせることが難しい．また，環境が同一であってもロボットがタスク遂行に必要とされる時間や安全性に応じて柔軟にシステムを可変させていくような仕組みを提供することも難しい．というのも，このようなロボットシステム全体の実行速度は，要求される時間や安全性，利用可能な計算機資源に従ってその都度決定されるべきものだからである．タスク遂行のために要求される時間が十分であり，安全性が必要な場合は時間をかけて認識し，実行することが求められる．その一方で，要求される時間が短い場合は，安全性や精度を下げることで素早く実行することが求められる．

このようにロボットシステムには要求される時間や安全性，利用可能な計算機資源に従って自動的に精度や認識領域を調整する機能が必要不可欠である．本論文ではこのようなシステム全体の必要時間を人間から与えられた要求度に従って決定するため，認識や動作計画，動作実行の時間に関するパラメータを制御可能な評価制御機構を備えたロボットシステムの構成法を主題に論じていく．本論文で論じる評価制御機構を備えたロボットシステムは，人間から与えられた要求度に従って認識，動作計画，動作実行のパラメータを可変させるだけではなく，タスク遂行経験に従ってシステム自身の挙動を柔軟に変化させるための基礎となるシステム構成である．

1.2　関連研究と研究の特色

1.2.1　ロボットシステムのスケジューリング研究

ロボットシステムの構築において高次のタスク計画の重要性が指摘されており，多くの研究がなされている．**Fig.** 1.1 にこれらの研究が問題として扱っている領域を図示し，整理する．

本論文では一連のタスクをプロジェクトと呼ぶ.

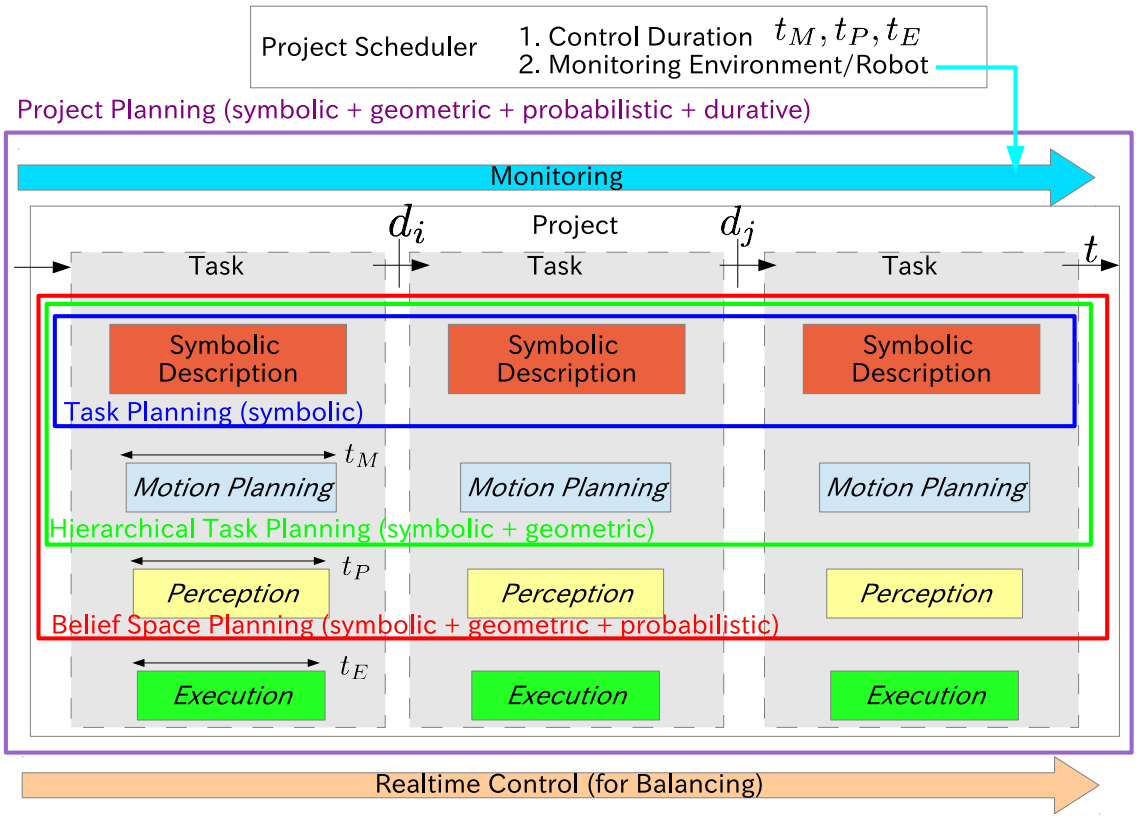


Fig1.1: Project Scheduling of Robot System

古典的 AI として知られるシンボリックで離散的な記述に従ってロボットの行動手順を計画する手法はタスクプランニング, タスク計画や行動計画と呼ばれる. ロボットや環境は連続的な幾何的な表現によって記述されるが, これらに対し抽象化された離散的なシンボリックな記述を導入することで各タスクの実行順序を計画する. シンボリックな記述としては STRIPS[2] による記述が代表的である. 各種タスクが実行前に必要とする前提条件およびタスク実行によって環境やロボットに影響を与える副作用をシンボリックな条件群として記述することで, 初期状態から目標状態までのタスク列を求めることをタスクプランニングと呼ぶ. タスクプランニングによって計画されたタスク列には認識や動作計画をどのような順序で実行していくかは考慮されていない. したがって, 各タスク内での認識や計画, 実行のタイミングは予めシステム実装者によって定義されたものを利用する.

このような古典的な AI をベースとするタスクプランニングをロボットシステム上位のスケジューリング機能として利用する場合, 離散的なシンボリックな記述と連続的な幾何的な記述の間に隔たりが生じてしまうという問題点が知られている [3]. シンボリックな記述と幾何

的な記述の間の隔たりは必要のないタスク手順を発生させてしまう可能性がある．例えば, 対象物の目の前に移動するタスク `moveTo(obj)` とそれに手を伸ばすマニピュレーションタスク `reach(obj)` を実行する必要があるとき, 物体 `obj` にロボットの腕が届く範囲をシンボリックな表現によって必要十分に記述しきることは難しい．そのため, 実際には物体 `obj` に腕が届く場所にロボットがいたとしても, 不必要な行動である `moveTo(obj)` を実行してしまうといった状況が発生する．このような状況を回避するためには, タスクプランニングの中に幾何的な記述を利用した推論機能を組み込む必要がある．Kaelbling らは Hierarchical Task Planning と呼ばれる階層的なアプローチを利用することでタスクプランニングと幾何的な記述レベルでの推論機能の統合を実現している [4]. Cambon らは, シンボリックなプランナを幾何的な記述レベルにおける動作計画器に対して拘束条件とヒューリスティック関数をあたえると位置づけることで離散的であるシンボリックな計画器と連続的である幾何的な計画器との統合を行なっている [5].

動作計画と同様に認識処理もタスクプランニングでは固定的な手順としてタスクに組み込まれている．認識処理もタスクプランニングに統合する手法として, Belief Space におけるタスクプランニングが提案されている [6]. Belief Space におけるプランニングでは, シンボリックな述語表現に確率的な表現を導入する．例えば, 物体 `A` が物体 `B` の上に存在するという記述である `on(A, B)` は `on(A, B, 0.85)` というように確からしさを用いて表現される．また, ロボットが移動するに従ってロボット自身の自己位置の確からしさが下がるようなモデルを導入する．確率的な記述を導入することで認識機能は確からしさを決定するための機能と位置づけることが可能となり , 認識機能をタスクプランニングに統合している．

シンボリックな空間でのグラフ探索によるロボットの行動手順計画とは異なり , 数理計画法によるスケジューリング問題を解くことでロボットの行動手順を求めるアプローチが提案されている．一般のスケジューリング問題においては, 数理計画法によるアプローチが有効なことが知られており , 工場などにおける生産計画として利用されている．高次のロボットシステムのスケジューリング機能としてタスクプランニングを導入することによる利点は複雑な環境やロボットの状態の依存関係を解決することが可能な点であった．このような依存関係が少ないようなプロジェクトにおいては, 数理計画法によるロボットシステムのスケジューリングによって与えられたデッドラインを満たしながら行動手順を効果的に計画可能である．Cotlin らは [7] 移動ロボットのタスクスケジューリングを混合整数線形計画問題として最適化することで 40 程度のタスクからなるプロジェクトのスケジューリングをデッドラインを満たしながら

実行できることを示した. しかしながら Cotlin らの手法では並列に実行可能なタスクに関しては考慮されていなかった. Mudrova らは区間代数 (Interval Algebra) を用いることでタスク間の並列性に関する依存関係を記述し, 混合整数計画問題として最適化する手法を示している [8].

これらの行動手順の計画では, 認識や計画にどの程度の時間をかけるかという点に関しては考慮されていなかった. 比留川らはロボットの自己干渉チェックを実時間で実現するため, 自己干渉チェックの形状モデルの詳細度と必要となる計算時間の関係を利用した実時間計算アルゴリズムをその先駆的な研究で示している [9]. これは計算品質と計算時間の関係を利用したインプリサイス計算によるリアルタイムシステムのスケジューリング手法をロボットシステムに適用したものと位置づけることができる [10, 11]. 本論文ではインプリサイス計算を用いたスケジューリングの考え方を利用することで, タスク中の認識・動作計画・実行にどの程度の時間を費やすかの計画を行う.

View Planning, Viewpoint Planning, Active Vision と呼ばれるアプローチでは, 認識と計画, 実行のフィードバックを繰り返すことによって認識の品質を向上させる手法が研究されている. Sommerlade らは複数のカメラから得られる情報量を最大化するようにカメラ位置を制御することで観測に有利なカメラ位置を決定する手法を示した [12]. Sanchiz らは next-best-view と呼ばれるアルゴリズムによって, 部屋全体の 3 次元点群を効率的に構築する手法を提案している [13]. これらの手法は効率的に認識の品質を向上させるために利用されるが, 目標となるタスクは認識そのものであることが多く, マニピュレーションのようなより上位タスクの目標に対して利用することが難しい.

複数のタスクを離散的に実行していくのではなく, 連続的にタスクを接続することで全体で必要となる時間を削減し, 複数のタスクを同時に実行する手法が提案されている. Keith らは 2 脚ロボット HRP-22 を用いて冷蔵庫から缶を取り出すタスクの並列化に関する研究 [14] や, 歩行しながら物体を把持する研究 [15] を行なっている. これらでは Stack of Tasks[16] と呼ばれる構造によってタスクが記述されている. ここでタスクという単語は逆運動学計算を意味する. Stack of Tasks ではロボットが実行すべきタスクの優先度を実行時に動的に切り替えることで, 全動作の整合性を保ちつつ動作をなめらかに接続し, 可能であれば並列に実行することが可能になっている. Suzuki らはタスクの達成時刻も含めた最適化を行うことで, 複数のタスクにおいて達成順序を含めたスケジューリング手法を示している [17]. しかしながら, これらの研究は軌道の最適化という問題点へのアプローチであり, 計算に時間がかかるような動作計

画を実行時にどのようにスケジューリングするかという点に関しては言及されていない.

1.2.2　ロボットシステム研究

古典的 AI をベースとしたロボットシステム

古典的 AI という言葉が意味する範囲は広いが, シンボリックな記述に基づくプランベースなシステムと解釈することができる. シンボリックな記述とは, 人や対象物, 場所などの名前を基本とし, 知識ベースをそれらの関係性として記述するものである. 例えばある物体 `obj-a` がある棚 `shelf-a` の中に存在する場合, その関係性は `(in obj-a shelf-a)` のように記述される. 古典的 AI はこのようなシンボリックな知識ベースにしたがって, 推論を行うシステムである. 推論は目標状態および初期状態をシンボリックに記述し, 初期状態から目標状態まで至るための遷移, すなわち行動列を推定することである. 推論には一階述語論理 [2, 18] や, 論理型プログラミング言語 Prolog などが利用される [19, 20]. ルールベースな記述を利用して行動を決定していくシステムは Blackboard System と呼ばれる.

このようなシンボリックな記述はロボットシステムにおいては高次の計画システムとして利用される. シンボリックにタスクを記述する必要がある古典的 AI の問題点として, タスクの記述が膨大になってしまうこと, さらに問題規模が大きくなると組み合わせ数が発散してしまうことが指摘されている. 前者の問題の解決に向けて, cyc プロジェクト [21] が膨大な知識ベースを構築している. Beetz らは cyc プロジェクトによる知識ベースを利用可能なアーキテクチャである CRAM[22] および知識ベース Knowrob[23] を開発している.

リアルタイム AI のためのサブサンプションアーキテクチャ

古典的 AI をベースとしたプランベースなシステムは, ロボットが動作をするためにはプランニングをしなくてはいけないためリアルタイム性が低いことが問題であった. これを解決するためのアプローチとして, サブサンプションアーキテクチャ [24] が広く知られている. サブサンプションアーキテクチャは意思決定を軽量・並列なモジュールを組み合わせることでシステムを構築しようというアーキテクチャである. サブサンプションアーキテクチャは iRobot 社[*1]の製品である掃除ロボット Roomba に搭載されていることでも知られている.

プランベースなシステムでは上位のタスク表現から順に動作計画・認識を階層的に行なっていくことで最終的なアクチュエータ指令値を決定していく. 一方サブサンプションアーキテ

[*1] http://www.irobot.com/

クチャでは軽量な意思決定モジュールがセンサ入力を受け取り, アクチュエータ目標値を出力する (**Fig.** 1.2). 軽量な意思決定モジュールはひとつだけではなく, システム中で複数存在する. これらのモジュールは並列に駆動し, 下位層は上位層の出力よりも優先される. 下位層には「バンパセンサが環境に衝突したら停止する」,「赤外線センサが障害物に近づきすぎていることを検知したら進行方向を変化させる」といった緊急性の高いモジュールが配置される. **Fig.** 1.2 においては, Level 0, Level 1 と記している四角形が意思決定モジュールである.

サブサンプションアーキテクチャは軽量なモジュールが並列に実行されるシステムのため, 環境の変化に対する反応が早く, 動的な環境に素早く対応可能な点が大きな長所であるシステム構成となっている. その一方で古典的 AI をベースとしたプランベースなシステムが得意とした手順が必要となるような高度なタスクに関してはモジュール数が多くなりシステム構築が難しくなってしまう. また, タスクの複雑さが上がりコストの高い認識が必要となってしまうとモジュールの処理が重くなり環境に素早く対応可能な長所が失われてしまう. 手順が必要となるような高度なタスクに関し Logic-Based Subsumption Architecture[25] と呼ばれるサブサンプションアーキテクチャの拡張アーキテクチャが提案されている. Logic-Based Subsumption Architecture では各モジュールがセンサ入力に加え, 上位レイヤの出力目標値を入力として受け取る. 各モジュールはこれらの入力を一階述語論理で記述されたルールにしたがって推論を行なっていくものである. これによってサブサンプションアーキテクチャに推論機能を統合可能となっている. しかしながら古典的 AI の短所をそのまま引き継いでおり, 上位のモジュールは動的な環境への素早い対応が難しい.

テレオペレーションと Supervised Autonomy

ロボットシステムを構築する上で, 自律的なロボットシステムを構成するという方針ではなく, ロボットを遠隔操縦することによってタスクを遂行させるというアプローチが提案されている [26]. 遠隔操縦によってタスクを遂行できることで, 人が入っていくと危険な環境や遠隔地におけるタスクをこなすことが可能である. 遠隔操縦はテレオペレーションと呼ばれ, 操縦者であるオペレータがセンサ値を観測し, 意思決定・認識・動作計画に該当する機能を担当する. オペレータはアクチュエータの目標値を出力し, ロボットはその目標値を実行する. このようにフィードバックループの中に常にオペレータが介在するテレオペレーションモデルを Human-in-the-Loop と呼ぶ. Human-in-the-Loop モデルの利点は, 多くの高度な機能を実装することなくオペレータが担うことができるためロボットは様々なタスクを実現することができる点である. しかしながら, オペレータへの負担が大きい, ロボットとオペレータの間の通信

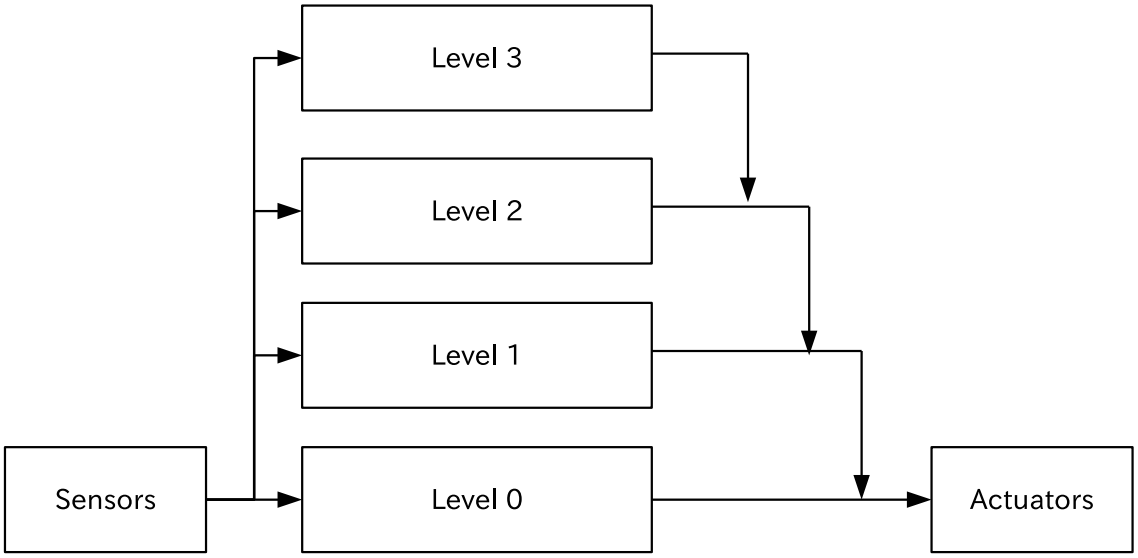


Fig1.2: Subsumption Architecture. From [24].

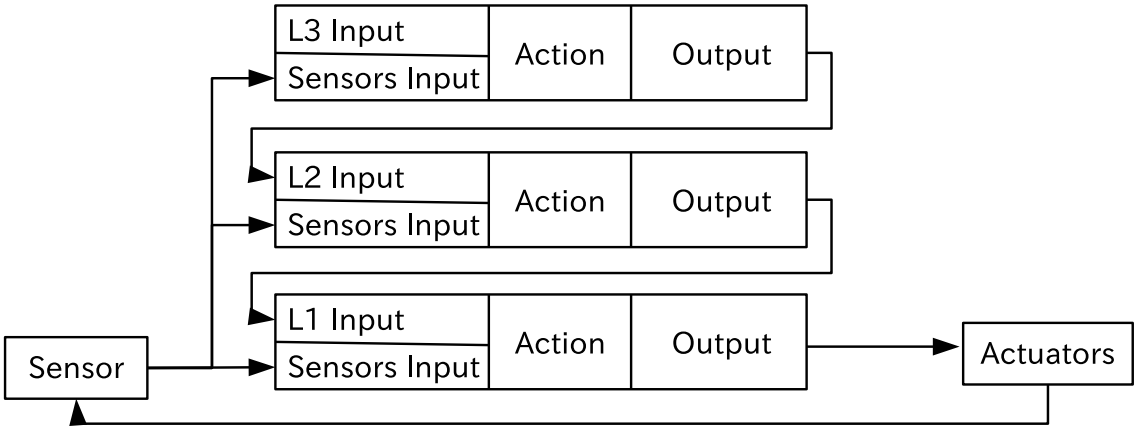


Fig1.3: Logic-Based Subsumption Architecture. From [25].

に遅延や帯域の制限がある場合, システムの動作周期が落ちてしまうといった欠点がある.

Supervised Autonomy[27] と呼ばれるアプローチでは, オペレータはアクチュエータ指令値というよりは動作計画器の目標やよりシンボリックな行動を指示するに留まる. ロボットは独自にフィードバックループを持ち, ロボットはオペレータから指示された目標に対して自律的に動く. このようにオペレータがシステムのフィードバックループの外に位置することで, ロボットとオペレータの間の通信に遅延や帯域の制限がある場合でも遠隔操縦が効果的に実現でき, オペレータの負担が小さい. しかしながら Human-in-the-Loop モデルと比べて動作計画や認識器に必要とされる機能は高度なものとなる.

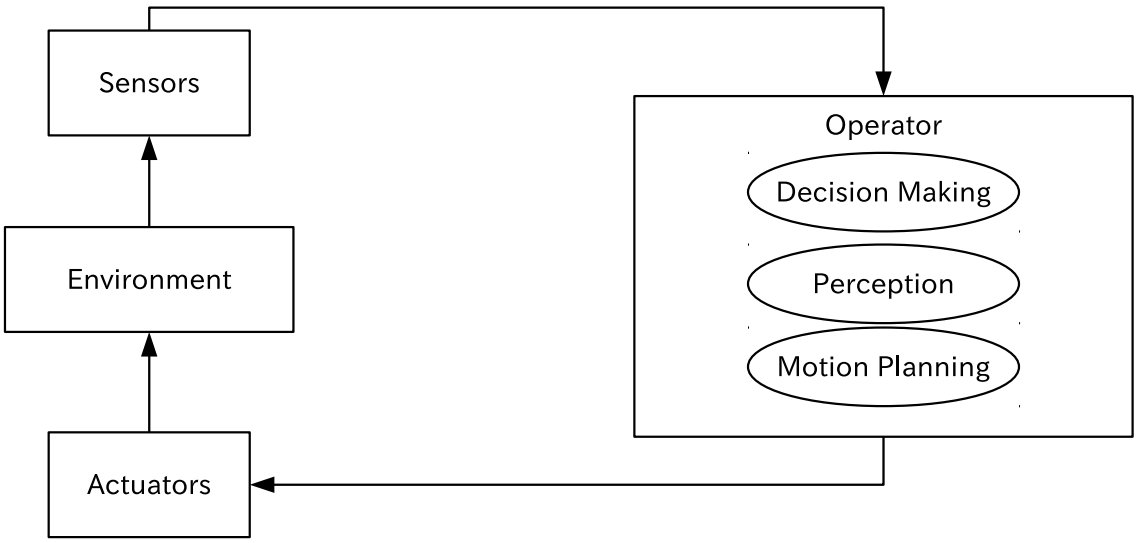


Fig1.4: Teleoperating Robot System: Human-in-the-Loop Strategy.

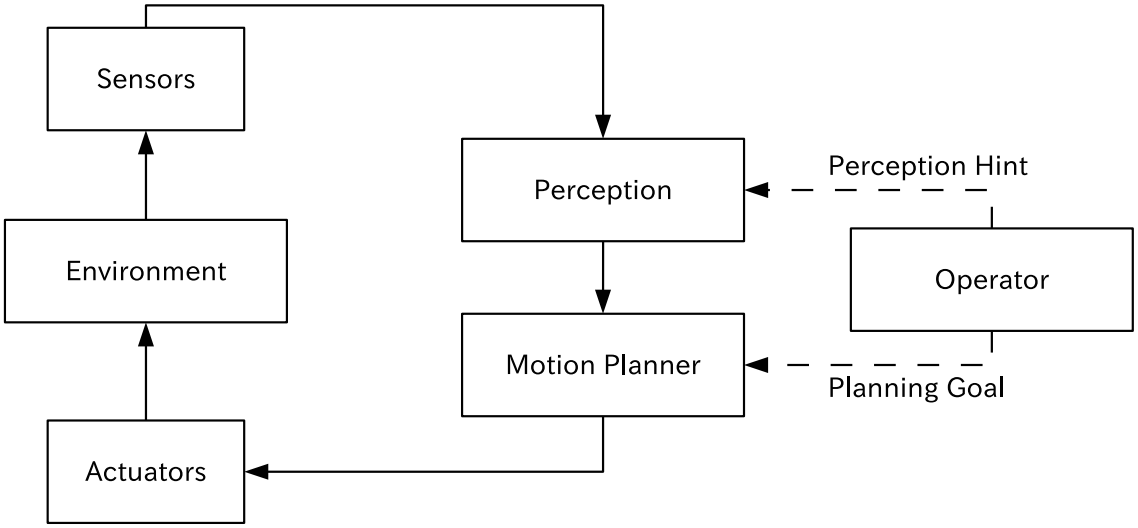


Fig1.5: Teleoperating Robot System: Supervised Autonomy.

SLAM ベースな移動ロボットの経路計画システム

SLAM[28, 29] は環境の地図生成と自己位置推定を同時に解くことでロボットの自己位置同定問題を解決するアプローチである．認識ではレーザレンジファインダを利用することが多く，パーティクルフィルタやカルマンフィルタといった再帰的ベイズ推定によって自己位置を推定すると同時に地図モデルを構築する．SLAM による自己位置推定結果と環境の地図をベースとして動作計画では局所計画と大域計画の 2 つに計画の粒度を分けることで，動的な環境にも

対応可能な計画システムを構築している．大域計画では少ない次元数で長距離の軌道を計画し, 局所計画では大域計画よりも詳細な障害物地図を利用してより短い距離を高速に計画する．動的な環境における突然の障害物は局所計画で素早く対応し, 全体的な進行方向は大域計画によってコントロールする．

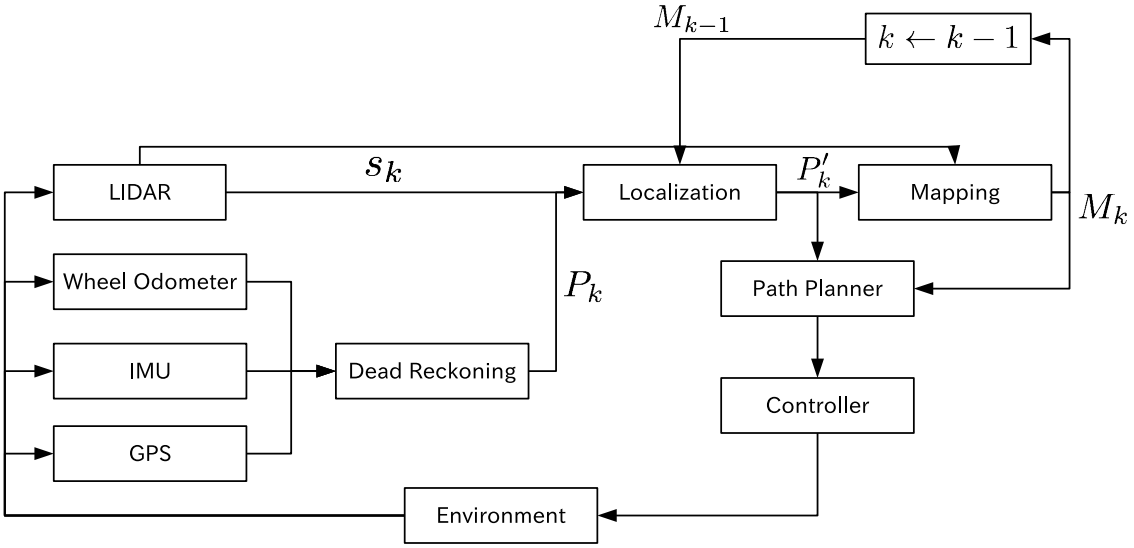


Fig1.6: SLAM Based Robot System

ロボット用分散ミドルウェア

ロボットのソフトウェアにはアクチュエータの制御はもちろん, センサの読み取り, 画像処理, レーザ処理による自己位置推定といった多くの機能が必要とされる．このような複数のモジュールからロボットシステムを構成するため, ロボット用のミドルウェアが多く研究開発されている．Player[30], OpenRTM[31], Tekkotsu[32], Orocos[33], ROS[34], LCM[35] といったものが具体的なミドルウェアとして開発されている．特に近年は ROS の利用が広がっており, ロボット研究の分野においてはデファクトスタンダードとなりつつある．企業が提供する研究用ロボットにおいても企業側がソフトウァサポートの一環として ROS をサポートすることが増えている．ROS を始めとするこれらのミドルウェアでは各種ソフトウェアモジュールをカプセル化して独立性を高め, それらをネットワーク通信によって相互につなぐことでシステムを構成していく．このようなロボット用ミドルウェアの普及によって, 複数のソフトウェアモジュールから構成されるロボットシステムにおいて複数の CPU コア, 複数の計算機を利用することによる並列化の恩恵を容易に受けられるようになった．例えば ROS ではプロセス間通信には出版購買モデルが利用されている (**Fig.** 1.7)．出版購買モデルを利用することで,

センサ入力を起点とするイベント駆動によるシステムが構築され, 並列な関係性のモジュールは複数の CPU コア, 複数の計算機にわたって自動的に並列実行される.

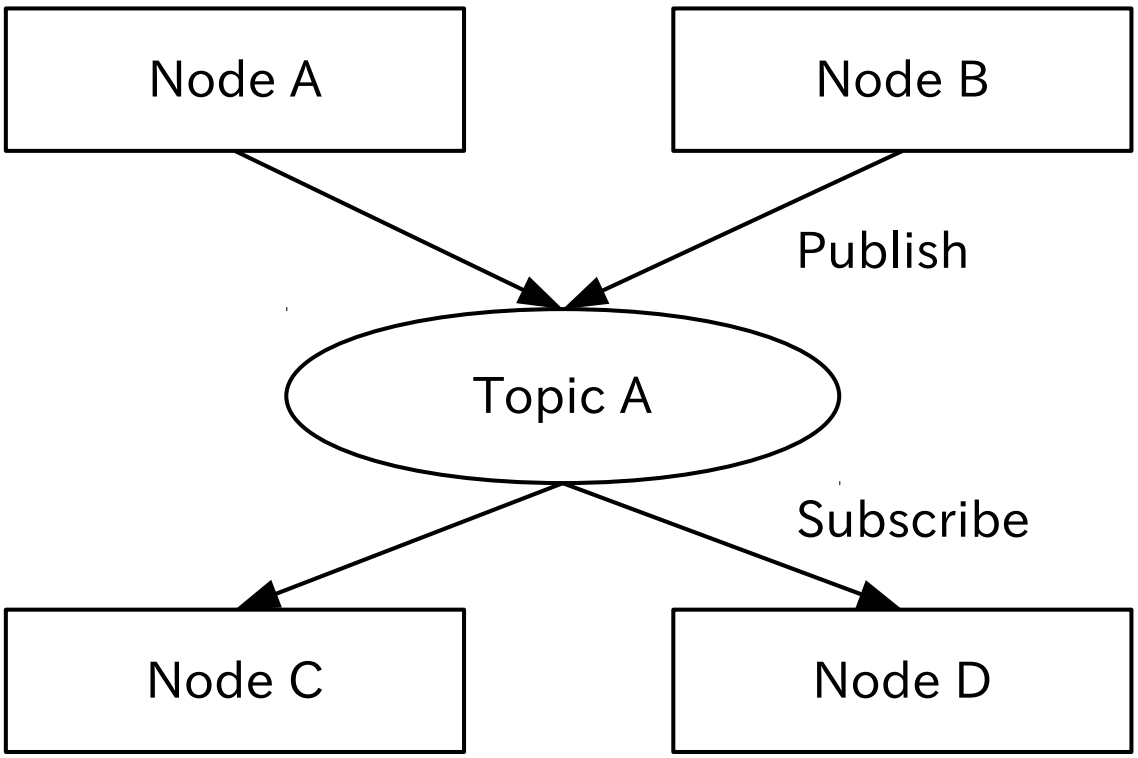


Fig1.7: ROS Communication Model: Publish-Subscribe Pattern

ヒューマノイドのロボットシステム研究

本小節でのべたような議論を背景として, ヒューマノイドのロボットシステム研究が進められている.

小倉らはタスク表現に STRIPS を用い, 高次のタスク計画から動作計画までを行う道具利用マニピュレーション可能な等身大ヒューマノイドの統合システムを構築した [36]. Okada らはこの統合システムに関して画像処理による認識機能を統合した [37, 38]. 統合された認識機能ではマニピュレーション対象の認識に加え, タスクプランニングによって生成された行動の前後で視覚によって行動の前提条件や効果の確認が含まれている. また, Noazawa らはタスク全体の記述はあらかじめ有限要素機械によって記述しておくことで, 実行時に対象物の質量に応じて動作計画や動作手順を変更可能なシステムを構築した [39]. これらは実行時に入手可能なセンサデータを利用することでシンボリックなプランベースなシステムにおいて環境の変化への即時性を向上させた研究であると言える.

一方ヒューマノイドロボットのマニピュレーションにおける大きな特徴は, ロボットが 2 つの腕を持つことである．双腕を利用することで, 複数のタスクを同時に実現することができ, タスク実現の高速化が可能である．Zöllner らは左右の腕の制御器を並列に持つことで, 左右の腕を同時に実行可能なシステムを構築した [40]．Zöllner らのシステムではタスクプランニングの結果を Petri Net によって表現し, タスク表現は人のデモンストレーションから獲得する [41] というものである．Stulp らは複数のタスクを両腕で同時に実現するためにはロボットの立ち位置が重要となることに着目し, Action-Related Places とよばれるタスク依存な立ち位置表現を導入した [42]．Action-Related Places を利用することで移動を含むマニピュレーションタスクの中で上位のシンボリックなプランを最適化することが可能である．

Rusu らは並列実行可能なソフトウェア環境として, 先に述べたた ROS と呼ばれるロボットミドルウェアを利用したシステム構成法を示している [43]．Rusu らのシステムでは, ROS において並行な関係にあるモジュールが並列に実行されることを利用し, 動作計画で必要となる環境モデルを常時構築することで行動実行時に必要となる認識器の計算コストを削減している．

1.2.3　本研究の位置づけ

ロボットの研究開発が進み, ハードウェアの面では信頼性・性能が向上し, ソフトウェアの面ではアルゴリズムの研究開発はもちろん, 計算機の演算性能の向上によってロボットに求められる認識や計画のレベルは高くなっている．これはヒューマノイドのようなロボットのタスク遂行のために用意されたのではない環境での活動を期待されるロボットにおいては顕著である．認識や計画のレベルが高くなるにつれ, 状況に応じてプランナが必要となり, 認識・動作計画・実行のスケジューリングが必要とされる．認識・動作計画・実行のスケジューリングとはロボットが要求された目標を達成するためにどのような手順で行動を実行していくかという行動計画から, それぞれの認識・動作計画・実行にどの程度の時間を費やすかに至るまでを決定する問題である．これはアクチュエータを直接制御するハードリアルタイムな実時間制御層よりも上位に位置し, ソフトリアルタイムなスケジューリングを行う高次の計画システムの構成方法に関する問題である．本論文ではこのような問題を, 一連のタスクに関してどのように時間を配分するかという観点から, プロジェクトスケジューリングと呼ぶ．本論文では認識・動作計画・実行にどの程度の時間を費やすか, またそれらをどのように組み合わせてプロジェクト全体のスケジューリングを行うかという点について論じていく．ロボットの統合システム研

究において, このようなシステムの構成要素にどの程度の時間を割り当てるかまで取り扱った研究はなされていない.

本論文では認識・動作計画・実行にどの程度の時間を費やすかを制御可能なロボットの統合システムを構築するため, 古典的 AI をベースにしたプランベースなシステムに評価制御機構と呼ぶ枠組みを導入する.

1.3　論文の構成

Fig. 1.8 に論文の構成を図示したものを示す. **Fig.** 1.8 は横軸に時間をとり, ロボットシステムが時間の進捗に合わせて認識・動作計画・実行をどのような順序で行なっていくかを図示したものである. P は認識処理, M は動作計画処理, E は動作実行および O は遠隔操縦者の操作を示している. また,t_P, t_M, t_E は認識, 動作計画および動作実行に必要となる時間である.

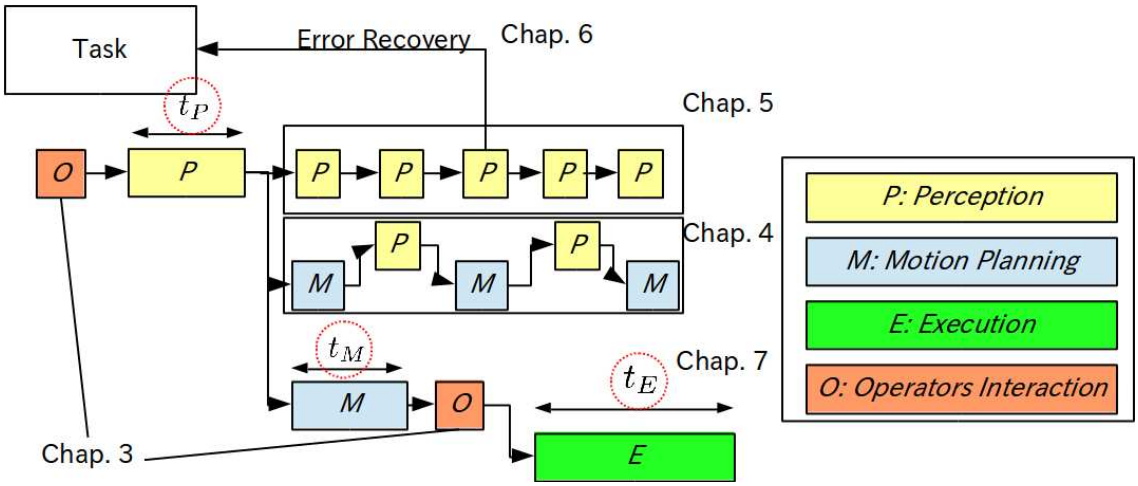


Fig1.8: Overview of Thesis

第 1 章「**序論**」において研究背景としてロボットシステムの研究について述べ, 要求される時間や安全性, 利用可能な計算機資源に従って自動的にシステム自身がパラメータを変化させる機能をロボットシステムに持たせることの重要性について述べた.

第 2 章「**認識計画実行機能の評価制御機構に基づくプロジェクトスケジューリング**」ではロボットシステムにおいて認識や動作計画のパラメータを変更可能な仕組みとして評価制御機構を提案する. 評価制御機構の特徴はタスク実行中の認識や動作計画で精度に影響を与えるパラメータを制御することでタスク遂行に必要となる時間を, 与えられた目標時間に対して最適化する機構である. 評価制御機構では実行時にシステムが自身の挙動を知るために認識, 動作計

画, 動作実行に関する品質と時間の関係性を表現する品質-時間テーブルが重要な構成要素となる. スケジューリングでは品質-時間テーブルを利用した品質の最適化を行う手法を提案する.

第 3 章「**遠隔操縦ロボットシステムのためのソフトウェア環境**」では評価制御機構を備えた統合ロボットシステムのベースとなる遠隔操縦型のロボットシステムにおいてユーザインタフェイスを中心としてロボットシステムをどのように設計するべきであるかについて災害対応ロボット競技会を題材として論じる. 人がフィードバックループに常に介在するような Human-in-the-loop 型の遠隔操縦ではなく, 人が時々にロボットにヒントをあたえ, 問題が発生しない限りにおいてはロボットが認識・計画・実行のフィードバックを自律的に行うことが可能な Supervised Autonomy 型のシステムを構成する. 自律的に動くシステムを基本とし, 行動の抽象度にしたがって, それぞれの抽象度にオペレータが介入可能なユーザインタフェイスを導入することで遠隔操縦において予期しない状況でも復帰可能なシステムを構成する.

第 4 章「**認識器と動作計画器の協調的サブスケジューリングによる足配置計画法**」では足配置計画法を題材にし, 動作計画と認識器を協調的に動作させることで高速に動作計画を行う手法について提案する. 動作計画器と認識器を協調的に動作させることで, 認識器は動作計画が必要としている領域を知ることができ, 必要となる認識処理を削減することが可能となる. 認識器と動作計画器の協調的スケジューリングによって足配置計画法の高速化手法が有効であることを実験を通じて確認する.

第 5 章「**継続的認識機能のための 3 次元点群追跡による物体認識器**」では 3 次元点群を利用したパーティクルフィルタによる実時間物体追跡手法について提案する. 継続的に対象物を認識する機能は, ロボットシステムのスケジューリングモデルの拡張および実行時の環境監視を実現するために重要となる機能である.

精度と計算時間および高速化のためのパラメータの関係について静的な環境での実験を行う.

第 6 章「**環境およびロボットの状態監視のための常時稼働型ソフトウェア**」では自律的な統合ロボットシステムで高次の意思決定機能となるシンボリックな記述に従ったタスク計画器を利用した行動実行時の監視機能について提案する. ロボットはタスク計画器によって計画された行動手順にしたがってタスクを遂行していくが, 実行時に発生するエラーに対して素早く対応することはプランベースなシステム構成に措いては難しい. 本研究ではこの環境への即時性を解決するため, 大域的・局所的なフィードバックという 2 つのフィードバックを導入する. 局所的なフィードバックは第 5 章で述べた実時間 3 次元点群追跡器を利用したビジュアルサー

ボである．大域的なフィードバックはシンボリックな記述によって得られる行動の前後および実行中に成立すべきシンボリックな条件を適時監視することで実現する．

第 7 章「評価制御機構に基づくプロジェクトスケジューリングの実装と評価」では評価制御機構における品質-時間テーブルの構築法について述べ, 品質-時間テーブルを利用したスケジューリング実験を行う．認識では 3 次元点群の低解像度化, 動作計画では衝突計算およびマニピュレーション軌道の詳細度, 動作実行部においては動作速度が与える安定性への影響を考慮して品質-時間テーブルを構築する．提案手法により構築した品質-時間テーブルを利用することで, 必要時間に応じたパラメータ選択が可能であることを実験を通じて示し, その有効性を確認する．

第 8 章「結論」において各章の内容から本研究を総括し，本研究の成果と貢献をまとめる．

第 2 章

認識計画実行機能の評価制御機構に基づく プロジェクト スケジュ ーリング

2.1　はじ めに

本章では, タスク遂行に必要とされる時間や品質にしたがってシステムが自動的にパラメータやアルゴリズムを 制御可能な評価制御機構について述べる.

ロボット システムは認識・ 計画・ 動作実行を 繰り 返して 行う 実世界システムである. 認識や計画のレ ベルが高く なってくるにつれて, 状況に 応じてプランナが必要になり , 認識・ 動作計画・ 動作実行のスケジューリングが重要となってくる. 認識・ 動作計画・ 動作実行のスケジューリングとはロボット が要求された目標を 達成するためにどのような 手順で行動を 実行していくかという 行動計画から , それぞれのタスク 要素である 認識・ 動作計画・ 動作実行にどの程度の時間を 費やすかに 至るまでを 決定する 問題である. これを 本論文ではプロジェクト スケジューリングと 呼ぶ.

評価制御機構とは, タスク 中の認識・ 動作計画・ 動作実行にどの程度の時間を 費やすかを 実行時に 制御するための枠組みである. 処理に 費やす時間は, その結果得られる 品質と 強い関係を 持つ. 必要時間をシステムが制御するためには, その結果変化する 品質の挙動をシステムが知識として 保持する 必要がある. このようなシステムがタスクの構成要素の時間と 品質に 関する 挙動を 知るための機構が評価制御機構の評価部分である. 一方, 実行時にタスク 構成要素の時間と 品質を 可変させる 機構が評価制御機構の制御部分である. 評価制御機構を 備えたロボット システムに 必要とされる 機能は以下の 3 点である.

1. 必要時間と品質を制御可能なパラメータを持った認識機能・動作計画機能・動作実行機能
2. 認識機能・動作計画機能・動作実行機能に関して, 必要時間と品質の関係を保持する品質-時間テーブル
3. タスク遂行のための時間的制約を満たすように必要となる時間と品質を実行時に制御するスケジューリング機能

本章ではこれらの機能を備えたロボットシステムの構成について述べる.

2.2 階層的なロボットシステムの構成

本節では多数のレイヤによって階層的に構成されるロボットシステムについて述べる. ロボットはハードウェアとして計算機に環境を認識するためのセンサ, 環境と相互作用するためのアクチュエータを付与した実世界システムである. このようなハードウェア構成によって様々なタスクを遂行するためには, 種々の機能が必要となる. その代表的なものとしては

1. 目標の位置まで移動するための移動計画機能
2. 移動の際に障害物となる環境中の障害物を認識する環境認識機能
3. マニピュレーション対象物の位置を認識するための画像処理機能
4. 目標対象物を把持するためのマニピュレーション動作計画機能

などが上げられる. これらの機能を組み合わせることで実世界システムとして実装するためには, 複数の記述階層を用意することが大きな助けとなる.

ロボットシステムの階層構造を明らかにするため, 井上の先駆的な議論を参考にすることで議論を進めていく [44]. 井上はロボットアームを対象にロボット言語の階層を次の 5 つに整理した.

1. レベル 0: コマンドレベル
   ロボットの関節角度の記述によってロボットの関節角度を制御するレベルである. その代表例として ML[45] が挙げられる.
2. レベル 1: 原始的動作レベル
   逆運動学等により, 手先や足先の座標記述によって手先・足先を動かすレベルである. VAL がその代表例である.

3. レベル 2: 構造的動作レベル

   制御構造を持ち, それによって出力を変化させるレベル. AL がその代表例である.

4. レベル 3: 対象物状態レベル

   ロボット アームが扱う 対象物の状態記述によるレベル. エジンバラ大学で開発された RAPT[46] がその代表例である.

5. レベル 4: 作業目標レベル

   シンボリックな目標状態の記述レベル.

各階層において, 目標は上位の階層から決定され, その目標を達成するための認識機能, さらに下位の階層への目標決定が順次なされていくようなシステムがロボット システムである. 本

Table2.1: ロボット 言語階層モデルに従った入出力関係

コマンドレベル	目標	各アクチュエータの目標関節角度
	センサ入力	各アクチュエータの現在関節角度を取得するためのエンコーダ出力
原始的動作レベル	目標	ロボット 座標系で表現されるエンド エフェクタの目標手先位置姿勢
	センサ入力	各関節のエンコーダ出力およびロボット のリンク構造と順運動学から計算される現在手先位置姿勢
構造的動作レベル	目標	ロボット 座標系で表現されるエンド エフェクタの目標手先位置姿勢列
	センサ入力	各関節のエンコーダ出力およびロボット のリンク構造と順運動学から計算される現在手先位置姿勢
対象物状態レベル	目標	対象物の目標位置姿勢
	センサ入力	カメラ画像等から推定される現在の対象物の位置姿勢
作業目標レベル	目標	シンボリックな記述による環境の目標状態. STRIPS[2] による記述が用いられる.
	センサ入力	カメラ画像等から推定される現在の対象物のシンボリックな記述

論文では原始的動作レベルおよび対象物状態レベルに焦点を当てて議論を進めていく. **Fig.** 2.1, **Table** 2.1 に各階層の記述とセンサからの入力関係を図示したものを示す. ここで, $\boldsymbol{\theta}$ はロボットの関節角度, $\boldsymbol{P}$ はエンド エフェクタの位置姿勢, I はカメラ画像, $\boldsymbol{x}$ は対象物体の位置姿勢である.

このような階層構造を対象物状態レベルを中心にして, システム構成図を再構成する. 対象物状態レベルよりも下位の階層は *Execution(Control)* としてブラックボックス化した (**Fig.** 2.2). **Fig.** 2.2 のように, 対象物状態レベルを中心としてロボット システムを捉えると, ロボッ

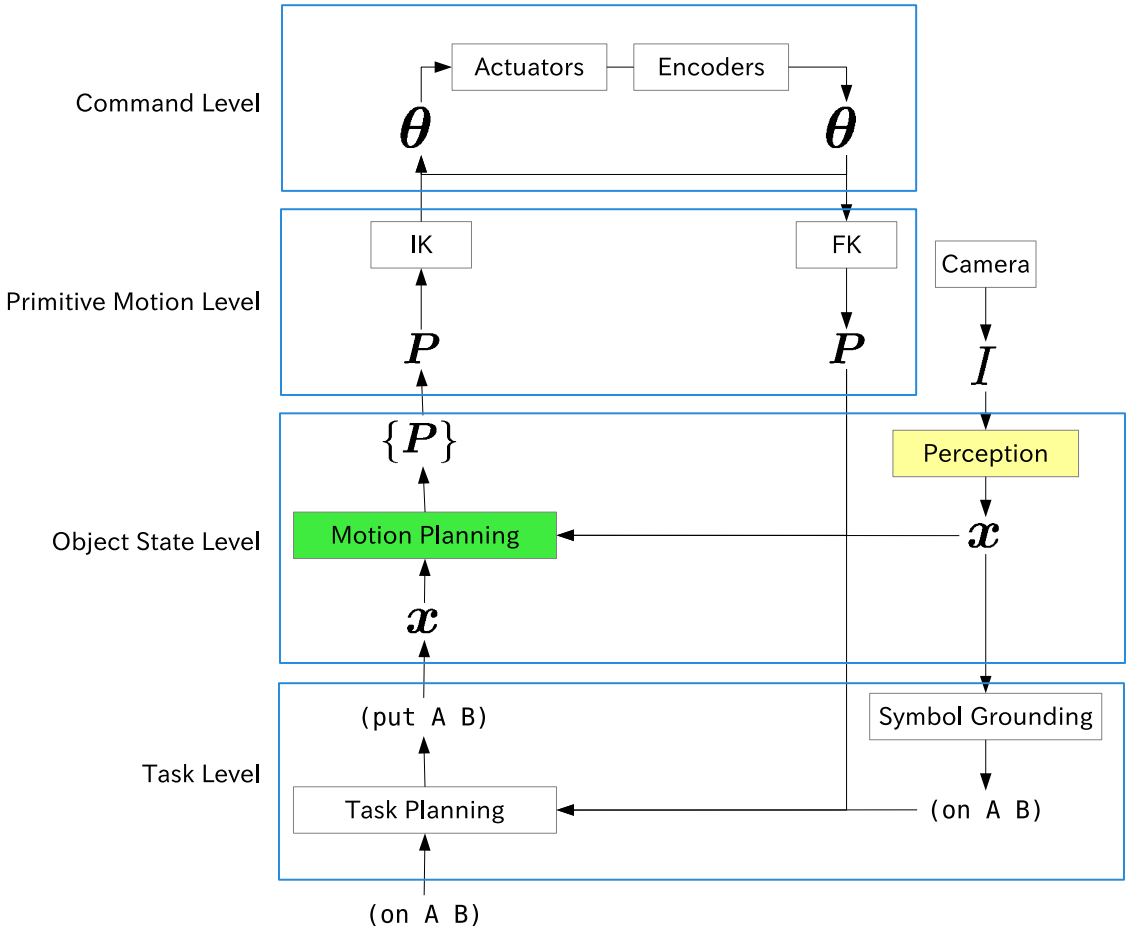


Fig2.1: Layered Robot System based on Robot Language Hierarchy

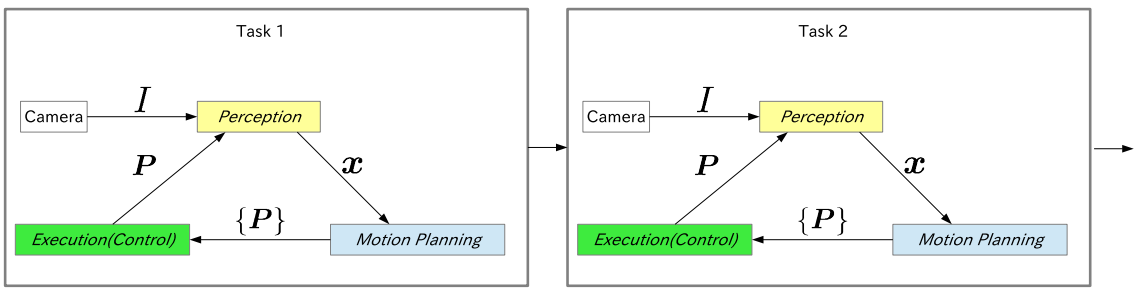


Fig2.2: Object State Level Centric Description of Robot System

トシステムは認識· 計画· 動作実行を順次実行していく実世界システムと考えることができる.

2.3　ロボットシステムにおけるプロジェクトスケジューリング機能

ロボットシステムは第 2.2 節で述べたように, 認識· 動作計画· 動作実行を順次実行していく実世界システムである. 本論文では, ロボットシステムの実現目標であるタスクの集合をプロジェクトと呼ぶ. 本論文の目標はプロジェクトにおいて各タスクの構成要素となる認識· 動作計画· 動作実行にどれだけの時間を費やすかを制御可能なシステムを構築することである. このような機能はロボットの制御に関するハードリアルタイムな実時間制御層の上位に位置し, ソフトリアルタイムなタスクスケジューリングを行う. 本節ではこのプロジェクトスケジューリングにおいてシステムが考慮すべき点について述べる.

2.3.1　タスクレベルのスケジューリング

タスク間の依存関係が存在しない簡潔な問題設定を考える. このような場合, 複数のタスクは同時に実行可能である (**Fig.** 2.3). 特に認識処理や計画処理は計算機における計算タスクのため, CPU コア数等の計算機資源に余裕のある限り 複数の処理を同時実行可能である.

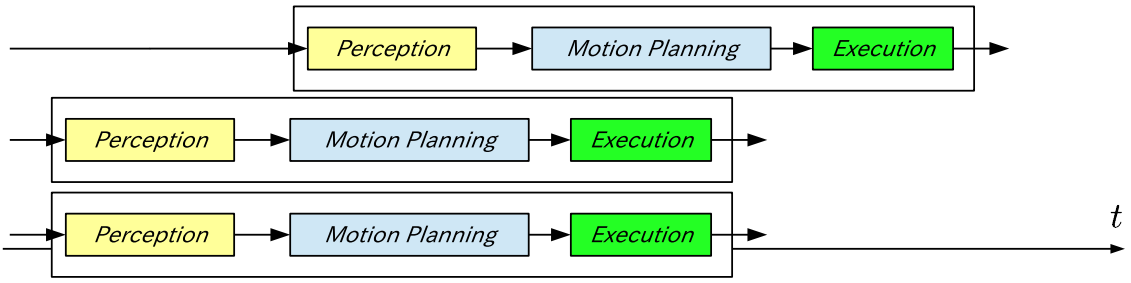


Fig2.3: Parallelized Task Timeline

ロボットシステムの実行順序をタスク T_i の連続として表現する. . プロジェクト $\{T_i\}$ を構成する認識部, 動作計画部, 動作実行部をそれぞれ P_i, M_i, E_i とする.

$$T_i = P_i + M_i + E_i \tag{2.1}$$

タスク T_i の時間に関するパラメータを **Table** 2.2, **Fig.** 2.4 に示す.

ロボットシステムにおけるスケジューリング機能として求められる機能は, 以下の二点を満足することである.

Table2.2: Time Attributes of Task T_i

Property Name	Description
t_i	Duration to execute T_i
r_i	Release time of T_i.
d_i	Deadline to execute T_i.
s_i	Datetime to start task T_i
e_i	Datetime to start task T_i. It is equal to $s_i + t_i$.
t_{P_i}	Duration to run P_i
s_{P_i}	Datetime to start P_i
e_{P_i}	Datetime to finish P_i. It is equal to $s_{P_i} + t_{P_i}$
t_{M_i}	Duration to run M_i
s_{M_i}	Datetime to start M_i
e_{M_i}	Datetime to finish M_i. It is equal to $s_{M_i} + t_{M_i}$
t_{E_i}	Duration to run E_i
s_{E_i}	Datetime to start E_i
e_{E_i}	Datetime to finish E_i. It is equal to $s_{E_i} + t_{E_i}$

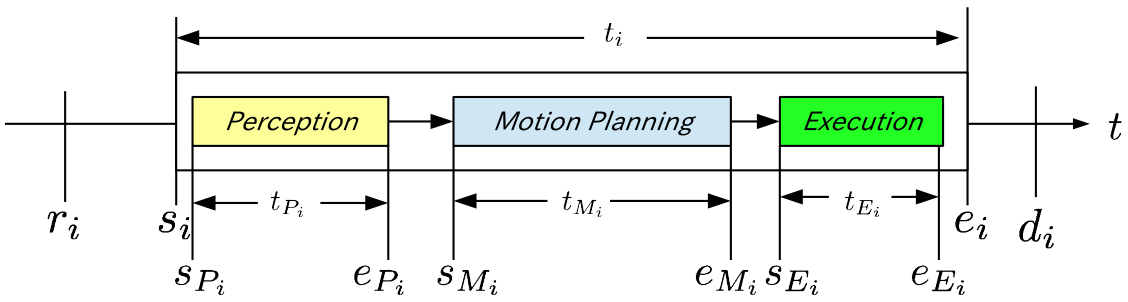


Fig2.4: Illustration of Time Attributes of Task T_i (Parameters from **Table** 2.2)

1. プロジェクト $\{T_i\}$ に関して, それらの相互的な依存関係を満足するようにタスク順序を決定する.
2. 全てのタスク T_i に関して, それらがデッドラインとして与えられた時刻 d_i よりも前に終了する.

$$e_i = s_i + t_i < d_i, \forall i \tag{2.2}$$

前者の依存関係において考慮すべき点は以下の 3 点である.

I ロボットのアクチュエータ資源の競合

ロボット のアクチュエータ資源は有限であり , 特にタスク遂行の際はロボット のルートリンクからエンド エフェクタに至るリンクのツリーはエンド エフェクタ数と 同数しか存在しない. 例えば双腕ロボット の右腕で 2 つの物体を 把持するというタスクが与えられた場合, 右腕のアクチュエータ資源が競合するため同時には実行できない. このような場合に 2 つの物体を 同時に把持するためには双腕を 利用する必要がある .

II ロボット のセンサ資源の競合

ロボット のセンサはそれぞれ視野内部でしかセンサデータを 取得できないため, 対象物を認識するためには視野内に対象物が存在しなくてはいけない. 例えば, 左右の離れた場所に存在する 2 つの物体を 一つのカメラで認識する場合, 同一の視点から 2 つの物体を 視野内に収めることができない. そのため物体を 認識するための視点計画およびロボット をその姿勢に動かすことが必要とされる .

III 環境の出力依存性

ロボット の動作手順や認識結果は環境の状態に影響を 受ける . 例えば, 戸で遮られた棚の中の物体を 把持するためには, 棚の中の物体認識およびその把持の前に戸をロボット で開ける必要がある .

これらの依存関係は, 各種タスクに関してシンボリックな記述を 用いることで解決することが可能である .

2.3.2　認識, 動作計画, 動作実行における依存関係

次に, 1 つのタスク内での依存関係について述べる . タスクは認識処理· 動作計画処理· 動作実行の 3 ステップから 構成される . **Fig.** 2.2 に示したシステム構成を 擬似コード で表現すると , **Alg.** 1 のように記述できる .

Algorithm 1 Perception, Motion Planning and Execution Loop of Robot System

```
1: while isFinished(Task) do
2:     object_pose = perception(robot_pose)
3:     robot_motion = motion_planning(object_pose)
4:     robot_pose = execute(robot_motion)
5: end while
```

擬似コード で表現することで明らかなように, 各ステップは前後に強い依存関係を 持ち, **Fig.** 2.8 のようなロボット システムのパイプライン化を 妨げるハザードと 捉えることができ

る (**Fig.** 2.5).

1. 認識-計画ハザード (Perception-Planning Hazard)

   認識によって対象物の位置・姿勢や環境モデルを推定しなくては動作計画をはじめることができない.

$$e_{P_i} = s_{P_i} + t_{P_i} \leq s_{M_i}, \forall i \tag{2.3}$$

2. 計画-実行ハザード (Planning-Execution Hazard)

   動作計画がアクチュエータ目標値を決定しなくてはアクチュエータへ指令値を送ることができない.

$$e_{M_i} = s_{M_i} + t_{M_i} \leq s_{E_i}, \forall i \tag{2.4}$$

3. 実行-認識ハザード (Execution-Perception Hazard)

   アクチュエータが駆動することでセンサ位置が変化する. そのため, 動作実行の前後でセンサ位置が変化する. 特にロボット自身の移動を考えると, 認識器が目標達成のために十分な精度の結果を返すためには対象物の近くに移動することが事前に必要となる.

$$e_{E_i} = s_{E_i} + t_{E_i} \leq s_{P_j}, i < j \tag{2.5}$$

これらのハザードが直接的な理由となり, ロボットの認識・動作計画・動作実行といった基本的な行動の並列化は難しい.

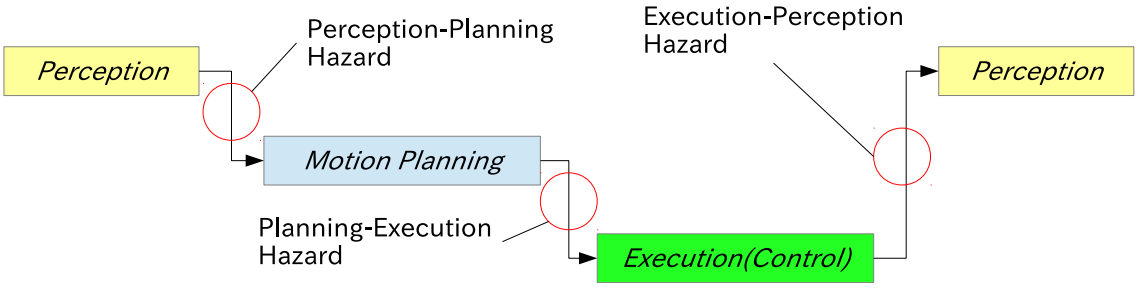


Fig2.5: Hazard of Task Execution in Pipeline Manner

動作計画における依存関係と並列実行

動作計画に関しては動作計画が対象とする問題に応じて並列化が可能な場合がある. 動作計画は解の種類によって Goal Planning と Trajectory Planning の 2 つに分類できる. Goal Planning は目標をどこに設定すればよいかという問題を解き, Trajectory Planning は目標が

明らかなことが前提で, 現在の状態から 目標状態までの遷移を 計画する. このような分類をした場合, Trajectory Planning を 行うためには事前に Goal Planning を 行う 必要がある.

移動計画を 例にとると, Goal Planning はロボット の移動位置をどこにするかという 計画, Trajectory Planning は現在位置から 目標位置までどのような 軌道を 描くことで目標位置まで到達可能かという 計画になる. マニピュレーション計画の場合, Goal Planning は把持位置を決定する Grasp Planning, Trajectory Planning では現在の腕の姿勢からエンド エフェクタを目標位置までを 動かす腕の関節角度列を 計画する.

その一方, 動作計画が対象とする 問題の相対的な 大きさに従って, 大域計画と 局所計画と 分類することも 可能である. 動作計画が対象とする 探索空間が大きく, ロボット を 動かすために必要な 粒度で探索するのが困難な 場合, はじめに 粗い粒度で大域計画として 計画し, それを 規範として 詳細な 粒度で局所計画を 行うことで高速に探索することが可能である. この手法は特に移動ロボット の経路計画においてよく 用いられる. **Fig.** 2.6 に移動ロボット の経路計画における 大域計画・ 局所計画の例を 示す. 経路計画においては, 大域計画では探索次元を x, y の 2 次元で行い, 局所計画は大域計画で生成されたプランを 参考に x, y, θ の 3 次元でさらに詳細な 障害物地図を 用いてロボット が動くべき 軌道を 計算する. このような 手法は Dynamic Window Approach[47] や Elastic Bands[48] として 知られている.

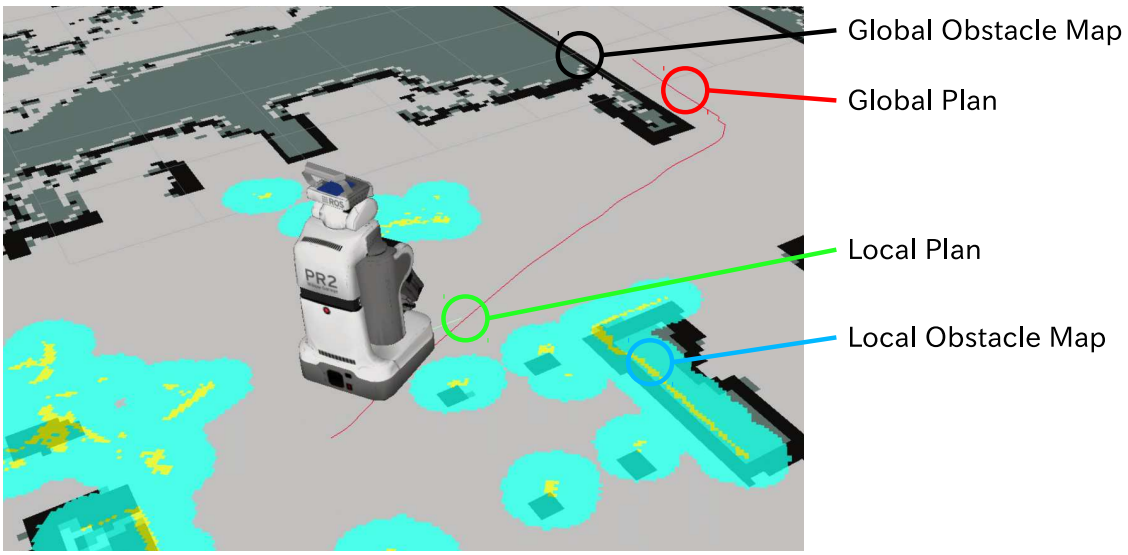


Fig2.6: Local and Global Planning in Navigation Planning

移動とマニピュレーションにおける 具体的な 動作計画は以下のような 個別の計画問題から 構成されている.

1. 立ち位置計画 (Stand Location Planning)
   対象物を把持するためにはロボットはどの位置でマニピュレーションをすべきかを計画する.
2. 移動計画 (Path Planning, Footstep Planning)
   現在の位置から目標となる立ち位置まで移動するためにどのようにロボットを動かすかという軌道を計画する.
3. 把持計画 (Grasp Planning)
   対象物の位置や形状を入力として, エンドエフェクタをどのように配置することで対象物の把持が可能であるかを計画する.
4. マニピュレーション計画 (Motion Planning, Manipulation Planning)
   現在のロボットの関節角から, 把持計画によって得られたエンドエフェクタ位置までロボットの腕を動かす軌道を計画する.

このような依存関係として整理した場合, 立ち位置計画がおわったあとの移動計画とマニピュレーション計画は互いに依存関係がない. 従ってこれらを同時に並列で計算することが可能である. **Fig.** 2.7 に移動とマニピュレーションに関して, 動作計画器および実行部を含めた依存関係を示す. この図から, 動作計画の依存関係が, アクチュエータ資源の競合を考慮した場合に実行部まで含めて独立であることがわかる. このように動作計画・動作実行部まで含めて独立なモジュール群がある場合, システムがこれらを並列に実行することでパフォーマンスを向上させることが期待できる. しかしその一方で, アクチュエータ資源の競合が存在しなくても実行のタイミングの同期が重要である場合は, 一つの計画問題, 動作実行器としてシステムは取り扱う必要がある.

2.3.3 認識, 動作計画, 動作実行の必要時間と品質の関係

認識処理, 動作計画処理, 動作実行に必要とされる時間 t を決定するには品質と計算時間, 動作実行時間の関係を考慮する必要がある. 本論文ではこれらの品質をパラメータ q で表現する. 一般に精度の良い計算結果を求めるには高い計算コスト, 計算時間が必要となる. 逆に精度の低い計算結果であれば必要となる計算コストは低くなり, 必要となる計算時間は短くなる. したがって, 認識処理, 動作計画処理は計算タスクであるため, 計算処理時間 t_{P_i}, t_{M_i} は精度や信頼性と言った計算や実行の結果に関する品質 q に依存する量である.

その一方でロボットの動作速度という観点から動作実行について考えると, 動作速度が速く

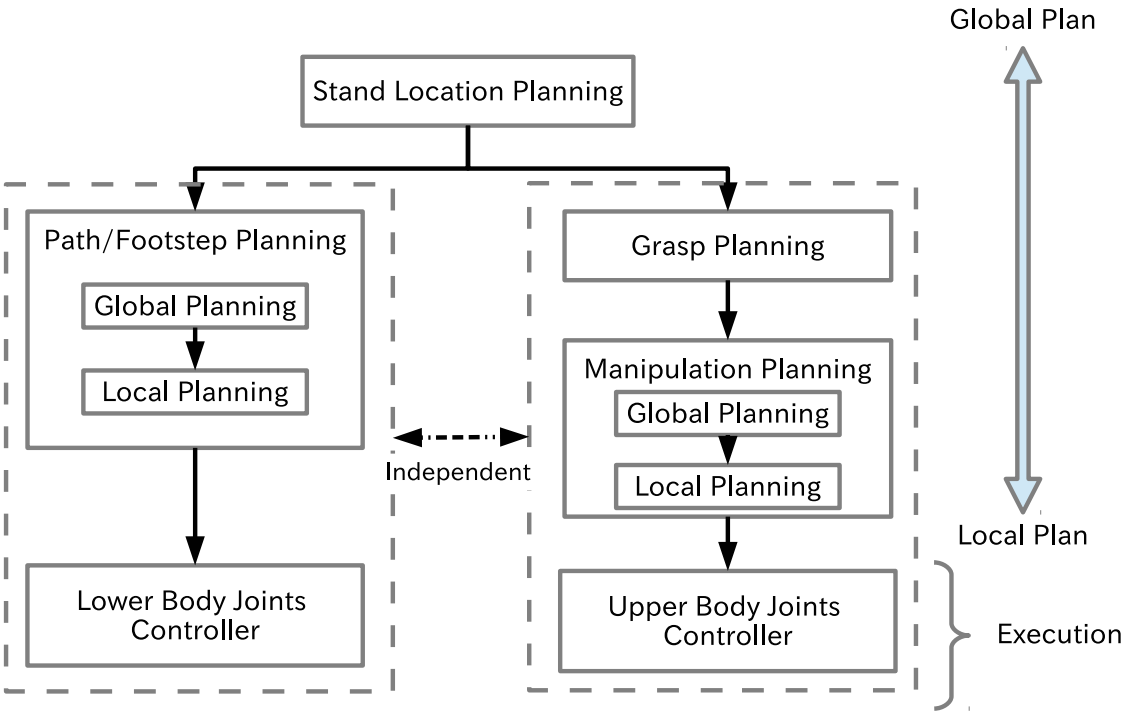


Fig2.7: Dependency between Planning Modules and Execution

なるにつれて以下の 2 点において品質は下がる.

- 環境との予期しない衝突による外乱

  環境が動的に変化したり, 認識処理や動作計画処理の精度の低さに起因し, 予期せずロボットと環境が衝突する可能性がある. この時, ロボットの動作速度が速いと環境との衝突からロボットに作用する運動量および角運動量が大きくなるため, 転倒につながったり環境やロボットを破壊してしまう可能性が高くなる.

- 運動量・角運動量による脚型ロボットのバランス制御の不安定化

  動作速度が速くなるほど運動量および角運動量によって脚型ロボットのバランス制御は困難になる. 動作速度が速くとも転倒を防ぐためには, 分解運動量制御 [49] 等によって関節角度を転倒を防ぐように制御によって修正する必要があるが, これは動作計画結果と異なるアクチュエータ目標を実行することになり, タスク遂行に致命的な影響を与える可能性がある.

これらの動作速度に依存した信頼性は環境に固定されているようなロボットでは問題になりにくく, 脚型ロボットでは転倒が発生するため特に顕著な問題となる.

2.3.4　実時間制御および環境, ロボット監視のための常時稼働型ソフトウェア

タスク遂行のために順に実行される認識, 動作計画, 動作実行といった処理と同時に, プロジェクト実行中に常に実行されるべき機能としては, 以下の 2 つが上げられる.

- サーボ制御およびバランス制御のための実時間制御プロセス
  ロボットの制御はハードリアルタイムなシステムとして知られている [50]. ロボットのアクチュエータ目標は周期的なデッドラインを満たすように制御される必要がある. また, 脚型ロボットのようにセンサフィードバックによる制御をしなくては転倒してしまうようなロボットの場合, 実時間制御プロセスの中でセンサフィードバック処理を実現する必要がある. このような実時間制御プロセスは本章で今まで述べたようなタスクよりも優先度が高い計算処理であり, 常時最優先度で計算されるべき処理である. 実践的には, このようなハードリアルタイムな処理が必要となる計算はロボット体内に特別に実時間 OS を搭載した制御用計算機を用意してその上で優先度の高い実時間プロセスとして実行するような構成を用いる.
- 環境およびロボット自身の監視機能
  外界センサによる環境の監視, 内界センサによるロボットの状態監視はタスクの実行順序を決定するのと同様に重要な構成要素である. 外乱や認識誤差によって予期しない環境の変化がタスク遂行中発生した場合, ロボットは動作列を修正したり, より上位の行動手順を再度計画することが必要となる. このような環境の変化を速やかにロボットの行動に反映させるためには, 環境を常時監視することが重要となる.

2.4　評価制御機構によるプロジェクトスケジューリング

本節では, プロジェクトスケジューリングのためにロボットシステムに評価制御機構とよぶ機能を導入する.

2.4.1　インプリサイス計算の制御によるプロジェクトスケジューリング

インプリサイス計算とは計算品質と計算時間の間にトレードオフが存在するような計算タスクである. インプリサイス計算の制御は計算品質と計算時間の間のトレードオフな関係を利用することである. インプリサイス計算の制御を利用することでリアルタイムシステム構築の際

に, デッドラインに応じた計算品質の制御が可能である [10, 11]. 本小節ではインプリサイス計算によるスケジューリングのアプローチをロボットシステムにおけるプロジェクトスケジューリング問題に適用する. これは本論文が提案する評価制御機構の制御に該当する機能である.

一般的な計算機の計算タスクとロボットシステムにおけるタスク, およびそれらのスケジューリングに関する差異について以下にまとめる.

- 実行タスク数

  計算機における計算タスクは OS 上で走る様々な計算タスクであり, 多種多様なものに及ぶ. その一方でロボットにおけるタスクは数が少ない. これに依存して, コンテクストスイッチの回数はロボットにおけるタスクの方が小さいという特徴を持つ.

- タスク同士の依存関係

  計算機における計算タスクはロボットのタスクに比べて依存関係が弱い. そのため, 各タスクをタイムスライスすることでタスクの進捗を制御するスケジューリングが行われる. その一方で先に述べたようにロボットのタスクは依存関係が強いため複数のタスクを並列に行なっていくことは難しい.

インプリサイス計算のスケジューリングではまずリアルタイムタスクに関しては最低限の精度が実現できる最小時間の計算を計算タスクの優先度に従って CPU に割り当てる. その後に精度を向上させるための計算に時間を割り当てるように計算タスクをタイムスライスする. しかしながらこのような違いにより, リアルタイムシステムのスケジューリングに用いられているインプリサイス計算のスケジューリング手法そのものをロボットシステムに適用することは難しい.

インプリサイス計算のスケジューリングでは, タスク T_i の構成要素である認識処理・動作計画処理および実行に必要となる時間 $t_{P_i}, t_{M_i}, t_{E_i}$ は定数ではなく, 制御可能な関数である. それぞれの品質を $q_{P_i}, q_{M_i}, q_{E_i}$ とすると, それらは時間パラメータ $t_{P_i}, t_{M_i}, t_{E_i}$ の関数となる.

$$q_{P_i} = q_{P_i}(t_{P_i}) \tag{2.6}$$
$$q_{M_i} = q_{P_i}(t_{M_i}) \tag{2.7}$$
$$q_{E_i} = q_{P_i}(t_{E_i}) \tag{2.8}$$

また, 一般に計算時間および実行時間を長くするほど得られる品質は高くなるため, 以下のような特徴を持つ.

$$\frac{dq_{P_i}}{dt_{P_i}} \geq 0 \tag{2.9}$$

$$\frac{dq_{M_i}}{dt_{M_i}} \geq 0 \tag{2.10}$$

$$\frac{dq_{E_i}}{dt_{E_i}} \geq 0 \tag{2.11}$$

プロジェクト 全体での必要時間 t は **eq.** 2.12 のように表現される.

$$t = \sum_i (t_{P_i} + t_{M_i} + t_{E_i}) \tag{2.12}$$

一方でプロジェクト 全体での品質 q は, 重み係数 w を用いて線形結合によって表現する.

$$q = \sum_i (w_{P_i} q_{P_i} + w_{M_i} q_{M_i} + w_{E_i} q_{E_i}) \tag{2.13}$$

重み係数 w は計算機上の計算タスクでは各プロセスの優先度に該当するものである.

プロジェクト スケジューリングの目的は, 全体の時間 t_{d} を人から与え, それにしたがって各処理に必要な時間を決定し, かつ品質 q を最大化することである.

$$t \leq t_{\mathrm{d}} \tag{2.14}$$

$$\max q = \max \left(\sum_i (w_{P_i} q_{P_i} + w_{M_i} q_{M_i} + w_{E_i} q_{E_i}) \right) \tag{2.15}$$

以降本論文ではデッドライン時刻 d_i ではなく, デッドライン時間 t_{d} を用いる.

品質 q を最大化する問題は, 一般に q と t の関係は非線形であることから, 非線形最適化の問題となる. 本論文ではシステム実装者がある程度初期値を決定しておくことで, その初期値に従って各パラメータを変化させていくことで q を与えられた制限内で最大化する. そのためのアルゴリズムを繰り返し計算回数を k として以下に示す.

$$t_k = \sum_i (t_{P_i,k} + t_{M_i,k} + t_{E_i,k}) \tag{2.16}$$

$$q_k = \sum_i (w_{P_i} q_{P_i,k} + w_{M_i} q_{M_i,k} + w_{E_i} q_{E_i,k}) \tag{2.17}$$

1. $t_k > t_{\mathrm{d}}$ の場合, 各 $q_{P_i,k}, q_{M_i,k}, q_{E_i,k}$ に関して, 微小量 Δt 動かした時に q の変化量が最も小さいものを選択する. このとき選ばれたものを S とする.

$$S \in \{P_i, M_i, E_i\} \tag{2.18}$$

$$w_S \left|\Delta q_S(t_{S,k})\right| \leq w_x \left|\Delta q_x(t_{x,k})\right| \tag{2.19}$$

$$\tag{2.20}$$

$t_k < t_{\mathrm{d}}$ の場合は q の変化量が最も大きいものを選択する.

$$w_S \left|\Delta q_S(t_{S,k})\right| \geq w_x \left|\Delta q_x(t_{x,k})\right| \tag{2.21}$$

$$\tag{2.22}$$

ただし , x は S 以外のタスク要素である .

$$x \in \{P_i, M_i, E_i\} \tag{2.23}$$
$$x \neq S \tag{2.24}$$

Δt は以下を 満たす微小量である .

$$\Delta t < 0 \ (\text{if } t < t_k) \tag{2.25}$$
$$\Delta t > 0 \ (\text{if } t > t_k) \tag{2.26}$$
$$\tag{2.27}$$

2. S に関して q_S および t_s を更新する .

$$q_{S,k+1} = q_{S,k}(t_{S,k} + \Delta t) \tag{2.28}$$
$$t_{S,k+1} = t_{S,k} + \Delta t \tag{2.29}$$
$$q_{x,k+1} = q_{x,k} \tag{2.30}$$
$$t_{x,k+1} = t_{x,k} \tag{2.31}$$
$$t_{k+1} = \sum_i (t_{P_i,k+1} + t_{M_i,k+1} + t_{E_i,k+1}) \tag{2.32}$$
$$q_{k+1} = \sum_i (w_{P_i} q_{P_i,k+1} + w_{M_i} q_{M_i,k+1} + w_{E_i} q_{E_i,k+1}) \tag{2.33}$$

3. $|t - t_{k+1}| < \Delta t$ の場合, 計算を 終了する .
4. ステップ (1) に戻る .

このアルゴリズムが簡易なタスク例においてどのように振る舞うかを **Fig.** 2.8 に示す. この簡易モデルでは, 認識処理, 動作計画処理および実行はそれぞれ 1 回のみであるとする .

$$T = P + M + E \tag{2.34}$$

(1) システム実装者によって初期値として $(t_{P,0}, q_{P,0}), (t_{M,0}, q_{M,0}), (t_{E,0}, q_{E,0})$ が与えられる . また, $q = w_P q_P(t_P), q = w_M q_M(t_M), q = w_E q_E(t_E)$ の関係はすでに既知であるとする . このとき, スケジューラは初期値から 時間を削減し なくてはいけないとする .

$$t_0 = t_{P,0} + t_{M,0} + t_{E,0} \tag{2.35}$$
$$t_0 > t_{\mathrm{d}} \tag{2.36}$$

(2) $w_P q_P, w_M q_M, w_E q_E$ に関して負の微小量 Δt だけ初期値から動かした値 $w_P q_P(t_{P,0} + \Delta t), w_M q_M(t_{M,0} + \Delta t), w_E q_E(t_{E,0} + \Delta t)$ を 計算する .

(3) ステップ (2) で計算した値を 比較し (**eq.** 2.38, **eq.** 2.39), 品質 q の変化量が最小である $w_M q_M$ を 操作対象とする .

$$w_M \left|\Delta q_M(t_{M,0})\right| \leq w_P \left|\Delta q_P(t_{P,0})\right| \tag{2.37}$$

$$w_M \left|\Delta q_M(t_{M,0})\right| \leq w_E \left|\Delta q_E(t_{E,0})\right| \tag{2.38}$$
$$t_{M,1} = t_{M,0} + \Delta t \tag{2.39}$$
$$q_{M,1} = w_M q_M(t_{M,1}) \tag{2.40}$$

(4) ステップ (2) と同様に再度負の微小量 Δt だけ各点を動かす.

(5) ステップ (4) の値を比較することで, 品質 q の変化量が最小である $w_E q_E$ を操作対象とする.

$$w_E \left|\Delta q_E(t_{E,1})\right| \leq w_P \left|\Delta q_P(t_{P,1})\right| \tag{2.41}$$
$$w_E \left|\Delta q_E(t_{E,1})\right| \leq w_M \left|\Delta q_M(t_{M,1})\right| \tag{2.42}$$
$$t_{E,2} = t_{E,1} + \Delta t \tag{2.43}$$
$$q_{E,2} = w_E q_E(t_{E,2}) \tag{2.44}$$

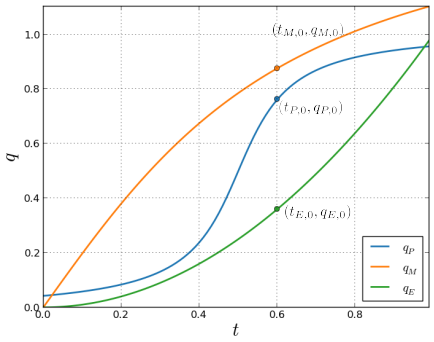
(6) 同様の操作を $t_k < t_\mathrm{d}$ となるまで繰り返す.

ここで示したアルゴリズムは強い初期値への依存性を持ち, 目的関数 q が最大であるとは限らない (**eq.** 2.15). しかしながら, システム実装者が与えた初期値から順に動かしていくような挙動によって, パラメータの変化は予期しやすいものとなる.

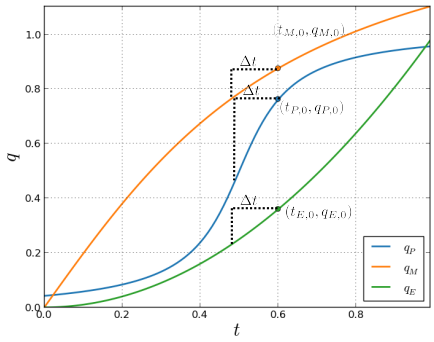
2.4.2　プロジェクトスケジューリングのための品質-時間テーブル

第 2.4.1 節で述べたように, 品質 q と時間 t の関係は一般に非線形であり, その関係は個別の認識器・動作計画器・制御器の実装およびロボットのアクチュエータやセンサ構成, 環境に依存する. 評価制御機構を備えたロボットシステムは, 時間と品質に影響を与えるパラメータを変化させることで, システム実装者が設定したパラメータによる実行結果を参照として計算時間・動作実行時間・精度の関係を予め試行し, その関係性をテーブルとして保持する. これは評価制御機構の評価に当たる機能である. このような品質に関わる指標は認識精度やロボットの転倒といった致命的な失敗まで様々なものが考えられる. **Table** 2.3 にシステムの抽象度に従って品質に関わる指標について整理する. 品質 q はこれらを統合的に判断し, 設計されるべき指標である. 本論文では特に認識処理, 動作計画処理, 制御レベルに着目し, タスク成功率に依存する量として定義する.

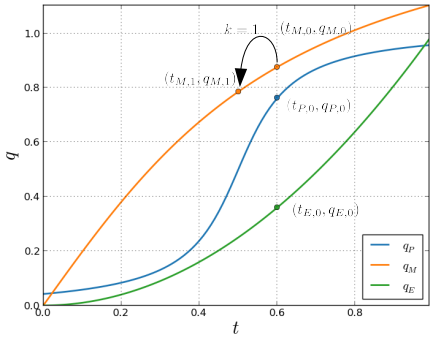
評価制御機構を実現するために重要となるのは, 品質と時間を制御可能なパラメータを持った認識機能・動作計画機能・動作実行機能である. これらのパラメータはタスクへの依存性に従って 2 種類に分けることができる. このような分類は, 選択したアルゴリズムに依存すると言い換えることも可能である. **Table** 2.4 に代表的なパラメータを示す. **Table** 2.4 に代表さ



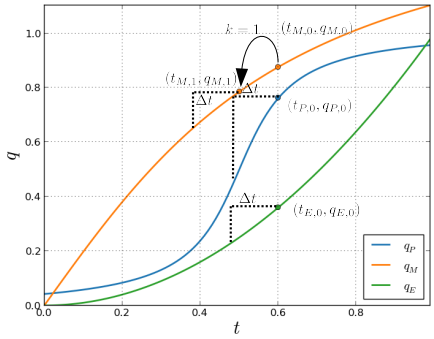
(1)



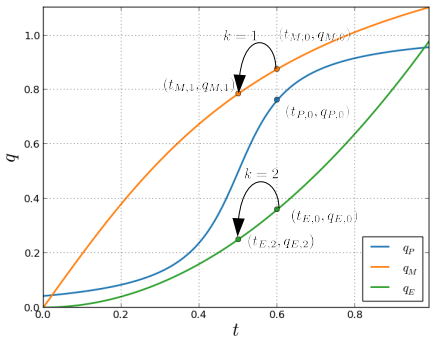
(2)



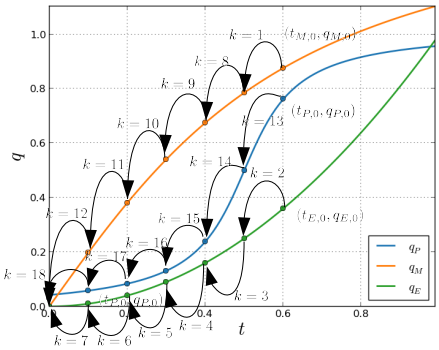
(3)



(4)



(5)



(6)

Fig2.8: Behavior of Project Scheduling Example of Simple Model

Table2.3: 品質-時間関係性テーブルにおいて考慮する指標例

抽象度レベル	品質に関連する指標
タスクレベル	タスクの成否
認識処理	計算時間・精度
動作計画処理	計算時間・精度・信頼度
制御レベル	致命的な失敗につながる制御の安定性 (転倒など)
ロボットセンサ	サンプリング精度・誤差
計算機資源	占有 CPU 数
ロボットアクチュエータ	アクチュエータ負荷

れるようなパラメータを変化させることによって, 品質 q と必要時間 t を可変に制御する. パラメータの種類によっては整数値をとるような離散的なパラメータが存在しうる. そのため, 第 2.4.1 節でのべた微小量 Δt は離散的な値を取る可能性がある. また, 一般に一つの認識機能および動作計画機能であっても品質 q と必要時間 t は複数のパラメータに依存する. そのような場合, 必要時間 t が一意に決定されるようにパラメータ間に制約を導入する, もしくは離散的なパラメータ集合の中から選択するといった拘束条件を導入する必要がある.

2.5 移動を伴うマニピュレーションプロジェクトスケジューリングにおける実行モデル

タスク内での認識処理, 動作計画処理, 動作実行のタイミングはプロジェクトスケジューリングにおける最適化対象である. すでに述べたように, Belief Space と呼ばれる確率的な記述を幾何的な記述に導入することで, 動作計画処理と実行に加え認識処理のタイミングを計画する手法が提案されている [6]. 本章ではこれまでさらに一つ下のレイヤである認識処理, 動作計画処理, 動作実行の時間を制御するという問題について取り扱っている. これらすべてを最適化問題として解くと計算量が肥大してしまうため, 本論文では強いヒューリスティックをもつ階層的なアプローチを用いる. タスクレベルの依存関係の解決 (第 2.3.1 節) にはシンボリックなタスクプランニング, もしくはあらかじめ定義してある手順に従って実行するとする. さらに認識処理, 動作計画処理, 動作実行の順序に関しては固定的なモデルを用いる. 本小節では移動を伴うマニピュレーションタスクに関して 3 種類の実行モデルを定義し, それぞれについて第 2.4.1 節で述べた品質と時間の関係について述べる. これらの実行モデルは連続する 2 つの

Table2.4: Parameter Examples for Perception and Planning to Control Time and Accuracy.

Module	Parameter Group	Parameter Name
Perception	Common Parameter	Resolution of Image
		Resolution of PointCloud
		Region-of-Interest of Sensor Field of View
	Task Specific Parameter	Number of Particles
		Number of Keypoints
		Offset from Supporting Plane
Motion Planning	Common Parameter	Sampling Resolution (Step Length) of Searching
		Tolerance of Goal Testing
		Number of Links to Move
		Number of Links to Take into Account Collision Checking
		Duration to Execute Planned Motion
	Task Specific Parameter	Trajectory for Manipulation
		Resolution of Trajectory

タスクである移動タスクとマニピュレーションタスクを複合させたプロジェクト に対して定義されるものである. これらの実行モデルからどの実行モデルを 利用するかを 選択するには, それぞれの実行モデルを 利用して必要時間 t_{d} を 与えたプロジェクト スケジューリングの結果得られる品質 q が最も 大きい物を 選択する.

2.5.1　移動前認識実行モデル

移動前認識実行モデルは, 移動の前に 認識を 行い, それを 元に移動計画及びマニピュレーション計画さらにはそれらの実行を 行うような実行モデルである (**Fig.** 2.9). このモデルの利点は, 認識処理および動作計画処理が一度しか必要ないため, 計算に 必要とされる時間が短縮できることである. その一方で移動タスクによってロボット の自己位置に誤差が発生するため, 移動距離が長くなるようなプロジェクト では誤差が大きくなり , タスク実現が難しい実行モデルである. ロボット の移動距離 l によって生じる認識品質 q_{P_0} および計画品質 q_{M_0} の低下を減衰項 $L(l)$ としてモデル化する. このとき, プロジェクト 全体でかかる時間は t は以下のように

なる.

$$t = t_{P_0} + t_{M_0} + t_{E_0} + t_{E_1} \tag{2.45}$$

一方品質 q は以下のようになる.

$$q = L(l)\left(w_{P_0} q_{P_0} + w_{M_0} q_{M_0}\right) + w_{E_0} q_{E_0} + w_{E_1} q_{E_1} \tag{2.46}$$

ただし, 正確な移動距離 l は認識 P_0 の結果得られるものであるため, プロジェクトスケジューリング時には粗い精度の事前知識に従って決定される.

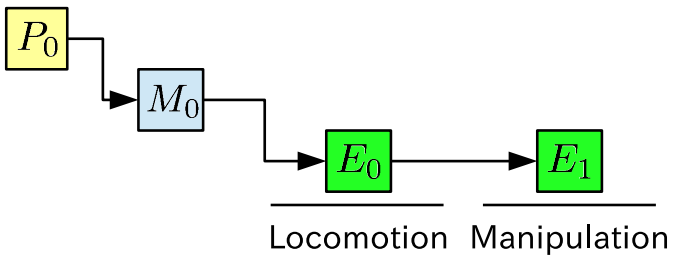


Fig2.9: Perception-before-Traversing Model

2.5.2 移動前後認識実行モデル

移動前後認識実行モデルは移動の前に認識を行い, それを元に移動計画を行うが, 移動後にもう一度認識処理を行ってマニピュレーション計画と実行を行うような実行モデルである (**Fig.** 2.10). これは各タスクに対して認識処理, 動作計画処理を行うものであり, 最も慎重な実行モデルであると言える. 移動前後認識実行モデルの利点は, 移動前認識実行モデルと比べて, ロボットの移動による誤差がマニピュレーションに影響を与えない点である. このとき, プロジェクト全体でかかる時間は t は以下のようになる.

$$t = t_{P_0} + t_{M_0} + t_{E_0} + t_{P_1} + t_{M_1} + t_{E_1} \tag{2.47}$$

一方品質 q は以下のようになる.

$$q = L(l)\left(w_{P_0} q_{P_0} + w_{M_0} q_{M_0}\right) + w_{E_0} q_{E_0} \tag{2.48}$$

$$+ w_{P_1} q_{P_1} + w_{M_1} q_{M_1} + w_{E_1} q_{E_1} \tag{2.49}$$

2.5.3 継続的認識機能による移動中認識実行モデル

移動中認識実行モデルは特殊な実行モデルである. 移動中認識実行モデルは移動中に認識器を継続して実行することで, 認識処理に必要となる計算時間を短縮する実行モデルである

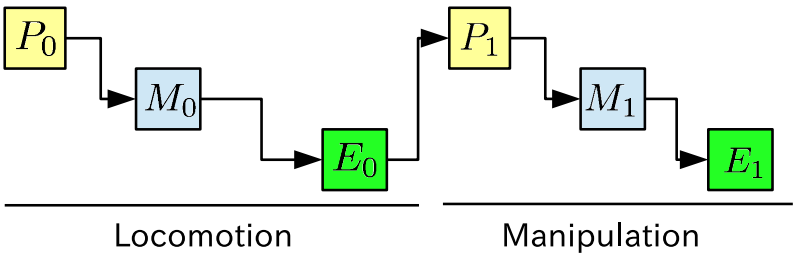


Fig2.10: Perception-after-Traversiing Model

(**Fig.** 2.11). また, 動作計画に関しても継続的な認識結果を 考慮できるため, 移動終了後に改めて認識した結果を 用いる 必要がなくなる. 移動中認識実行モデルのためには, 認識機能に関して特別に継続的な認識が可能な認識機能が必要となる. このとき, プロジェクト 全体でかかる時間 t および品質 q は以下のようになる.

$$t = t_P + t_{M_0} + t_{E_0} + t_{E_1} \tag{2.50}$$

$$q = w_P q_P + w_{M_1} q_{M_1} + w_{E_0} q_{E_0} + + w_{E_1} q_{E_1} \tag{2.51}$$

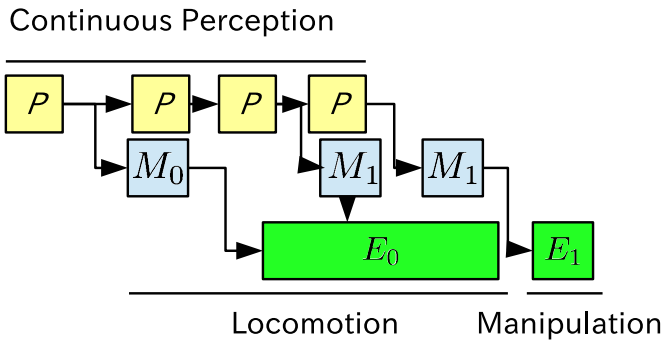


Fig2.11: Perception-during-Traversiing Model

2.5.4　特殊な実行モデルとしての認識器と動作計画器の協調的サブスケジューリング

本小節では先に述べた 3 つの実行モデルに加えて認識器と 動作計画器を 協調的に利用していく特殊なスケジューリングモデルについて述べる. **Fig.** 2.12 にその概要図を 示す. 認識器と動作計画器の処理タイミングを 協調的に制御することで, 動作計画器の進捗を 手がかりとして必要な領域のみに注視領域を 絞り 認識処理を 行うことができる. このような最適化されたスケジューリングは, 注視領域を 制御するという 特性から, 動作計画器が目標となるような物体位置というよりは環境構造を 利用するような動作計画において有効である.

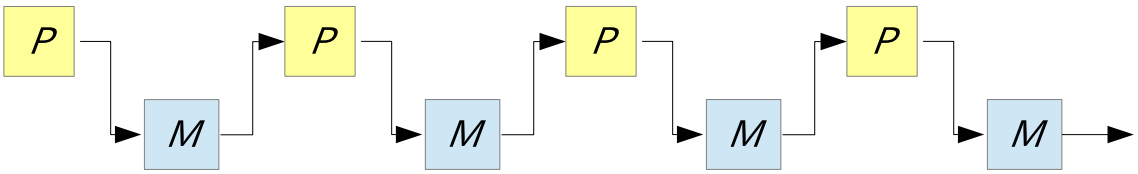


Fig2.12: Collaborative Sub-Scheduling of Perception and Motion Planning

2.6　評価制御機構を備えたロボットシステム構成

本章におけるこれまでの議論を踏まえ, 本論文で提案する全体システム構成を **Fig.** 2.13 に示す.

- 品質-時間テーブル (Quality-Time Table)

  第 2.4.2 節で述べた品質と計算時間および実行時間を調節可能なパラメータの関係性を保持するテーブルである. このテーブルはシステム設計者が設定したデフォルト値を基準として, パラメータを変化させた時の挙動について実測データを下に作成される. 認識器および動作計画器はこのテーブルを参照することで, 人から支持された要求度に従ったパラメータを利用する. タスクに依存するパラメータ, 依存しないパラメータによらず, パラメータの値はタスクに依存するためテーブルはタスクごとに保持される.

- プロジェクトスケジューラ (Project Scheduler)

  プロジェクトスケジューラはタスクプランナや遠隔操縦インタフェイスによって指示されたタスク列に対して, 第 2.4.1 節において述べたインプリサイス計算の制御によるプロジェクトスケジューリングを行う. プロジェクトスケジューラは認識器, 動作計画器および実行器に対してスケジューリングの結果得られた品質 q および時間 t, さらにそれらを実現する具体的なパラメータ集合を実行時に設定する.

- 監視機構 (Execution Monitor)

  監視機構はタスク実行中に第 2.3.4 節で議論したようにタスク状態を監視するための機構である. この監視機構は 2 種類に分類される.

  – シンボル監視機能

    シンボル監視機能はタスクプランニング部と連携する監視機構である. タスクプランニング部によって生成される行動列が期待するシンボリックな環境記述の変化をセンサ入力を下に推定し, 監視する. この監視機能が異常を検知した場合, ロボット

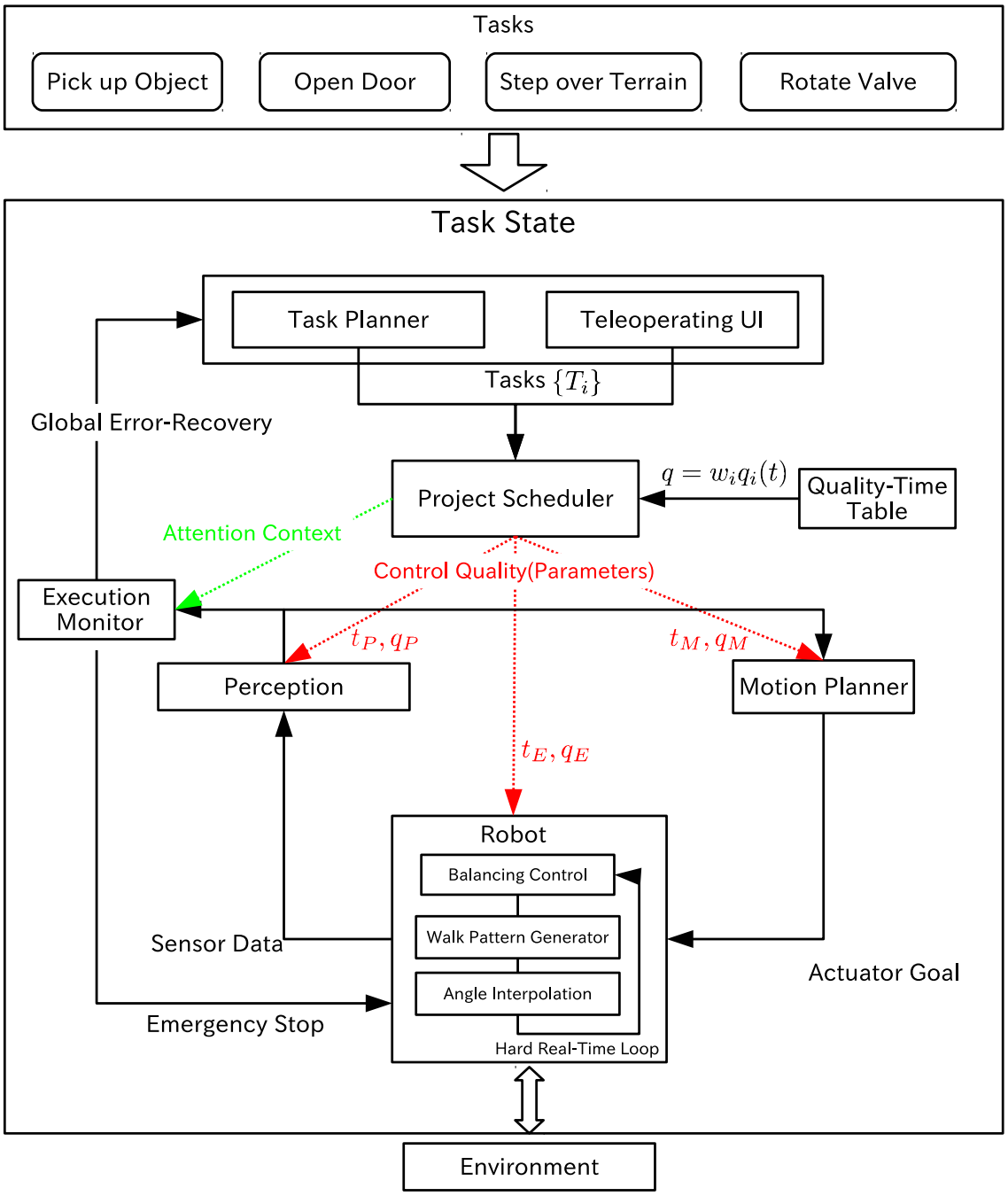


Fig2.13: Overview of Evaluation-Controlled Robot System

の動作を停止させ再度タスクプランニングを行うことでグローバルな行動フィードバックを構成する.

– 動作監視機能

動作監視機能はシンボル監視機構に含まれない監視機構を指す. この監視機構は主に予期しない環境との衝突や視覚によって認識困難な小さな障害物による床面状態の悪化などを監視する. 動作監視機能が異常を検知した場合, シンボル監視機構と同様にロボットの動作を停止させる. 停止した後には再計画をしても異常を解消できない可能性が高い. というのもこれは事前の動作計画では予期出来なかった異常のためである. 本システムではこのような場合, 遠隔操縦インタフェイスを通じてオペレータに指示を仰ぐことでタスク遂行を目指す.

- 低レイヤのロボット制御システム (Control System)

低レイヤのロボット制御システムは常にセンサフィードバックループを高速な実時間処理によって実行する. **Fig.** 2.13 においては, Robot と書かれた四角の中に相当する部分である. 低レイヤの制御システムに要求される機能は多岐に渡るが, 目標関節角度までアクチュエータを動かすための補間機能, 転倒を防ぐためのバランス制御, 歩行のための脚軌道生成といったものが代表的なものとしてあげられる. 特に転倒を防ぐためのバランス制御は重要な機能である.

2.7 おわりに

本章では, タスク遂行に必要とされる要求時間にしたがってシステムが自動的にパラメータを制御可能な認識計画実行機能の評価制御機構を備えたシステムを構成するための要素について述べた.

評価制御機構を備えたロボットシステムでは以下のような点が重要であった.

1. 認識, 動作計画, 動作実行のそれぞれおいて, 品質と必要時間を制御可能なパラメータを持つものを利用する. そのようなパラメータは特定の認識器, 動作計画器によらない共通のパラメータとタスクに強く依存するパラメータの 2 種類に分類することが可能であった.
2. 精度や信頼性を反映した品質と必要となる時間の関係性を品質-時間テーブルとしてシステムが保持しておく. 品質-時間テーブルは実行時のプロジェクトスケジューリング

に利用された.

3. タスク遂行時には品質-時間テーブルを 参照することで, 実行時に 必要となる 品質と 時間を 制御する . 品質を 制御する 際には, システム実装者が与えた 初期値に 基づき, 与えられた 必要時間を 満たしながら 品質を 最大化するようなパラメータを 選択するアルゴリズムについて 述べた.

このような 構成によって, システム実装者によって 注意深く 調整されてきたパラメータをシステムが与えられた 必要時間にしたがって 自動的に 決定可能であることについて 述べた.

第 3 章

遠隔操縦ロボットシステムのためのソフトウェア環境

3.1 はじめに

本章では, 第 2 章で述べた認識処理, 動作計画処理および実行の繰り返しであるロボットシステムに対し, オペレータによる遠隔操縦の統合法について述べる (**Fig.** 3.1). また, 本章で示す遠隔操縦システムは災害対応競技会で実際に運用し, その結果及び考察についてまとめる. 最後に競技会での結果の総括として実際に競技中にどのような行動に対して時間を費やしていたかの解析を行う. 遠隔操縦インタフェイスは提案するシステム構成のなかでタスク計画機能

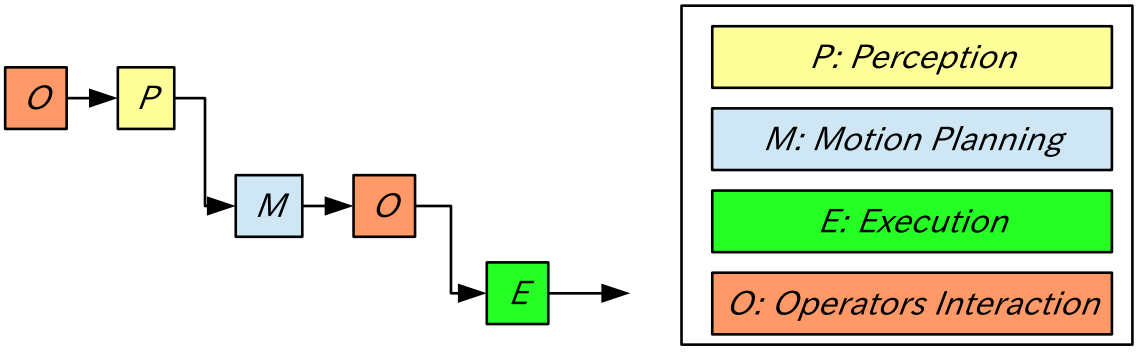


Fig3.1: Overview of Teleoperating User-Interface and Sytem Integration

と共に上位に位置し, タスク遂行のための意思決定を行う. タスク計画機能は既知の環境に関してはよく機能するが, 遠隔操縦システムに期待されることは環境に関する事前知識が不確かな状況でロボットでのタスク遂行を実現することである. 本章で示すシステムを災害対応ロボット競技会において利用し, その競技会での発生した予期しない問題に対してどのように遠隔操縦システムが機能したかについてまとめる.

DARPA Robotics Challenge(DRC)[1] は 2015 年 6 月 5 日から 6 日にかけて米国, カリフォルニア州で行われた災害対応ロボット競技会である. DRC の目的は災害対応のためのロボットを開発し, 災害対応シナリオの中でロボットがどの程度活躍できるかを明らかにすることである. DRC に参加したチームは以下の 8 個のタスクを 60 分以内にロボットの遠隔操縦によって達成することを求められる.

- *Vehicle*: Polaris Ranger 車をロボットによってスタート地点から目的地まで運転する. 経路には障害物や壁が設置されているため, ロボットはこれらを回避しながら目的地に到着する必要がある.
- *Egress*: Polaris Ranger 車からロボットを降車させ, 屋内環境へと続くドアの前までロボットを移動させる.
- *Door*: ドアノブを回転し開くことでドアを通過する.
- *Valve*: 屋内環境に設置してあるバルブを 360 度回転させる.
- *Drill*: 屋内環境に設置してあるドリルを把持し, 周辺に設置されている壁を切断する. 壁には目標となる領域が図示されており, その領域を残すことなく壁から切断することが求められる.
- *Rubble*: コンクリートブロックによって作られた不整地, もしくは瓦礫の設置してある領域のどちらかを踏破する.
- *Special Task*: 競技会直前までは内容が伏せられているマニピュレーションタスク. 内容はボタンを押す, ホースを接続する, レバーを倒す等である.
- *Stair*: 屋外環境に設置された階段を登る. 階段の一方には手すりが設置されており, この手すりを利用しても良い.

参加チームはこれらのタスクを 1 つ完遂するごとに 1 ポイントを獲得することができ, すべて完遂できた場合は 8 ポイントを獲得することとなる. 競技中はロボットの転倒を防ぐためのクレーンやテザーを設置することは許されない. ロボットが競技中に転倒した場合は, ロボットを屋内環境入り口のドアの手前まで移動させることができるが, 最低 10 分間のインターバルが設けられる.

ロボット実機とオペレータの間の通信は制限されている. *Valve*, *Drill*, *Rubble* および *Special Task* は屋内タスクとして定義されており, 屋内タスクにおける通信は屋外タスクに比べて厳しく制限される. DRC 競技会では, ロボット実機と Field Computer (ロボットの近く

に設置された計算機), Field Computer と OSU(Operator Station Unit, 遠隔操縦サイト) の 2 つの境界に関して通信量の制限が設けられる.

- wifi ネットワークによるロボット実機と Field Computer の通信:
  ロボット実機と Field Computer の間の通信は wifi によって行われ, その帯域は 300 Mbps に制限される. wifi によってロボットを離れた場所にある計算機とつなぐシステム構成は, ロボットを環境に有線で繋がずに動かすためには必須となる.
- Field Computer と OSU 間の通信制限:
  Field Computer と OSU 間では 2 つの通信経路が定義される. 2 つの通信経路に関してそれぞれ異なった通信制限が課せられる.
  - Narrow Path:
    Narrow Path と呼ばれる通信経路では, 通信帯域は 9600 bps に制限されるが, Field Computer から OSU, OSU から Field Computer という双方向の通信が可能である. また Narrow Path を経由する通信は常に利用することができる. 帯域制限を超えたパケットはネットワークスイッチのキューに追加される. これは帯域制限を超えた通信は Bufferbloat[51] を引き起こす事を意味する. Bufferbloat はネットワークが許容できる帯域を超える通信が行われると大きな遅延が発生する現象である. 通信遅延が発生するとオペレータは誤った操作をロボットに指示してしまったり, ロボットへの指示を慎重になりすぎてしまう傾向がある. そのため, 本遠隔操縦システムでは Bufferbloat を避けるため, Narrow Path を経由した通信は厳しく制御する.
  - Fast Path:
    Fast Path と呼ばれる通信経路では, 通信帯域は 300 Mbps に制限され, Narrow Path と比較して多くのデータを送信することができる. その一方で通信の方向は Field Computer から OSU の片方向のみが許可される. また, Fast Path は屋外タスクの間は常に利用可能であるが, 屋内タスクでは利用可能なタイミングが限られる. 通信可能なタイミングは変化するが, 平均して 30 秒間に 1 秒程度利用可能である. この場合 29 秒間は通信が断絶していることになる. Narrow Path における通信とは異なり , 帯域を超えたパケットや, 通信不可能なときに送信されたパケットはキューに追加されることなく棄却される.

これらの通信制限は擬似遅延発生装置によって生成される．この通信制限は競技会のためにルール化されたものであるが, 実環境のネットワークで発生する問題に影響を受けている．Narrow Path の帯域制限は狭帯域インターネット回線, Fast Path における帯域制限は wifi や携帯電話のような無線ネットワークを想定していると考えられる．

遠隔操縦システムにおいて, ユーザインタフェイスは重要なソフトウェア機能であり , その設計はオペレータおよびロボットのパフォーマンスに大きな影響を及ぼす．T.Koolen らはヒューマノイドのための遠隔操作ユーザインタフェイスを”coactive design”[52] と呼ばれるアプローチに従って提案した [53]．このユーザインタフェイスは Virtual Robotics Challenge[54, 55] と呼ばれる , DRC 以前に行われた DRC 出場チームを決定するためのシミュレーションによる予備競技会のために作られたものである．このユーザインタフェイスは 3 次元可視化機能によって 3 人称視点によってロボットの状態, 3 次元点群, 動作計画結果を同時に可視化する．一方 2 次元可視化機能も備えており , ロボットの状態と動作計画結果がロボットに搭載されたステレオカメラ画像に重畳して可視化される．K.Stefan らは DRC Trials と呼ばれる DRC 出場チームを決定するためのロボット実機による予備競技会のための遠隔操作ユーザインタフェイスを開発している [56]．これも同様に 3 次元可視化機能, 2 次元可視化機能を備えており , 半透明のロボットモデルとして動作計画結果がセンサデータに重畳される．本章で提案するユーザインタフェイスもこのような 3 次元可視化機能, 2 次元可視化機能を中心に設計する．本章で提案する遠隔操縦システムでは, ロボットの現在状態を可視化した特別なセンサビューワとその可視化にのみ責任を持つオペレータを特別に採用する．このセンサビューワは予期しない状況に対応できるように, アクチュエータの温度のような低レイヤのセンサ情報を , ステレオ画像や 3 次元点群と同様に可視化する．

遠隔操縦のパフォーマンスと , オペレータが利用できるセンサデータの劣化は強い相関を持つ．Darken らは偵察タスクにおいて視野画像の更新周期がオペレータの操縦パフォーマンスに影響を与えることを示した [57]．実験では更新周期を 1.4 fps から 21.37 fps まで変化させ, 遅い更新周期では致命的なパフォーマンス低下が見られると示している．Van Erp と Padmos らは車の運転タスクにおいて同様に視野画像の更新周期がオペレータの操縦パフォーマンスに影響を与えることを示している [58]．Van Erp らの操縦実験では, 10fps を下回る更新周期では致命的なパフォーマンスの低下が見られたとしている．しかしながら , これらの実験が前提としているような更新周期は DRC が設定している通信制限のもとでは実現が難しい．したがって本論文では, Human-in-the-loop と呼ばれるオペレータがロボットのフィードバック

ループに常に介在して操作するようなモデルではなく, Supervised Autonomy[27] に従って制限された通信を介したシステム構築をする．Supervised Autonomy はオペレータが時折ロボットの動作目標や認識のための補助を行い, その後はロボット側のフィードバックループに従って動くようなシステム構成である．

本章は以降以下のような構成になっている．第 3.2 節においてハードウェア, 第 3.3 節においてソフトウェア構成の概要を示す．第 3.4 節では, 信頼性の低い帯域制限された通信環境での遠隔操縦システムの構成について述べる．遠隔操縦のためのユーザインタフェイスは第 3.5 節において議論する．第 3.6 節では DRC 競技会における結果と, その中で本論文で示したシステムがどのように機能したかを述べる．第 3.7 節において本章の結論と今後の展望について議論する．

3.2　ハードウェア構成の概要

本節では, 本論文で利用するロボット JAXON(**Fig.** 3.2) の概要を示す．JAXON は高速かつ高トルクな動作が必要とされるタスクのために開発された等身大ヒューマノイドロボットである．目標としている高速かつ高トルクな動作としては全身を動かす高速なダンス動作や, 災害環境下におけるタスク遂行を目標としている．JAXON は全身 33 自由度を持つ 2 脚 2 腕型

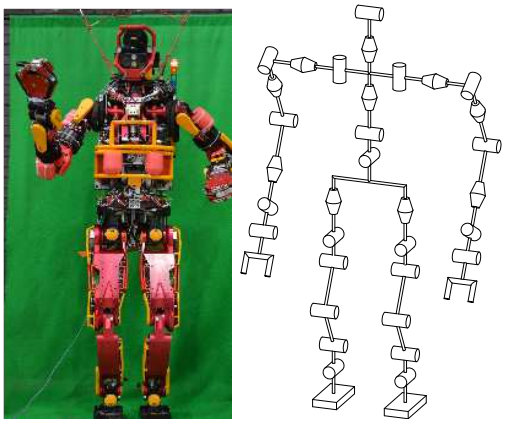


Fig3.2: JAXON: 188cm Height and 127kg Weights.

ヒューマノイドロボットである．8 自由度の双腕アーム, 6 自由度の脚, 3 自由度の腰部および 2 自由度の首部から構成される．JAXON の全長は 188cm であり, その重量は 127kg である．JAXON の特徴は, 水冷システムによるモータの高負荷運用である [59, 60]．モータに大きな電流を流すと発熱による焼損が予期されるが, 短い時間であれば外部から強制的に水冷するこ

とで定格スペック以上のパフォーマンスを出すことが可能である．これを利用して高速かつ高トルクな出力を実現している．各関節は相対的な位置を高精度で検出するロータリエンコーダとホール素子により絶対的な位置を計測するアブソリュートエンコーダの 2 種類をセンサとして備えている．外界センサとしては頭部に Carnegie Robotics 社の Multisense SL を頭部に備えている．Multisense SL はスリップリングによってロール軸方向に無限回転する hokuyo 社のレーザレンジファインダとステレオカメラから構成される．バッテリとしては約 1 時間超のスタンドアローンな活動ができるだけのリチウム-フェライトバッテリが搭載されている．計算機システムとしては, 内部に 2 つの計算機を備えている: 一つは制御用計算機であり , もうひとつは画像処理用計算機である．制御用計算機は 4 コアの Core i7-2600K 3.40 GHz で主記憶は 8GB である．画像処理用計算機は 4 コアの Core i7-4770R 3.90 GHz で主記憶は 16GB である．

3.3　遠隔操縦ソフトウェア構成の概要

本章で示すソフトウェアアーキテクチャは 3 つの階層から構成される (**Fig.** 3.3).

- 制御レイヤ:

  制御レイヤは実時間センサフィードバックを担当する階層である．実時間センサフィードバックは 500 Hz 周期で実行される．制御レイヤは歩行軌道生成, エンドエフェクタを環境に馴染ませるためのインピーダンス制御 [61],

  転倒防止のための全身のバランス制御 [62, 63] を含む．制御レイヤは制御用計算機に実装される．制御レイヤ内の各モジュールは OpenRTM[31] によって構成され, OpenRTM の実時間制御, 実時間通信機能を利用している．

- 認識計画レイヤ:

  認識計画レイヤは認識およびマニピュレーションのための動作計画のためのモジュール群から構成される．これらのモジュールはロボットに組み込まれた計算機と Field Computer にまたがって実装される．

  マニピュレーションのための動作計画器は *Valve* タスクにおいてバルブを回転するような際にヒューマノイド全身の関節角を利用した動作列を生成する．動作計画モジュールは逆運動学を用いて全身の関節角度列を生成するが, その中では関節角度限界回避 [64] や自己衝突回避 [65] を考慮している．一方認識モジュールは回転レーザとステレオカメ

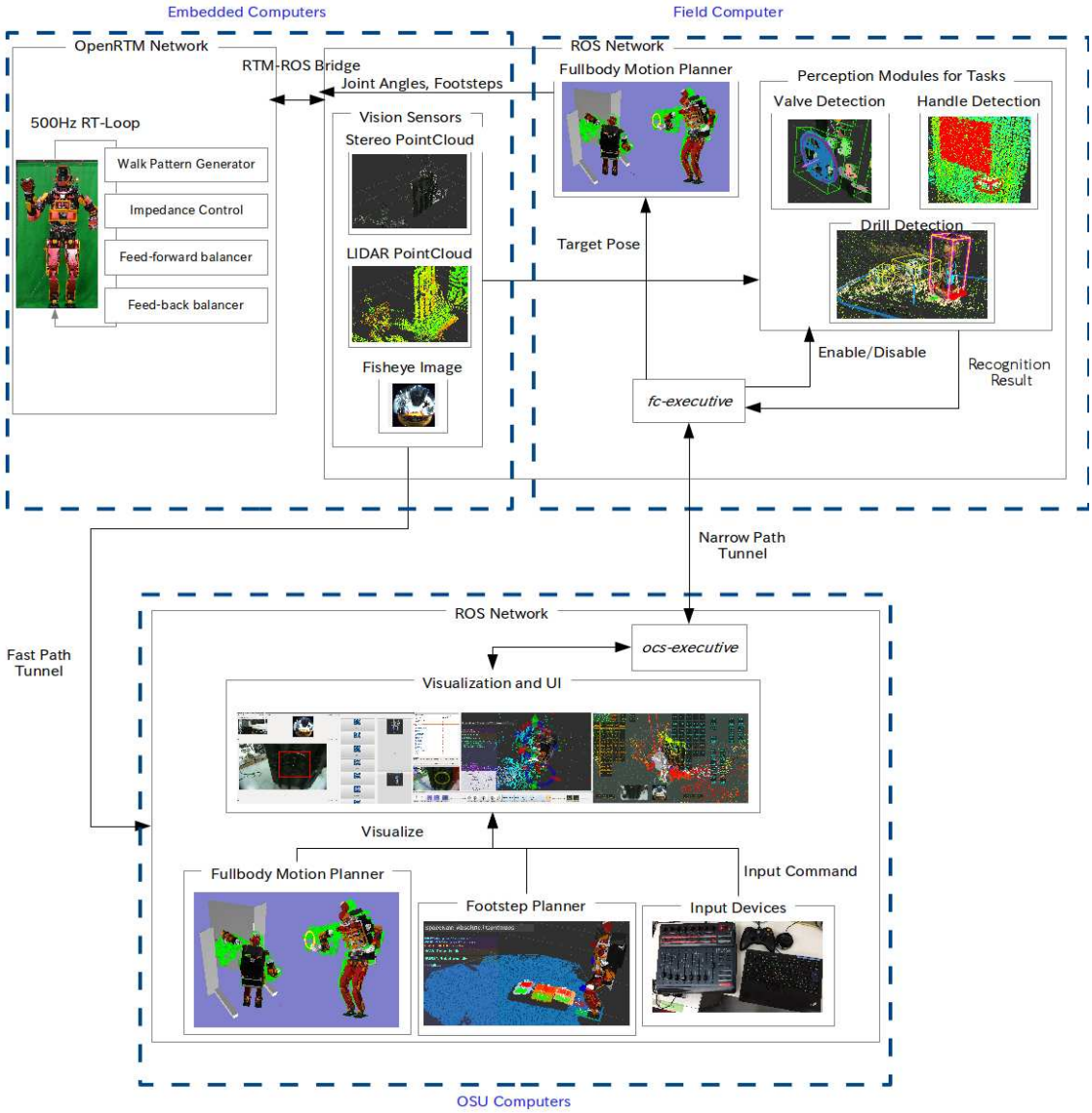


Fig3.3: Overview of Teleoperation Software Architecture

ラの 3 次元点群を利用して対象物や環境の認識を行う．認識計画レイヤの各モジュールは ROS[34] によって相互に通信し，認識計画レイヤはひとつの ROS ネットワークを構成する．

認識計画レイヤの中で特殊な位置づけにあるのが *fc-executive* と呼ばれるモジュールである．*fc-executive* はこのレイヤの管理・調停モジュールとして振る舞う．*fc-executive* は次に述べる可視化操縦レイヤからオペレータの指令値を受け取り，それを各モジュールへの指令値として解釈し指示する．認識モジュールに対しては，タスクに応じてどのモジュールを有効にし，どのモジュールを無効にするかを制御する．動作計画モジュー

ルに対しては, 認識モジュールの認識結果や遠隔操縦者の指令値を目標値として入力し, 動作列を生成させる. *fc-executive* は動作実行中に一定の箇所に Break Point を保持しており, オペレータは可視化操縦レイヤを通して動作実行中に動作を停止させることができる.

- 可視化操縦レイヤ:

  可視化操縦レイヤはセンサデータを可視化するための 3 次元可視化ツールキットとオペレータがロボットに指令するための操縦インタフェイスから主に構成される. 操縦インタフェイスは 2 次元および 3 次元インタフェイスの 2 つから成り, これらは OSU に設置されている入力デバイスによって操作される. これらの詳細に関しては第 3.5 節にて述べる. 可視化操縦レイヤを構成するソフトウェア群は認識計画レイヤと同様に ROS によって相互に通信し, 認識計画レイヤとは異なるもうひとつの ROS ネットワークを構成する.

  認識計画レイヤにおいて *fc-executive* が管理・調停モジュールとして機能したように, 可視化操縦レイヤでは *ocs-executive* が管理・調停モジュールとして機能する. *ocs-executive* はオペレータからの指示を操縦インタフェイスを通して解釈し, *fc-executive* へと送る. またその一方で, *ocs-executive* は *fc-executive* を経由して認識結果を受け取り, それを 3 次元可視化ツールキットへと出力することで, オペレータに認識結果を掲示する. *ocs-executive* はロボット・環境・オペレータをモデル化してシナリオとして内部に記述するため, 有限要素機械によるグラフ記述を採用している. これによってオペレータが意図しない操作を誤ってしてしまった場合に, *ocs-executive* はその操作を無視することで致命的な操作ミスを防ぐ.

本節で述べたように, このシステム構成は通信を ROS に大きく依存した構成であるが, 認識計画レイヤと可視化操縦レイヤでそれぞれ別の ROS ネットワークから構成されている. これは遠隔操縦対象のロボットとオペレータの間の通信制限に起因する. そのため, この間での通信を実現するために, 特別なストリーミングプロトコルを用いる.

3.4　低信頼性・低帯域対応のための通信システムの設計

3.4.1　通信システムの概要

本小節では, 通信システムの概要について述べる. そのための問題設定として, DRC 競技会の通信制限を採用する. 通信帯域制限を超えて通信するために, 本システムでは以下のようなアプローチを採る:

- ロボット – Field Computer 間の通信:
  - ステレオカメラの 3 次元点群に関しては x-y-z の浮動小数点数の配列を用いる代わりに, 深度画像を用いる. 深度画像を用いることで, 約 8 分の 1 のデータサイズにすることができる. 深度画像はカメラパラメータを利用することで x-y-z 表記の 3 次元点群データ構造に変換可能である.
  - ステレオカメラの色画像に関しては, JPEG 圧縮をする. このデータは 3 次元点群に色をつける, およびオペレータに現在のロボットの視野画像を提供するという用途で用いられるため, JPEG 圧縮によるデータ欠損は大きな影響を与えないと判断した.
  - 低レイテンシ・高いスループットを要求するプロセスはロボット体内の計算機で実行する. 特に IMU や力センサのような制御レイヤの出力を利用するプロセスは制御用計算機で実行する.
- Field Computer – OSU 間の通信:
  - 2 つの独立した ROS ネットワークの利用

    すでに述べたように, Field Computer と OSU でそれぞれ 2 つの独立した ROS ネットワークを利用する. これは, ROS は ROS ネットワークにおいて DNS サーバのように振る舞う中央プロセスに対して, 常に信頼性の高い通信が必要だからである. またノード間においても常に信頼性の高い通信が期待されている. 先に述べた帯域制限によって Field Computer と OSU 間の通信は制限されているため, このような条件を満たすことは難しい.
  - 点群および画像の通信のための Fast Path の利用

    点群や画像を通信するためには Narrow Path の 9600 bps という帯域制限は十分ではないため, Fast Path を利用してこれらのデータをロボットからオペレータへ

と送信する.

– ロボットの基本センサ情報のための Narrow Path の利用

関節角度や IMU によるロボットの傾きは Narrow Path を利用してロボットからオペレータへと送信する. これらのデータは画像や点群と比べると十分に小さく, またオペレータが常に最新の値を確認できることで, 遠隔操縦のパフォーマンスを落とさないことが期待できるためである.

– Narrow Path を利用した操縦コマンドの送受信

操縦コマンドは常時送るというよりは, 目標値の設定などタスクの始まりのみ通信するので, そのための通信経路としては Narrow Path を利用する. これはオペレータからロボットという方向の通信手段は Narrow Path しか利用できないということも理由となっている.

– OSU および Field Computer の両方への動作計画器の設置

ロボットの動作をオペレータがロボットへと指示するためにはロボットの全身の関節角度列を送る必要があるが, これは許された帯域を超えてしまう可能性がある. そのため, OSU および Field Computer の両方に動作計画器を設置することで, 動作計画の目標値のみを通信する. これによって, オペレータが実行前に動作計画結果をプレビューすることが可能となる.

3.4.2　Field Computer と OSU 間の通信のための通信プロトコル

本システムでは, Field Computer と OSU 間で設定されている帯域制限を満たしながらの通信を実現するため, 簡潔なネットワークプロトコルを開発し利用する.

ハンドシェイクを必要とするような通信プロトコルは Field Computer と OSU の間の通信には適さない. というのも, 通信を確立するためのハンドシェイクによって必要とされる通信量は, Narrow Path の帯域制限条件を超えてしまう可能性があり, 帯域制限条件を超えることによって生じてしまう Bufferbloat は遠隔操縦に致命的な影響を与えてしまうからである. TCP/IP による通信は three-way handshake と呼ばれるハンドシェイクを通信確立時に必要とする. より抽象化されたプロトコルである ROS や ZeroMQ[66] といったものもノード間の接続を確立するためにハンドシェイクを必要とする.

本システムではこのような問題を回避するため, UDP をベースとした簡潔なストリーミングによる通信プロトコルを採用する. ストリーミングベースなプロトコルを採用することで,

再接続が必要となった時でも速やかに通信を再開することができる．再接続が必要な状況としては Fast Path の通信制限に起因する通信断絶, および予期しないシステムの問題による通信プロセスの再起動を考慮している．本システムではこれらのストリーミング通信路を「トンネル」と呼ぶ. Fast Path と Narrow Path のためのトンネルは送信プロセスと受信プロセスの 2 つによって構成され, これらは通信開始時に特殊な初期化を必要としないように設計されている．そのため, 通信の断絶が起きた場合でも即座に通信を再開可能である．

Narrow Path トンネルのプロトコル設計

Narrow Path で許容されている通信帯域は 9600 bps であるため, 通信量は可能な限り小さくする必要がある．本システムはすでに述べているように, ROS による通信に大きく依存しているため, ROS の通信フォーマットをバイパスするトンネルとして設計されている．通信フォーマットとしては ROS のサブセットとして定義した．この ROS サブセットの主たる制限としては, ROS で可能なメッセージの入れ子構造, 及び可変長データの禁止である．**Table** 3.1 に本システムで採用した ROS サブセットと ROS の通信フォーマットの比較を示す．こ

Table3.1: Comparison between ROS Full Spec and ROS Subset Protocol

	ROS full spec protocol	ROS subset protocol
Array length	Fixed and variable length	Fixed length
Nested message	Supported	Not supported

のフォーマット制限によって, Narrow Path のためのトンネルでは, データ長を示すデータフィールドを削除することができる．

Table 3.2 にロボットの基本センサ情報として常に Field Computer から OSU へと送信している通信データの構成について示す．これらは先に述べた ROS のサブセットによって記述されている．**Table** 3.2 に掲載されているセンサの選択理由は**第** 3.5 節において後ほど議論する．センサデータをすべて浮動小数点数型で表現してしまうと, 必要な通信量は増加してしまい, Narrow Path の通信帯域制限を超えてしまう．そのため, センサデータを遠隔操縦に必要となる精度に応じて 2 つのグループに分類している．1 つは精度が必要なため, 浮動小数点数型で表現するデータであり , もうひとつは精度が必要でないため 8bit(char) で表現するデータである．精度が必要でないセンサデータでは, あらかじめセンサデータの値域を定義しておき , その区間を 8bit で分割して表現する．これらの分類は例えば, 関節角度は手先やカメラ位置を

Table3.2: Sensor Data and Type in Basic Sensor Data Tunnel

sensor	data type	bit length
joint angles	float	35x32
joint effort	char	35x8
temperature of actuator	char	35x8
odometry origin	float	6x32
estimated ground pose	float	6x32
estimated odometry origin on ground	float	6x32
imu orientation	float	4x32
force/torque sensors	char	4x6x8
difference of absolute and relative encoders	char	33x8
current robot state	char	8
battery voltage	char	3x8

オペレータが正確に知るため精度が必要であるが, モータ温度はオペレータにとって精度の良い情報は必要ではないといった判断に従う.

Table 3.3 に通信に必要なデータサイズをまとめた. **Table** 3.3 に示すように, ロボットの基本センサ情報は浮動小数点数型ですべてを表現し 可変長データを利用するのに比べて, 約 2 分の 1 のデータサイズで表現することが可能となっている.

Table3.3: Comparison of Sensor Data Compression. Data size is represented without Transport Layer.

	Variable Length Array	Fixed Length Array
Float Values	6376 bits	5992 bits
Char Compression	3256 bits	2872 bits

実際にはロボットの基本センサ情報を含めて 6 つの Narrow Path トンネルを Field Computer と OSU の間で通信するために利用している.

- ロボットの基本センサ情報を送るための 1 方向トンネル:

  ロボットの基本センサ情報は 1 Hz の周期で Field Computer から OSU へと送信する. **Table** 3.2 にその詳細を示しているが, これらの情報はオペレータにとって重要である

ため, 常時更新する必要がある.

- カメラパラメータを変更するための 1 方向トンネル:
  照明条件に応じてカメラのゲインや露出を変更するため, オペレータが OSU から Field Computer へ値を更新する必要がある.
- *fc-executive* と *ocs-executive* が通信するための双方向トンネル:
  この双方向トンネルは *fc-executive* と *ocs-executive* が通信するために利用される. 通信データはタスクを示すフラグと, 整数および浮動小数点数型の配列から構成される.
- *Vehicle* タスクの遠隔操縦のための双方向トンネル:
  Vehicle タスクはロボットが障害物に衝突する前にハンドルを回すことによって進行方向を変えなくてはいけないため遠隔操縦に高い更新周期が必要とされる. 実験を通してこの周期には 5Hz 程度必要であるとし, *Vehicle* タスクの間のみ利用する通信経路として双方向のトンネルを用意している. このトンネルでは 5Hz の速度を実現するために, *fc-executive* と *ocs-executive* の間で利用されるメッセージの代わりに, 特殊化された小さなメッセージを利用する,

Fast Path トンネルのためのプロトコル設計

Fast Path の通信帯域は 300 Mbps であり十分に大きいため, これは Narrow Path で考慮したようなデータサイズの切り詰めは必要ない. そのため, Fast Path では ROS のフォーマットとシリアライゼーションをそのまま利用する. ROS のデータフォーマットをそのまま利用するため, Field Computer の ROS ネットワークの中で OSU で利用したいトピックを指定することでそれらのトピックが透過的に OSU 側で利用できるような構成を目標としてプロトコルを設計する.

Fast Path トンネルは Field Computer で実行される送信部と OSU で実行される受信部から構成される (**Fig.** 3.4). 送信部は以下の手順に従って Field Computer の ROS ネットワークからデータを OSU へと転送する:

1. Field Computer の ROS ネットワークの中で, OSU において利用したいトピックを Subscribe する. 代表的なものとしてはステレオカメラの画像や回転レーザの 3 次元点群である.
2. 得られた複数のデータを一つの ROS のメッセージに結合する.
3. 結合されたメッセージを ROS のシリアライズ機能を利用してバイナリ列に変換する

4. バイナリ列を MTU を満足するようにパケットに分割する

その一方で受信部は以下の手順に従ってデータを OSU の ROS ネットワークに再構成する:

1. 送信部から送られてくるパケットを集める
2. パケットを結合することでバイナリ列を復元し, ROS のシリアライズ機能を利用して複数データが結合された一つのメッセージを得る
3. 結合されたメッセージをそれぞれ分割し, OSU の ROS ネットワークに Publish する

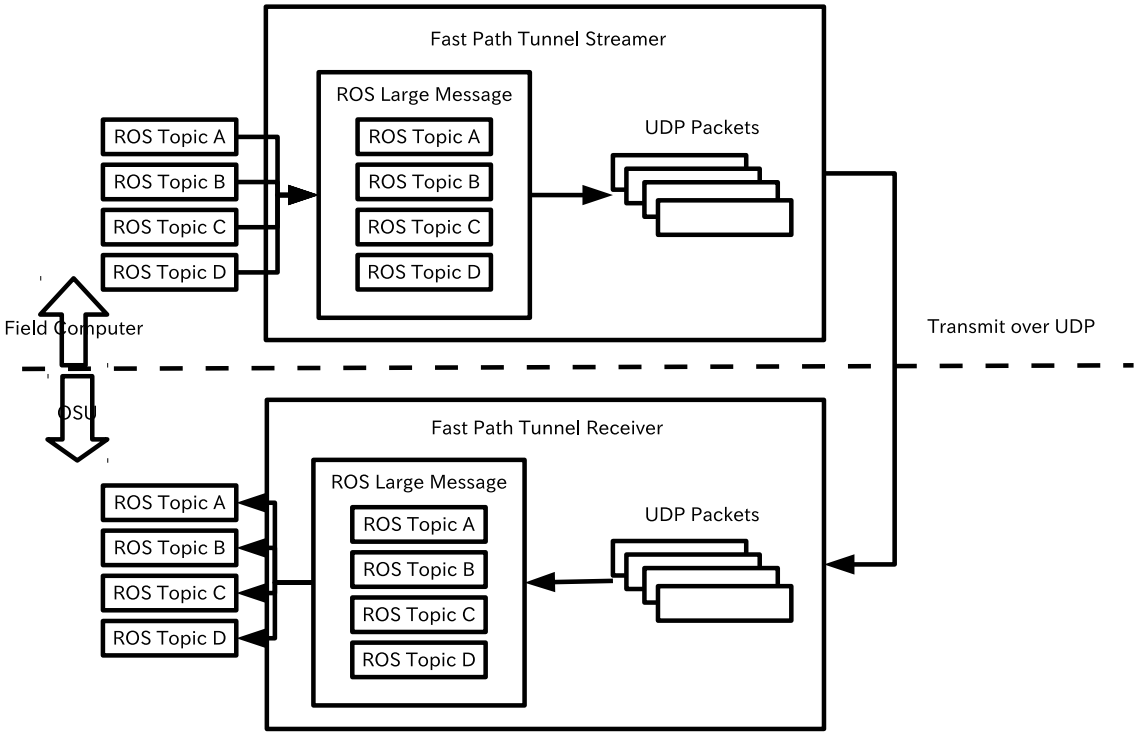


Fig3.4: Overview Diagram of Fast Path Tunnel and Structure of UDP Packet

Fig. 3.5 に Fast Path トンネルで利用される UDP パケット構造を図示する. UDP パケットは 12 バイトのヘッダを持つ:

- *Sequence ID*:

  Sequence ID は送信部が一つメッセージを送り終わるたびに 1 つずつ増加する ID である.

- *Packet ID*:

  Packet ID は一つのメッセージが分割されたパケットそれぞれがユニークにもつ ID で

ある. その値域は 0 から $N-1$ である. ここで N は *Packet Num* に等しい.

- *Packet Num*:

  Packet Num は一つのメッセージが分割されたパケット の数を 示す.

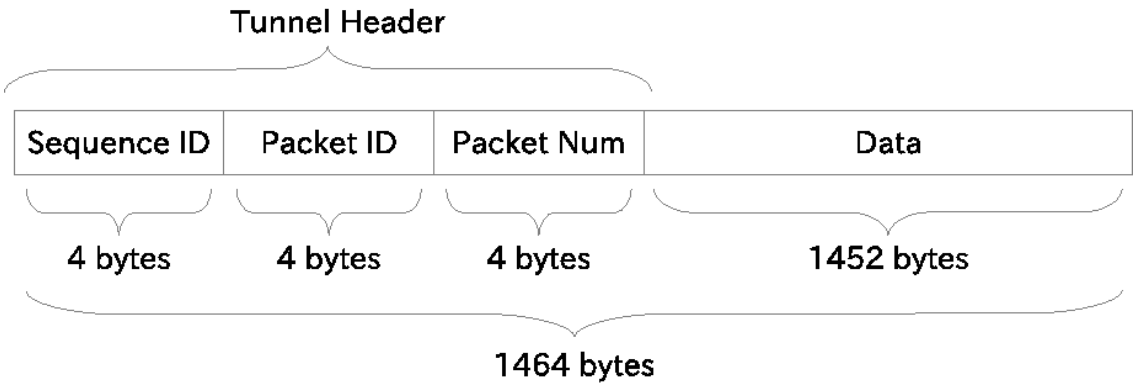


Fig3.5: Packet Strucure of Fast Path Tunnel

3.5　遠隔操縦およびセンサデータ監視のためのユーザインタフェイスの設計

3.5.1　センサデータの可視化と異常監視のためのユーザインタフェイス

センサデータを常に監視することはオペレータにとって重要なタスクである. というのも, 予期しないエラーが遠隔操縦中に発生した場合, 様々なセンサから統合的に状況を判断することがオペレータには求められるからである. その一方で, ロボットに装備されているセンサの数は多く , それらを効率的に可視化することは難しい.

まず, 遠隔操縦タスクにおいてオペレータにとって重要なセンサを以下に列挙する:

- 関節角度と力センサ:

  これらはロボット の状態を知るために最も重要な内界センサである. 関節角度に関してはロボット のエンド エフェクタの位置を知るために精度良い値が必要である. 一方力センサは環境との接触状態を知るために有用であり , この用途に限っては精度は重要ではない.

- ステレ オカメラと回転レーザから得られる 3 次元点群データ:

これらのデータは環境の 3 次元構造を把握するために重要なセンサデータである．ステレオカメラから得られる 3 次元点群は精度の面ではレーザに劣るが, 同時にステレオカメラから取得できる色情報を持っているため, オペレータが環境を把握するために有用である．またレーザはステレオカメラに比べて画角が広く, 遠距離のデータも信頼性高く所得することができる．従って, オペレータが環境を広く把握するために有用である．

- 魚眼カメラとステレオカメラのカメラ画像:

  3 次元点群に加えて, カメラ画像はオペレータが環境を把握する上で重要である．ステレオカメラに加えて魚眼カメラを利用することで, ロボット周囲の環境を一度に広く把握することができる．

- モータの電流値と温度:

  モータの負荷を知るためには, 温度と電流値が適している．特に温度は重要である．というのも, モータが破損する主な原因は焼損や高温による減磁であり, 温度はこれらに直接影響するパラメータであるためである．温度と電流値は瞬間的な値と同様に時系列での変化が重要である．歩行中など負荷が一時的に高くなる場合においては瞬間的な値は多くなりがちなため, 積算された値の変化が重要なためである．

- ロータリエンコーダによる相対位置とホール素子による絶対位置の差分:

  ホール素子による位置計測はノイズが大きいため, JAXON の実時間制御ではロータリエンコーダによる相対位置を利用している．しかし負荷の高い動作を行った場合, ハーモニックギアの歯飛びが生じてしまうことがある．歯飛びが起きてしまうと, 以後のタスクでエンコーダの零点ずれが生じてしまうため致命的な問題を引き起こす可能性が高い．これを避けるために, ホール素子による絶対位置を利用して再度ロータリエンコーダの零点キャリブレーションを行う必要がある．この必要性をオペレータが遠隔操縦中に判断する必要があるため歯飛びを検出するためのロータリエンコーダとホール素子の値の差分を監視する必要がある．

- バッテリの出力電圧:

  バッテリ残量を知るため, バッテリの出力電圧を知ることは重要である．

Fig. 3.6 にこれらのセンサデータを可視化したセンサビューワを示す．3 次元で可視化可能なものは 3 次元 CG として可視化されている．ロボットの関節角度や力センサ, 3 次元点群がそれに当たる．3 次元 CG として可視化が馴染まないものに関しては HUD (Head Up Display) による線グラフとして, 3 次元 CG の上に 2 次元の値として可視化している．HUD

は「 前面ガラス」の上にデータを重畳して可視化する手法である .「 前面ガラス」は飛行機のコックピットにおけるフロントグラスであり , ビデオゲームでは 3 次元 CG の手前の射影平面である . HUD は操縦者に視界を奪うことなく補助的な情報を掲示する手法であることが知られている [67]. オペレータに対し危険な値であることを知らせるため, 予め指定してあるしきい値を超えると色を変化させる (**Fig.** 3.7). 線グラフとして可視化することの利点は, 値の時系列変化を可視化できることである .

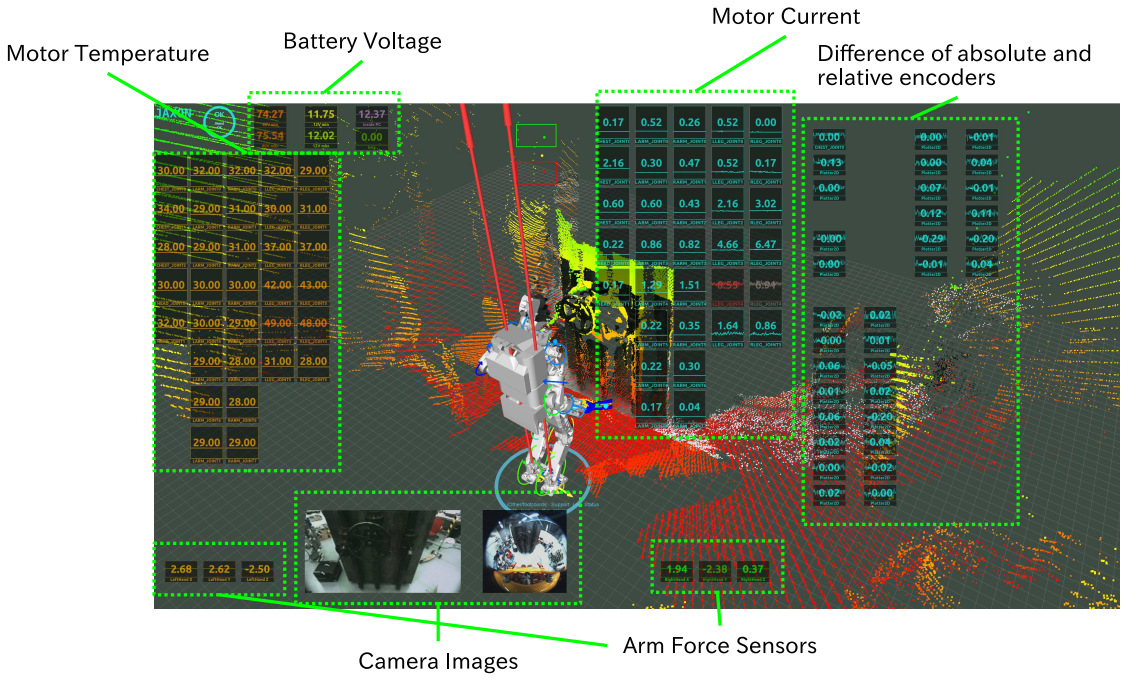


Fig3.6: Sensor Viewer for OSU Operators

Fig3.7: HUD Plot on Sensor Viewer:
Left) Expected value and colored in blue. Right) High value and red color means warning.

3.5.2　遠隔操縦のためのユーザインタフェイス設計

遠隔操縦のためには 2 次元の GUI および 3 次元の GUI の 2 つをオペレータは利用する. オペレータは自律化の抽象度に従った複数の階層に関して, ロボットの行動に介入できる:

- 2 次元 GUI を利用して注視領域を指定する. 指定された注視領域を利用し認識器, その結果を用いた動作計画が実行される.
- 3 次元 GUI で認識結果を変更することによって動作計画の入力を指示する.
- 3 次元 GUI でエンドエフェクタの位置を指定することでロボットの手先位置を直接指定する.

想定通りにタスクが遂行できている場合は, オペレータは 2 次元 GUI によって注視領域を指定することのみによってタスクを遂行していくことができる.

2 次元 GUI は中央にステレオ画像が表示されており支配的なレイアウトを占める (**Fig.** 3.8). このステレオ画像の上で注視領域をオペレータがマウスで指示することでその領域がヒントとして認識器で利用される. **Fig.** 3.8 において, 赤い四角形はオペレータによって指示された注視領域である. ステレオ画像に加え, 2 次元 GUI は魚眼カメラ画像を上部に配置してあり, 魚眼カメラ画像上でも同様に注視領域を選択可能である. 2 次元 GUI にはタイル状に *Valve* や *Walk-To Command* のような, どのタスクに対して取り組んでいるかのコンテクストを指示するボタンと実行, 中止, 動作の緊急停止を指示するボタンが配置されている. また, マニピュレーションにおいて左右どちらの腕を使うか, 立ち位置も含めた動作計画を行うかなどのパラメータを指定するためのチェックボタンを下部に配置している.

3 次元 GUI は主にオペレータが認識や動作計画結果を確認するために利用される. 認識結果は黄色い 3 次元オブジェクトとして可視化され, その結果は同時にロボットの視野画像にも重畳される (**Fig.** 3.9). 緊急時のため, オペレータはロボットの動作を変更することが可能である. そのための 3 次元マーカ [68] を利用して認識結果の位置姿勢が修正可能なインターフェイスを内蔵している. また, 3 次元マーカは緊急時にロボットのエンドエフェクタ位置を直接指定するための用途にも利用される. 3 次元 GUI はオペレータが最も長時間見ることになる画面であるため, 通信状況を常にオペレータに知らせるためのテキスト表示部分を備える (**Fig.** 3.10). テキストはオペレータがセンサデータを確認することを邪魔しないため HUD として可視化され, その内容な最新の通信がいつ行われたかを示す. 通信が期待されている秒数よりも

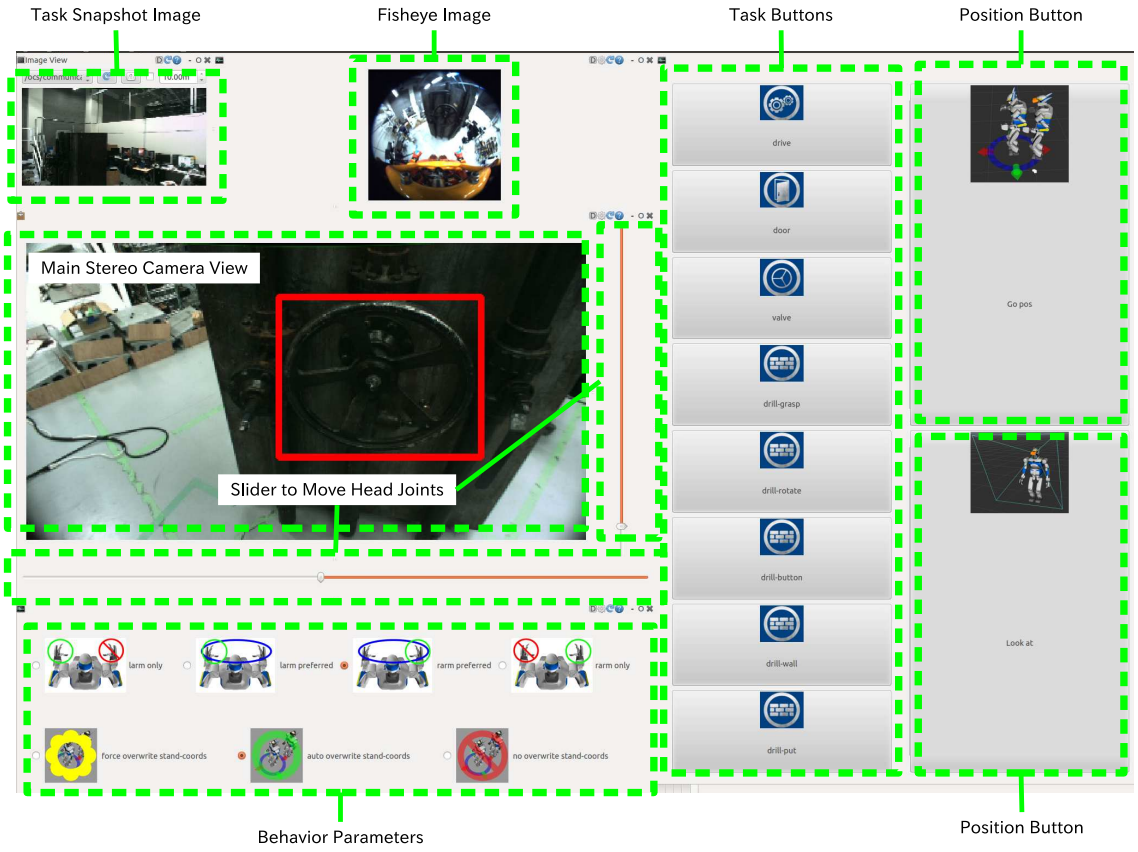


Fig3.8: 2-D GUI to Operate Robot

長い時間得られない場合は, テキストの色が変化し, オペレータへの注意を促す. オペレータは操作のためにマウスやキーボードなどのいくつかの入力インタフェイスを利用することができる (**Fig.** 3.11). 3 次元マウスとゲームコントローラはエンドエフェクタの位置を調整するために利用される. MIDI コントローラ (BEHRINGER 社製 B-Control) は動作のためのパラメータを変更するために利用される. パラメータとしては動作の実行速度や認識されたバルブの半径などを変更する. BEHRINGER 社製 B-Control を利用したのは, スライダがモータ駆動されており, 計算機から動かすことができるためである. この特徴を利用して認識が終わるたびにスライダの位置を認識結果に対応する場所に自動的に動かす.

3.6 DRC 競技会の結果

DRC 競技会の結果, NEDO-JSK チームは 4 点を *Vehicle*, *Door*, *Valve*, *Stair* タスクで獲得した (**Fig.** 3.13).

競技では, オペレータの負担を考慮して, 3 人のオペレータを OCS に採用した (**Fig.** 3.12).

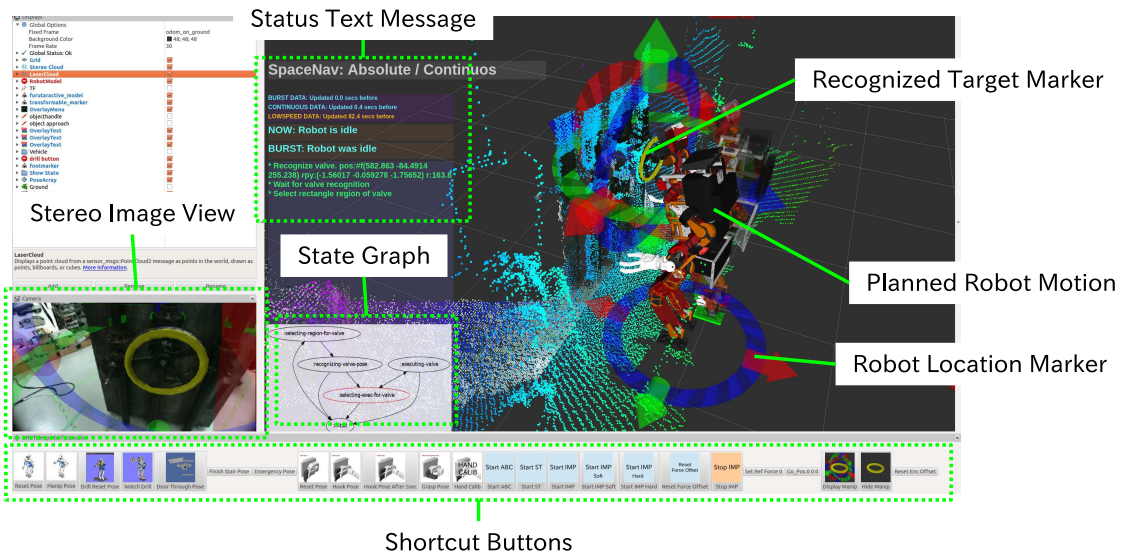


Fig3.9: 3-D GUI to Confirm Recognition Result and Planned Motion.

Fig3.10: Visualization of Network Status Overlaying 3-D GUI

オペレータの負担が増えると, オペレータの注意力が散漫してしまい, 遠隔操縦において致命的なミスが増えてしまうという知見から, オペレータの負担を分散するためこのような構成人数にした. 逆にオペレータの人数が増えると, オペレータ間の会話によるコミュニケーションのコストが増えてしまい, タスク遂行に必要な時間が増えてしまったり, オペレータの集中力が散漫としてしまった. メインオペレータは唯一ロボットを操作することが許されたオペレータである. メインオペレータは 2 次元 GUI, 3 次元 GUI の正面に位置する. 補佐オペレータはメインオペレータの横に位置し, メインオペレータの操作を確認する役目を担っている. 補佐オペレータは全てのタスクにおいてメインオペレータと同様にロボットの操作ができる能力を有する人物を採用した. 監視オペレータは常にセンサビューワを監視することが求められるオペレータである. センサビューワはモータ温度など多くの情報を表示しているため, それらの情報を理解することのみに集中する必要がある. 監視オペレータがセンサ値に異常を発見した際は, すみやかに対処法をメインオペレータに伝え, メインオペレータは異常対応のためロ

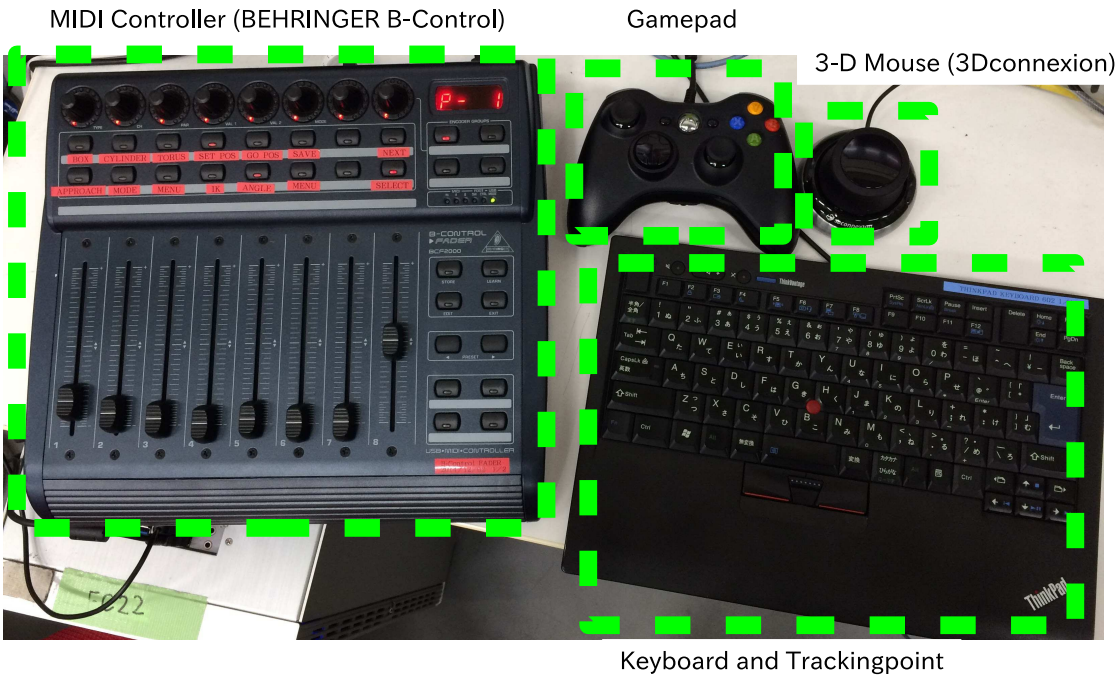


Fig3.11: Input Devices including MIDI Controller, 3-D Mouse, Gamepad and Keyboard

ボットを操作する.

Fig3.12: Three Responsible Operators in OCS

1 日目の競技においては, NEDO-JSK チームは *Vehicle* および *Door* タスクで 2 点を獲得した. *Door* タスク遂行中に, ロボットはドアを開けることに成功したが, タスクは屋外環境で行われたため, 予期しない強風によってドアは再び閉まるという事態が発生した (**Fig.** 3.14). 動作計画器に予め与えてあったマニピュレーション軌道では, 開いた時のドアの角度が強風に

Fig3.13: Run on 6th, June, 2015. (Images from [69].)

対して開き続けるためには不十分であった. このような事態に対応するため, オペレータは予め与えてあったドア軌道に基づいた動作を途中で停止することで割り込み, 3 次元マーカを利用して直接ロボットのエンドエフェクタ位置をさらにドアを奥に押すように指定することでドアを開けることに成功した.

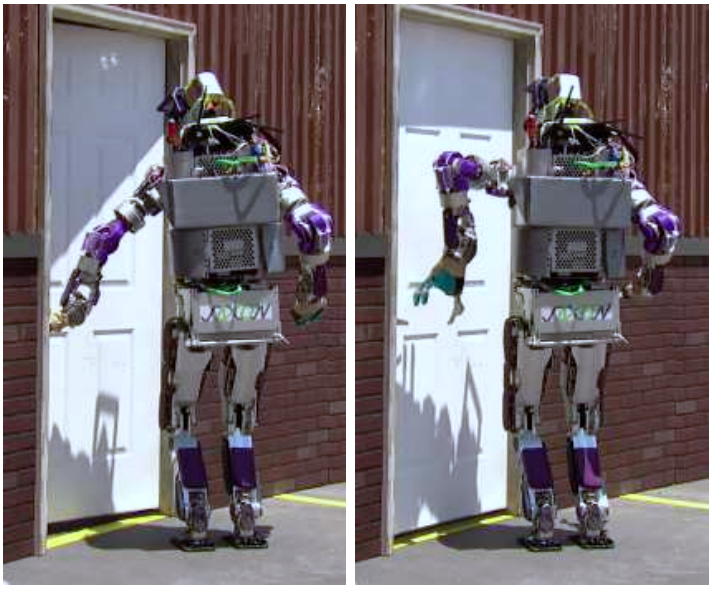


Fig3.14: Opened Door Blew Shut by Unexpected Wind Conditions (Images from [69].)

Door タスクを突破することはできたが, 競技が進み *Valve* タスクの途中でロボットは転倒

してしまった (**Fig.** 3.15). 転倒時の衝撃により , ロボットの腕リンクは競技続行が難しいほど破損した (**Fig.** 3.17). 転倒時, オペレータおよびシステムは以下のような手順で進んでいた:

Fig3.15: Falling During Valve Task on the First Day (Images from [69].)

1. オペレータはステレオ画像からバルブの位置を指示するため注視領域を選択した.
2. 認識器はバルブの位置・姿勢・半径を注視領域に従って検出した.
3. 動作計画器は立ち位置を含めた動作列を計画. バルブが左側にあったため, 計画された立ち位置は左側への移動を必要とした.
4. オペレータは認識結果と動作結果を 3 次元 GUI 上で確認した.
5. ロボットは計画された立ち位置へ移動した.
6. ロボットは計画されたマニピュレーション軌道を実行した.
7. 腕の動作を実行中, 右腕がバルブに衝突し , ロボットは転倒した.

オペレータは移動前にバルブの認識結果はタスク遂行に十分な精度が出ていたと判断し , 計画された動作列・立ち位置も問題ないと判断した (ステップ 2-4). しかしながら , ロボットのオドメトリが十分な精度ではなく , 目標とした立ち位置よりも前に移動してしまった. そのため, 事前に計画していた腕の動作を実行することでバルブと腕が衝突してしまった. 衝突した場所は腕の末端に設置された力センサよりも上であったため, インピーダンス制御による環境との衝突緩和が機能せず, 結果的にロボットの転倒へとつながった.

競技二日目, NEDO-JSK は *Vehicle*, *Door*, *Valve*, *Stair* を遂行することで 4 得点を得た.

Drill タスク遂行中, ロボットは転倒してしまったが損傷は少なく , タスク続行は可能と判断し開始 59 分で *Stair* を遂行した. 競技二日目の *Valve* タスクでは, 初日の *Valve* タスク中での転倒があったため, オペレータはロボットが立ち位置を移動した後に (ステップ 6) 動作に割り込み停止させることで, オドメトリのエラーを 3 次元 GUI で確認した. オペレータは 3 次元マーカを利用することで, オドメトリのエラーを修正した動作計画を再度行うことで *Valve* タスクを遂行した. しかし , *Drill* タスク中にロボットは転倒してしまった. 転倒の原因を追求することは難しいが, これは低レイヤのハードウェアもしくは電装系の問題だったと考えられる. 初日の転倒で腕から地面と衝突したのとは異なり , ロボットは背中から転倒した. ロボットの背中部には衝撃吸収のためのクッションがつけてあり , 転倒時の損傷は軽度であった.

初日, 二日目の競技を通して, オペレータはセンサビューワを通してアクチュエータの温度が予期せず上昇していることを確認した. これは長距離の歩行時に内力が溜まってしまったためだと考えられる. 監視オペレータがアクチュエータの過熱を観測した場合は, メインオペレータに対してその場でロボットを足踏みさせて内力を逃がすよう指示をすることで問題を回避した. 監視オペレータはアクチュエータの過熱と同様に, ハーモニックギアの歯飛びを観測した. 監視オペレータが歯飛びに気づくと , メインオペレータは一度ロボットを停止させ, ホールセンサを利用して再度関節角のキャリブレーションを行った.

3.6.1　競技における遠隔操縦システムの必要時間の考察

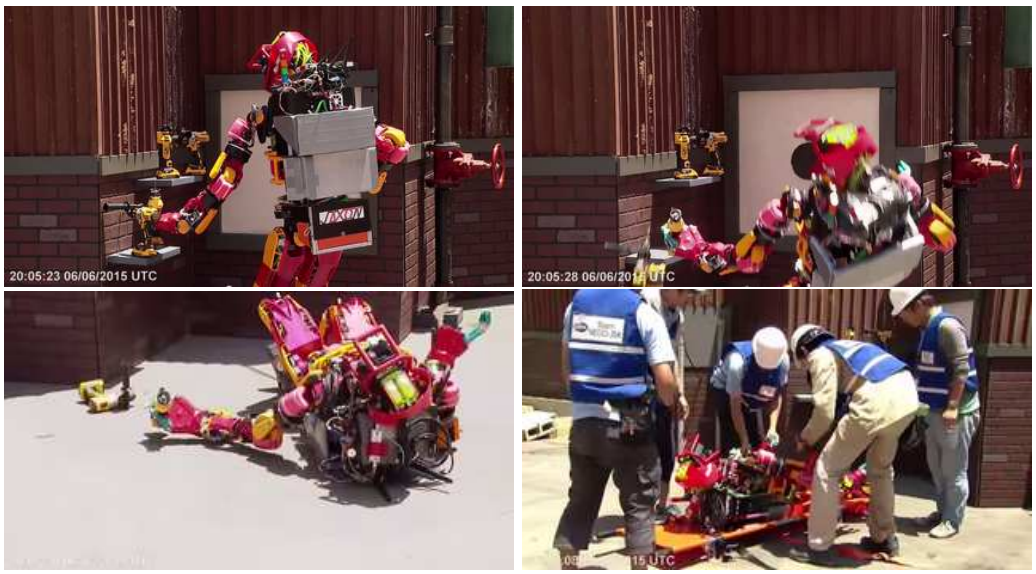


Fig3.16: Falling During Drill Task on the Second Day (Images from [69].)

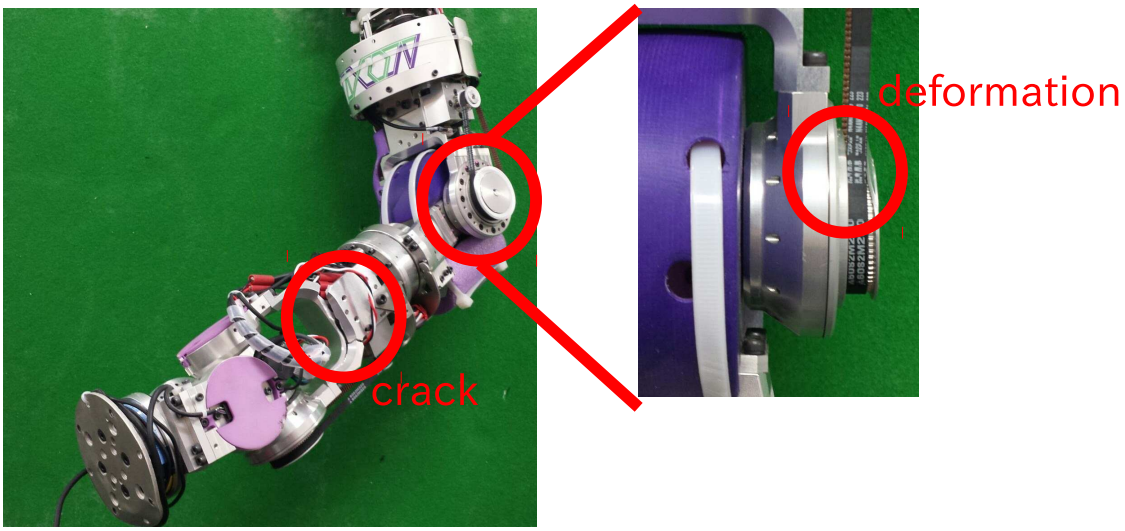


Fig3.17: Damage Caused by Falling on the First Day

DRC 競技会において NEDO-JSK チームの競技が時間別に見ていくとどのようにロボットが動いていたかを **Fig.** 3.18 に示す. ここで注目すべきは, 競技のタスク全体を通してどのようなロボットの行動に時間が割かれていたかである. **Fig.** 3.19 に各行動に費やした時間の比較を示す. moving head は視点を変更するために頭を動かした時間, handling は車のハンドルを操作する時間, driving は車が前進している時間, manipulation は腕を動かしていた時間, walking はロボットが歩行に費やした時間, stop はロボットが停止ていた時間である. reset は転倒等によるペナルティの時間である. moving head, handling, driving, manipulation, walking は実行のための時間のため, **Table** 2.2 におけるパラメータを利用するとその合計は $\sum_i t_{E_i}$ と等しい. また, **Fig.** 3.20 にロボットが停止していた時間 $\sum_i (t_{P_i} + t_{M_i} + t_{O_i})$ とロボットが動いていた時間 $\sum t_{E_i}$ の比較を示す. ただし, ここで t_{O_i} はタスク T_i のためのオペレータが操作していた時間である. これらの分析に従うと, ロボットは動いている時間よりも停止している時間のほうが長いという事がわかる.

$$\sum_i t_{E_i} < \sum_i (t_{P_i} + t_{M_i} + t_{O_i}) \tag{3.1}$$

遠隔操縦システムでは, **Fig.** 3.21 に橙で示したブロックで示すように, 認識の前, 動作計画の前にオペレータの操作が介入しうる. これは **Fig.** 2.5 に示したロボットシステムのパイプライン表示と比較すると, オペレータの操作が介入する場合, その分だけシステムの停止時間が伸びてしまうということを意味する. この停止時間の中には認識・動作計画のために費やした計算時間, オペレータの操作時間, およびオペレータが通信を待っている時間が組み合わされ

たものである．分析に利用した競技結果においては, オペレータはロボットの転倒を恐れるため必要以上に慎重な遠隔操作をしてしまった. これは第 2.4.1 節の議論と対応させると, 品質 q を向上させようとした結果必要時間 t が大きくなってしまったという事である．そのため, 8 タスク全体で 1 時間という制約を満たすことが出来なかった．本章で示した遠隔操縦システムでは, 認識処理, 動作計画および実行に関して品質 q と必要時間 t を明示的に変化させることが難しかった．そのためオペレータは, オペレータ自身の挙動で品質 q と必要時間 t を制御していた．本論文が目指す必要なタスク時間に対してパラメータを変化させるようなロボットシステムであれば, 必要以上に慎重な操作をするオペレータに変わってシステムが適した必要時間と安全性に従ってロボットを動かしていくことが可能であった．

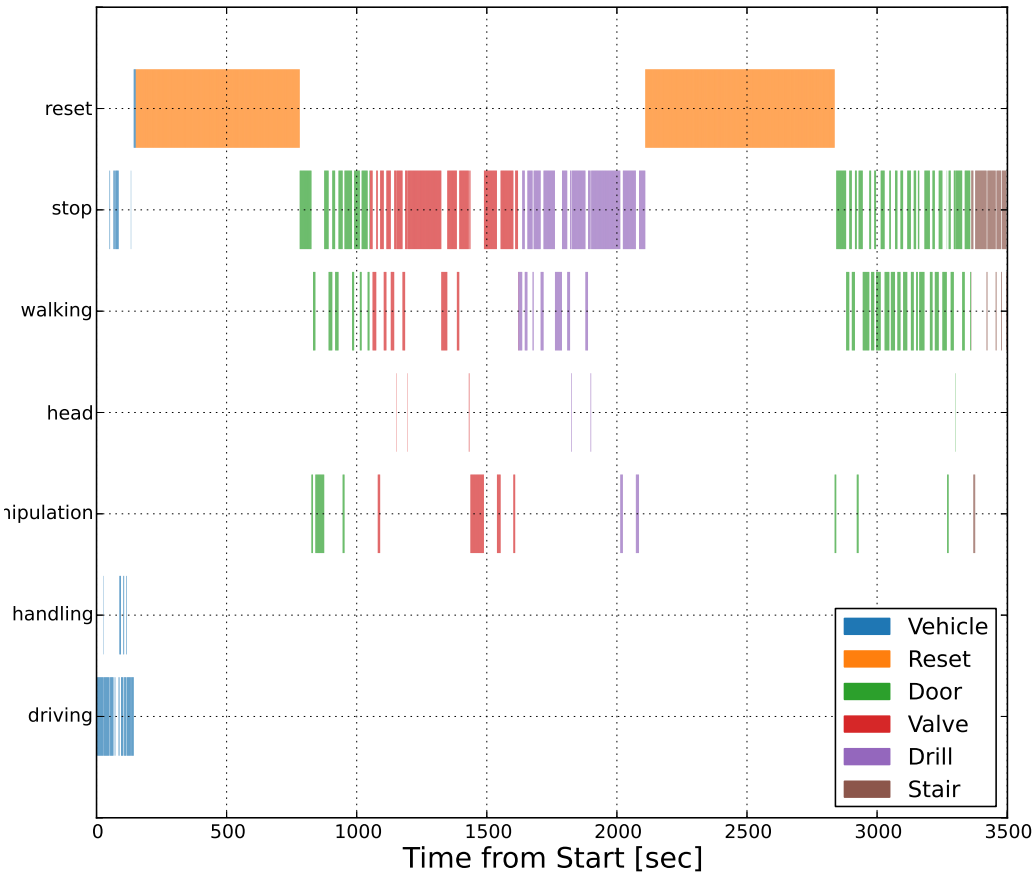


Fig3.18: Timeline of DRC Run of NEDO-JSK Team, 2nd Day

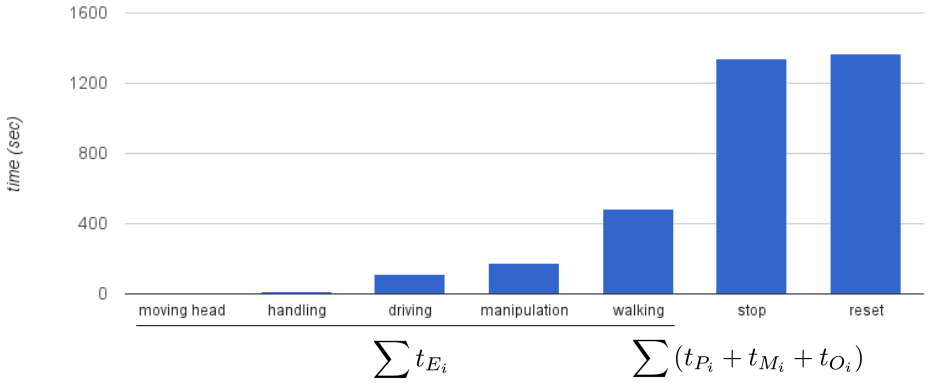


Fig3.19: Comparison between Duration of Each Robot Action.

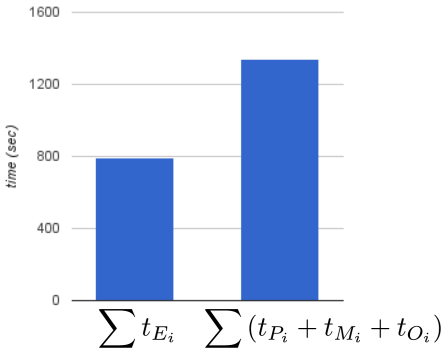


Fig3.20: Comparison between Duration of Moving $\sum_i t_{E_i}$ and Idle $\sum_i (t_{P_i} + t_{M_i} + t_{O_i})$

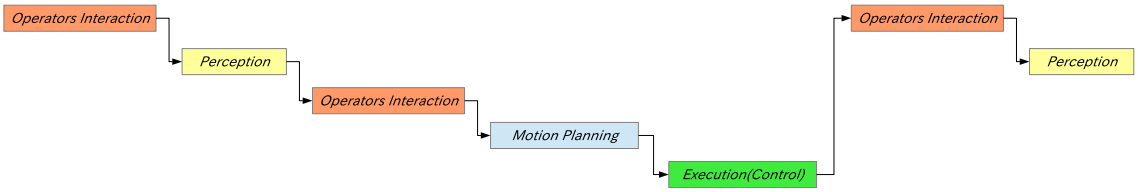


Fig3.21: Hazard in Teleoperating Robotic System in Pipeline Manner

3.7　おわりに

本章では, 等身大ヒューマノイドの遠隔操縦システムの構成法について示した. その中でも特に, ユーザインタフェイスの設計を中心に述べた. 遠隔操縦システムの特徴は以下のようなものである. 1) Supervised Autonomy を実現するため, オペレータからの簡易な指示によってロボットが動作するシステム. 2) 予期しない状況に対応するための低レイヤのセンサデータ可視化. 3) システムの自動化レイヤに対応付けたユーザインタフェイス. 4) 帯域制限や切断が生じるような通信環境でも機能可能な遠隔通信機能. 本章で示した遠隔操縦システム DRC 競技会で NEDO-JSK チームおよび HRP2-Tokyo チームによって利用された.

実際の競技を通して, 本節で示した遠隔操縦システムはいくつかの予期しない状況に対して対応することが可能であった. 特に動作実行中にオペレータがシステムに割り込み停止させる機能は, 複数自動化レイヤに対応したユーザインタフェイスを用いることで一日目の *Door* タスク, 二日目の *Valve* タスクにおいてよく機能した. また, 低レイヤのセンサデータ可視化とそれ専用のオペレータを用意することで, 歩行中の予期しない負荷の増大に対応することが可能であった.

しかしながら, ソフトウェアおよびハードウェアの両面において, ロボットの転倒に対しては十分に機能しなかったと言える. 転倒に対して強いシステムを作るため, 以下の 3 点に関して特に改良が必要である.

1. 転倒しても破損しないようなハードウェアに改良する.
2. 転倒時に足を踏み出すと言った挙動を制御レイヤに組み込むことで, 転倒を防ぐような制御ソフトウェアに改良する.
3. より実時間性の高いユーザインタフェイスを実現可能なように, 通信システムを改良する.

特にユーザインタフェイスに関しては, 実時間性の高い視覚センサデータをオペレータが観測可能なようにするという改良点が考えられる. 本システムでは Narrow Path を通して視覚データを送るようなことをしていない. Narrow Path は帯域制限が厳しいが, 圧縮率の高い画像をオペレータが常に監視することが出来れば, オペレータは転倒につながるような環境との接触を転倒前に予期することができた可能性がある.

また, 1 日目と 2 日目の転倒による破損の違いは興味深いアプローチを示唆している. 1 日

目は左腕から転倒することで, ロボットのリンクに致命的な損傷を発生させてしまった. 一方で 2 日目の転倒では, ロボットの背中から転倒したため, ハードウェアに致命的な破損は生じず, その後に *Stair* タスクを完遂することができた. これは転倒部位によってロボットのハードウェアに与える損傷は異なることを意味している. したがってロボットが転倒を始め, 制御によって転倒を防止するのが難しいような状況であっても, 転倒部位を反射的に制御することが出来れば, ハードウェアへの損傷を軽くすることが可能であろう.

また, 競技中にどのようなタスクに時間を費やしていたかを調べることで, ロボットの停止時間が全体の半分以上を占めるような結果であったことがわかった. これは競技会という特性上, オペレータがロボットの操作や確認手順で慎重になってしまい多くの時間を費やしてしまったということが大きな理由であった. それが原因となり, 8 タスク全体で 1 時間という制約を満たすことが出来なかった. 本論文が目指す必要な時間に対してパラメータを変化させるようなロボットシステムであれば, 必要以上に慎重な操作をするオペレータに変わってシステムが適した必要時間と品質に従ってロボットを動かし, 与えられた時間内に必要なタスク全体を完遂することが可能であったと言える.

第 4 章

認識器と動作計画器の協調的サブスケジューリングによる足配置計画法

4.1 はじめに

本章では, 動作計画器と認識器を協調的に動作させることで動作計画の進捗を利用して認識の注視領域を制御する足配置計画法を提案する. これは第 2.5.4 節において特殊な実行モデルとして述べた認識器と動作計画器の協調的サブスケジューリングの具体的な実装にあたる (**Fig.** 4.1).

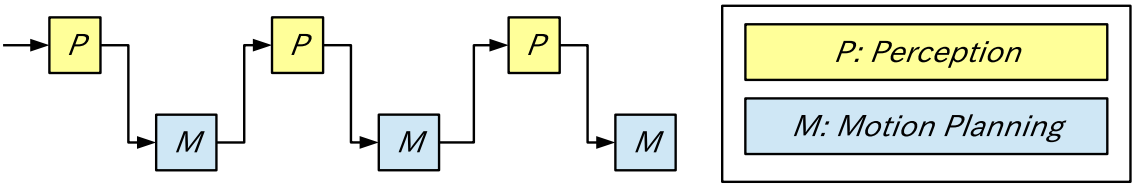


Fig4.1: Overview of Footstep Planner Integrated with Collaborative Sub-Scheduling of Perception and Motion Planning

本章で提案する足配置計画法の特徴は, 通常環境構造を認識しモデル化した後に動作計画を行うところを, 動作計画と同時に認識処理を進めていくことで動作計画の中で必要となる領域のみを認識・モデル化することである. このような構成にすることで, 計算時間が必要とされる環境構造を動作計画に適した形にモデル化する認識処理を事前に実行する必要がなくなる. その結果, センサの視野全体に対して認識処理を走らせてしまう場合と異なり, 認識処理は計画される動作の複雑さに依存した計算時間を必要とする. 評価制御機構を備えたロボットシステムの中では, 動作計画の計算コストを制御するパラメータによって必要となる認識の計算コストも制御することが可能なものとして位置づけられる. 本章ではこのような足配置計画にお

いて動作計画の計算コストを制御する代表的なパラメータとして量子化刻み幅についても言及する.

4.2 関連研究

ロボット研究において, 経路計画は重要な分野として研究が盛んにおこなわれている [70, 71]. 二足歩行型のヒューマノイドロボットの分野では, 転倒することなく二足歩行を実現するための制御法についての研究が多くなされている [72, 73, 74]. 未知の環境下において脚型ロボットを目標値に移動させるためには, 歩行のために動作計画と環境の認識機能を統合することは必要不可欠である. また, 二足歩行型のヒューマノイドロボットは移動台車型のそれと比べて, 段差や障害物を乗り越えた移動経路を実現することができる. これは足配置計画と呼ばれる問題領域であり , 二足歩行型のヒューマノイドロボットにおいて経路計画を行うためには重要な手法である.

このような足配置計画問題を解くためには, グラフ探索ベースな手法が用いられる [75, 76]. グラフ探索を用いた手法では, 離散的な足配置集合を利用して, 探索ツリーを展開してくことで, 目標値に至る軌道を計算する. Xia らはサンプリングベースなグラフ探索手法である RRT[77] を用いて, 一般的なグラフ探索手法よりも高速に足配置計画問題を特殊法を示している [78]. Hornung らは, D* Lite という A*を拡張したインクリメンタルな探索手法を用いて高速に足配置計画問題を解く手法を示している [79]. 一方, 認識との統合という観点からでは, 環境の正確な 3 次元形状モデルを利用したアプローチが示されている [80, 81]. 足配置計画問題によらず, 動作計画手法は環境の正確な 3 次元形状モデルをその前提とすることでロボットの動作をシミューレーション可能な環境モデルの中で探索を行うという事が一般的な枠組みである. Chestunutt らは 2 足型ロボットにピッチ方向に回転する 2 次元レーザを取り付け, 事前に正確な 3 次元形状モデルを持たずに足配置計画を行う手法を示した [82]. この手法では, 足が配置可能な領域は一定の平面をもつ領域であるという仮定を用いている. Maier らは障害物を検知するための 3 次元グリッドマップを kinect[83] を利用して生成する手法を示した [84]. レーザと比較して, kinect のような深度カメラを利用する利点は 3 次元点群を高速に取得できることである. その一方で欠点としては 3 次元点群の精度がレーザに比べて劣ってしまうことである. Fankhauser らは 4 脚ロボットの足配置計画問題において, ロボットを中心とした Heightmap とよばれる 2.5 次元の 3 次元点群データ構造を kinect を用いて高速に生成する手法を示した [85]. Nishiwaki らは 2 足歩行ロボットのための統合ナビゲーションシステムを , 2

次元レーザを用いた環境のモデル化, 足配置計画, 転倒防止のためのバランス制御, 歩行軌道を指示するためのユーザインタフェイスを統合したシステム構成法について示した [86]. これらの足配置計画に関する既存研究では, 認識処理と計画処理は分離されており, 本章で提案するような認識と計画を協調的に統合した最適化手法については述べられていない.

一方マニピュレーションのための動作計画においては, 認識と計画を協調的に統合する手法が用いられている. マニピュレーションのための動作計画で利用される認識処理は, 3 次元形状モデルによって表現される環境とロボットの形状モデルを用いた衝突判定である [77, 87]. 衝突判定は動作計画における探索ツリーの展開中に実行され, 高速化のため探索空間全体では衝突判定は行わない. 近年では 3 次元点群が取得可能なセンサの利用が進み, 環境を 3 次元点群を利用した衝突判定が利用されている [88, 89, 90]. 本章で述べる認識と計画を協調的に統合した足配置計画法は, このような動作計画における動作計画と認識処理の統合をより高度な環境認識処理へと拡張させたものとして位置づけることができる.

本章では, 動作計画器の進捗を利用して認識器の注視領域を制御する足配置計画法を提案する. その特徴は, 探索時に環境が連続した平面で構成されているという近似を導入することによって, 動作計画に必要となる認識処理を可能な限り遅延することで認識処理を削減するというものである. 具体的な探索手法としては離散グラフの最良優先探索を用い, 認識の事前処理として SLAM を利用した環境の Heightmap モデルを利用する (**第** 4.5 節). **Fig.** 4.2 に本章で述べる足配置計画全体の構成図を示す.

4.3　離散グラフ探索による足配置計画法

本節では離散グラフ探索による足配置計画法について説明する. 多脚ロボットの足配置計画は離散的なグラフ探索としてモデル化される (**Fig.** 4.3). 離散的なグラフ探索として足配置計画をモデル化するには, 足配置がグラフのノードとなる. 探索のコストは必要となる歩数とし, それは初期ノードからのノード深さと同値である.

ノード N が保持する状態ベクトルは **eq.** 4.1 で表される.

$$N = (\boldsymbol{p}, \boldsymbol{q}, l) \tag{4.1}$$

eq. 4.1 において, $\boldsymbol{p}$ は足裏の位置, $\boldsymbol{q}$ は足裏の姿勢, l は左もしくは右を示すフラグである. グラフ探索によってこれらの状態量を目標ノードに近づけ, 目標に至るまでに通過するノードが足配置計画の出力となる. l は左右の足を示すフラグであり, ノードを展開していくたびに反転させていく. そのため, 実際にグラフ探索により探索されるパラメータは $\boldsymbol{p}$ および $\boldsymbol{q}$ の 2 つ

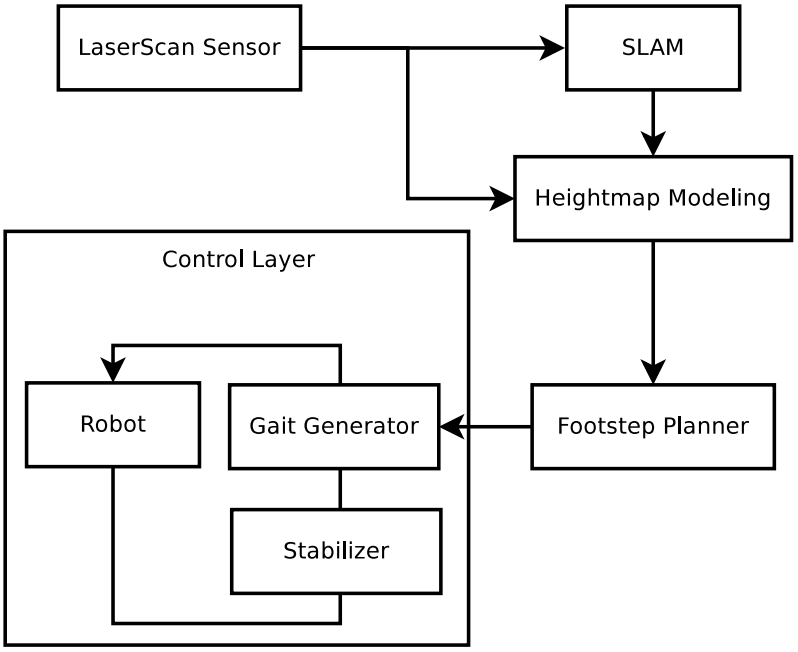


Fig4.2: Overview of Footstep Planning System

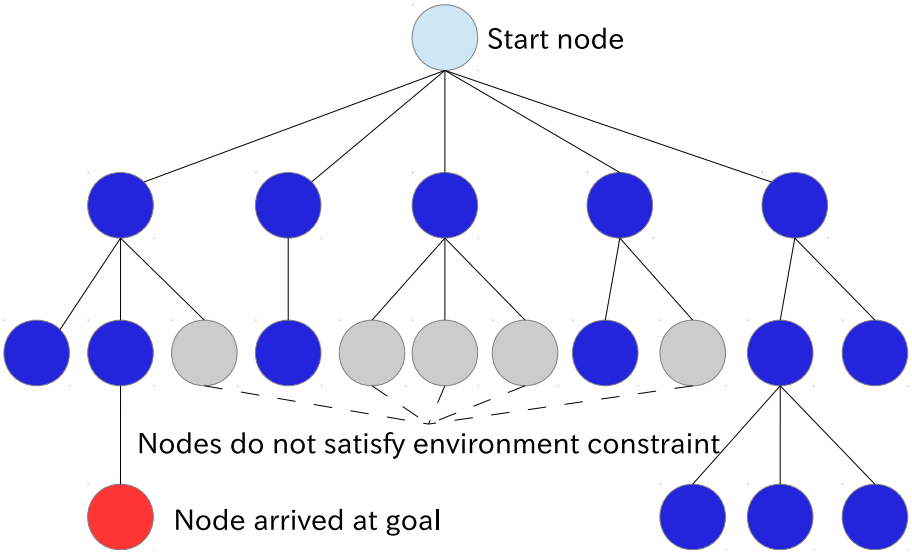


Fig4.3: Graphical Modeling of Footstep Planning

のベクトルである.

Alg. 2 に最良優先探索による足配置探索アルゴリズムを示す. ここで S は初期足配置ノード, G は目標足配置姿勢, O, C はそれぞれオープンリスト, クローズドリストである. $expand(N)$ はノード N から次に到達可能な足配置ノードを返す関数, $pop(O), push(succ, O)$ はそれぞれオープンリストからの pop 操作と push 操作である.

Algorithm 2 Footstep Planning with Best-First Search

```
1: O ← {S}
2: C ← ∅
3: while O do
4:    N ← pop(O)
5:    if N is close to G then
6:       return path(N)
7:    end if
8:    succ ← expand(N)
9:    push(succ, O)
10:   C ← N + C
11: end while
```

オープンリスト O は優先度付きキューとして実装され, 優先度はオープンリストにノードを追加する際に計算される. ノード n の優先度は **eq.** 4.2 に従い計算される.

$$f(n) = g(n) + h^*(n) \tag{4.2}$$

ここで $g(n)$ は初期ノードからノード n へのコストである.

4.3.1　二足歩行ロボットの探索基本オペレータ群

Alg. 2 に示したグラフ探索の中で, ノード N が次に到達可能なノードの展開方法は **Fig.** 4.4 のようになる. これは支持脚に対して, 次に遊脚がどの位置に足を踏み出し可能であるかを示す連続的な領域である. この領域内の足配置は以下の条件を満足するものである.

1. 歩行動作中に左右の足が衝突しない
2. 歩行動作中に転倒しない

グラフ探索の中でノード展開にこの領域を利用するため, **Fig.** 4.5 のように離散的な足配置として近似する. このとき, 量子化の刻み幅を小さくした場合, グラフ探索時のノード数が増大

し, 結果として探索にかかる時間が増えてしまう. そのため, 探索時に充分な数だけ離散化された基本オペレータを定義する.

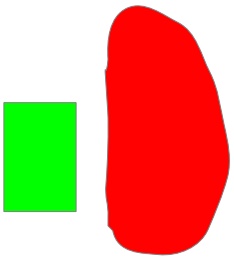


Fig4.4: Continuous Region of Footstep Successors

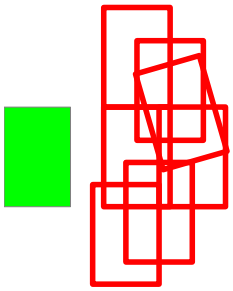


Fig4.5: Discrete Footstep Successors

4.3.2　足配置計画のためのヒューリスティック関数

すでに述べたように, この足配置計画では優先度付きキューを用いてグラフ探索を行う. そのためのヒューリスティク関数は以下のように表現される [91].

$$h^*(N) = w_d D(N) + w_\rho \rho(N) + w_g \chi(N, N_g) \tag{4.3}$$

ここで, N はコンフィギュレーション空間におけるノードである. $D(N)$ はノードの探索深さ, $\rho(N)$ は回転するような経路に対する罰則項, $\chi(N, N_g)$ は現在のノードから目標までの見積りコストである. w_d, w_ρ, w_g はそれぞれの重み係数である. 本研究では, $\rho(N)$ は用いず, $w_\rho = 0$ とする.

eq. 4.3 において, 重要となるのは, 見積りコスト関数 $\chi(N, N_g)$ の選定である. 本論文では, 見積りコスト関数の評価のため, 以下の 3 種類の評価関数を用意した. $L(N)$ はステップ数を単位として持つため, これらは全ての評価関数はステップ数を返す. ここで, $\boldsymbol{p}, \boldsymbol{p}_g$ はそれぞれノード N, N_g の位置ベクトル, $\boldsymbol{e}, \boldsymbol{e}_g$ はそれぞれノード N, N_g の x 方向の単位ベクトルである. また, $l_{\max}, \theta_{\max}$ は基本オペレータ群から計算されるオペレータの最大並進移動量と最大回転移動量とする.

1. straightDistance 現在のロボットの位置から目標座標までを直線で結び, その距離を歩幅の最大値で割ったものである ([91]).

$$\text{straightDistance}(N, N_g) = \frac{|\boldsymbol{p} - \boldsymbol{p}_g|}{l_{max}} \tag{4.4}$$

2. stepCostDistance straightDistance の評価に加え, 回転成分も考慮したものである (**Fig.** 4.6). 回転成分は, 現在のノード N から目標ノード N_g の方向を向くために必要な角度 $\Delta\theta_1$ と, その方向からさらに目標ノード N_g と同じ姿勢になるために必要な角度 $\Delta\theta_2$ を考慮する. w_s, w_r をそれぞれ直進コスト, 回転コストの重みとして以下のように定式化される.

$$\text{stepCostDistance}(N, N_g) = w_s \frac{\Delta l}{l_{\max}} + w_r \frac{|\theta_1| + |\theta_2|}{\theta_{\max}} \tag{4.5}$$

$$\Delta l = |\boldsymbol{p} - \boldsymbol{p}_g| \tag{4.6}$$

$$\Delta\theta_1 = \arccos(\boldsymbol{e} \cdot \boldsymbol{d}) \tag{4.7}$$

$$\Delta\theta_2 = \arccos(\boldsymbol{e}_g \cdot \boldsymbol{d}) \tag{4.8}$$

$$\boldsymbol{d} = \frac{\boldsymbol{p}_g - \boldsymbol{p}}{|\boldsymbol{p} - \boldsymbol{p}_g|} \tag{4.9}$$

このヒューリスティック関数は, 以下のような軌道を規範としている.

（ a）ゴールに向かってロボットを回転させる.

（ b）直進によってゴール位置まで進む.

（ c）その場回転によってゴール姿勢を満足する

このような軌道は条件によっては実際の最短ゴール到達ステップ数よりも大きい値を返す可能性がある. そのため, A*の最短経路成立条件である $h^*(N) < g(N)$ を満たさない可能性がある. これは計画される足配置の軌跡が最短ではない場合が存在しうるということである. 本論文ではこの stepCostDistance をヒューリスティック関数として用いる.

4.3.3　2 次元足配置計画の拡張による 3 次元足配置計画

本小節では, 足配置計画方の 2 次元での実現方法を述べ, それを 3 次元へと拡張する方法について述べる. 3 次元での足配置は, 2 次元から 3 次元への射影面を考えることで, 2 次元での足配置計画と同様に考えることができる.

3 次元足配置計画における状態量と射影操作

2 次元の足配置計画における状態量は **eq.** 4.1 より , 2 次元座標の位置 x, y と回転角 θ を用いて

$$N = (x, y, \theta, l) \tag{4.10}$$

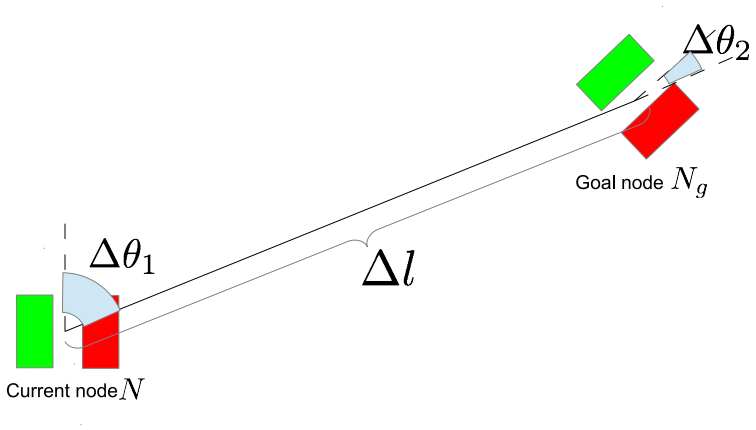


Fig4.6: stepCostDistance Heuristic Function

のように表現できる. 基本オペレータ群 T も 2 次元座標の変換として定義できるので,

$$T_l = (t_{0,l}, t_{1,l}, \cdots, t_{k,l}) \tag{4.11}$$
$$T_r = (t_{0,r}, t_{1,r}, \cdots, t_{k,r}) \tag{4.12}$$
$$t_{i,l} = (x_{i,l}, y_{i,l}, \theta_{i,l}) \tag{4.13}$$
$$t_{i,r} = (x_{i,l}, -y_{i,l}, -\theta_{i,l}) \tag{4.14}$$

$t_{i,l}, t_{i,r}$ は左右で対称なため, どちらか片方のみを定義すれば十分である. あるノード N から到達可能なノード N_i' は

$$\theta_i' = \theta + \theta_i \tag{4.15}$$
$$\boldsymbol{x}_i' = R(\theta_i)\boldsymbol{x} \tag{4.16}$$

となる. ただし,

$$\boldsymbol{x}_i' = \begin{pmatrix} x_i' \\ y_i' \end{pmatrix} \tag{4.17}$$
$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.18}$$
$$R(\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} \tag{4.19}$$

とした. 一方で 3 次元空間における足配置計画の場合, ノード N が保持する状態量は

$$N = (x, y, z, \phi, \psi, \theta, l) \tag{4.20}$$

となる. ϕ, ψ, θ はそれぞれロール角, ピッチ角, ヨー角の角度である.

3 次元足配置計画を 2 次元足配置計画と 全く 同様に離散的な基本オペレータを 3 次元空間に拡張するというアプローチでは以下のような問題がある.

1. 状態数の次元の増加による探索時間の増大
2. 環境モデルに対する足位置の精度

特に後者については足配置計画固有の問題となる. 環境モデルが与えられた時に, 足位置は環境モデルに対して接地していることが求められる. しかしながら, 後述の **eq.** 4.24 に代表される状態空間の量子化誤差により, 足位置は環境モデルに対して精度よく接地させることが難しくなる.

これら 2 つの問題を解決するため, xy 平面上における任意の位置姿勢は射影操作 $\mathrm{proj}(N_{2d}, E)$ によって環境モデル E に対して一意に 3 次元空間に射影可能であるという仮定を導入する.

$$N = \mathrm{proj}(N_{2d}, E) \tag{4.21}$$

ただし,

$$N_{2d} = (x, y, \theta) \tag{4.22}$$
$$= N_{z=0,\phi=0,\psi=0} \tag{4.23}$$

とする. 環境モデル E に対し, 足配置が 2 次元空間から 3 次元空間に射影できるため, 探索パラメータを 6 から 3 に削減させることができる. これによって状態次元数の増加による探索時間の増大を回避できる. また一方で射影操作 $\mathrm{proj}(N_{2d}, E)$ は量子化誤差を含まないため, 環境モデルに対する足位置の精度を確保できる. この仮定は, 多くの環境が床面や段差, 斜面などで構成できるという観点から合理的である. この仮定が成立しないのは崖や垂直はしごのような z 軸方向に特徴の多い環境である. **Fig.** 4.7 に射影操作 $\mathrm{proj}(N_{2d}, E)$ を図示する. 足位置は平面上にしか接地できないとすると, 射影操作 $\mathrm{proj}(N_{2d}, E)$ は環境から推定される平面 P に対し, 法線を揃えるように z 方向に移動させるものである.

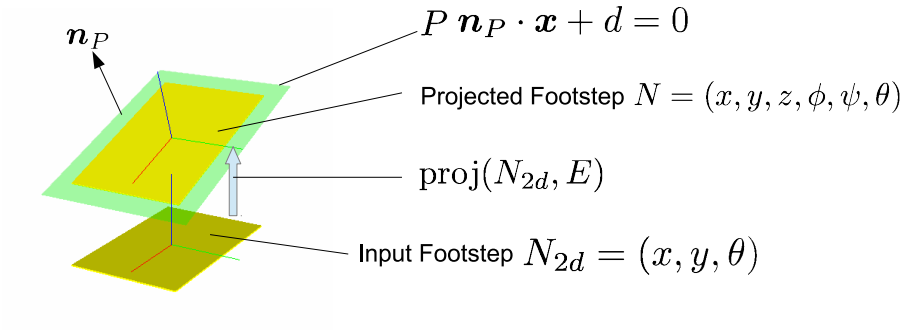


Fig4.7: Projection of Footstep onto Plane

Algorithm 3 3-D Footstep Planning with A*

```
O ← {S}
C ← ∅
while O do
    N ← pop(O)
    if N is G then
        return path(N)
    end if
    succ ← ∅
    for n in expand(N) do
        succ ← project(n)
    end for
    C ← N + C
end while
```

4.3.4　グリッドキャッシュによるクローズドリスト探索の高速化

離散グラフによる歩行配置計画アルゴリズム (**Alg.** 2) の計算量は, ナイーブな実装だとノード展開数を n とすると, $O(n^2)$ となる. これはノード N がすでにクローズドリストに追加されているかどうかを検査する計算はノード数と同じだけ行われ, 一回の検査に線形な時間 $O(n)$ がかかってしまうからである. 線形探索では位置, 姿勢がそれぞれ一定のしきい値以下のノードがクローズドリストに追加されているかを評価する (**eq.** 4.24).

$$|\boldsymbol{p}_n - \boldsymbol{q}_n| < thr_p \wedge |\boldsymbol{q}_n - \boldsymbol{q}_n| < thr_q \tag{4.24}$$

これらしきい値パラメータ thr_p, thr_q は, 探索の粒度を決定するパラメータという特徴を持つ. クローズドリスト探索の計算量の削減は, 足配置計画の速度を決める上で重要である. そこで

計算量 $O(n)$ である探索を $O(1)$ とするため, 離散化されたグリッド状のキャッシュ構造を導入する. これは多次元配列として実装され, 挿入, 探索はどちらも配列の添字アクセスによって実現されるため, 計算量は $O(1)$ となる.

射影操作 $\mathrm{proj}(N_{2d}, E)$ の導入により , 2 次元空間, 3 次元空間に依存せず探索対象となる状態量は x, y, θ の 3 つであるので, グリッド状のキャッシュは 3 次元配列で十分である. 各軸の添字は

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix} = \begin{pmatrix} \left[\frac{x}{\Delta x}\right] \\ \left[\frac{y}{\Delta y}\right] \\ \left[\frac{\theta}{\Delta \theta}\right] \end{pmatrix} \tag{4.25}$$

として計算される. ただし , $\Delta x, \Delta y, \Delta \theta$ はそれぞれ x, y, θ に関する量子化の刻み幅とする. 探索で利用する空間全体の 3 次元配列を事前に確保しておくことは難しいため, **Fig.** 4.8 のように, 展開されたノードを含む局所的なグリッドを逐次確保していく. **Fig.** 4.8 は x, y のみで探索した場合の挙動を図示したものである. 量子化刻み幅 $\Delta x, \Delta y, \Delta \theta$ は探索の速度と精度に大

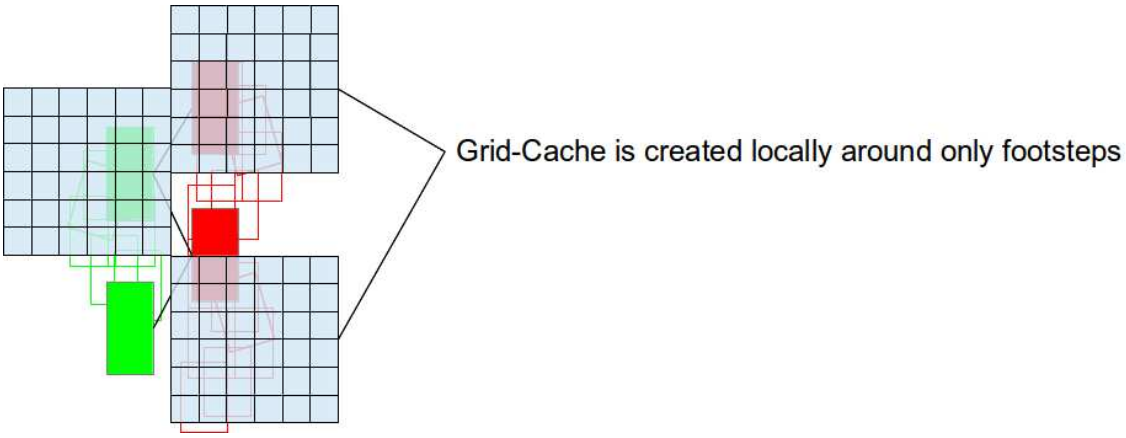


Fig4.8: Grid-Cache for Closed List of Footsteps

きな影響を与えるパラメータである. これらのパラメータを大きくすると, 探索に必要とされる時間は減少する一方, 複雑な環境においてはプランを生成することが難しくなる. 逆に刻み幅を小さくすると探索に必要となる時間は増加するが, 複雑な環境においてもプランを生成することが可能となる.

4.3.5　ローカル足位置修正による段差環境の探索高速化

離散グラフ探索による足配置計画では, 足配置の基本オペレータは離散化されていることはすでに述べた. このため, 段差のような環境付近では, 足配置を環境モデルに支持されるよう

に配置することが難しい. というのも, ヒューマノイドロボットは同サイズの人に比べ歩幅が短く, 足裏の大きさと同程度であることが多いためである. 例えば, 足裏サイズが 250mm であり, 歩行の足幅が 300mm の場合, 段差のような不連続な平面を突破するには 50mm の範囲内に足を配置する必要がある. 探索時にこのような狭い領域しか許されていない状況では多くの候補ノードが環境に対して支持されていないとして棄却されてしまう. これを解決するため, 段差付近では Chestunut らが提案する足配置の局所的な探索 [92] を導入する. **Alg.** 4 にそのアルゴリズムを示す.

Algorithm 4 3-D Footstep Planning with Local Refinement

```
1: O ← {S}
2: C ← ∅
3: while O do
4:    N ← pop(O)
5:    if N is G then
6:       return path(N)
7:    end if
8:    succ ← ∅
9:    for n in expand(N) do
10:      n′ ← project(n)
11:      if n′ is supported by E then
12:         push(n, O)
13:      else if n′ is partially supported by E then
14:         push(locallyMove(n′), O)
15:      end if
16:   end for
17:   C ← N + C
18: end while
```

4.4 動作計画器と認識器の協調的スケジューリングによる近似足配置計画法

3 次元足配置計画では, 前節で述べたように足位置を 2 次元状態空間から 3 次元状態空間に射影するための平面を認識する必要がある. 従来の手法では事前に平面領域を認識し, その後に足配置計画を行うアプローチであった. これは足配置計画に限らず, 動作計画手法は事前に認識された環境モデルを必要とする. 本節では事前に認識処理可能な限り行わずに, 足配置計

画の中で認識を同時に協調的に進めていく計画法を提案する. さらに認識処理を軽減させるため, 環境平面仮説と呼ぶ仮定を導入した近似を用いることで高速な足配置計画が可能である. 環境平面仮説とは, 環境の大部分は連続する平面によって構成されるという仮説である.

4.4.1　足配置周辺のローカル点群を利用した部分的平面認識

3 次元点群モデルに対するローカル候補点群の抽出

環境モデルとして 3 次元点群を利用することで, 平面 P は RANSAC による平面推定を用いて計算することができる. 3 次元点群から平面を推定するために RANSAC によるモデル推定は広く利用されている手法である [93]. 具体的な足配置の射影手順は以下のようになる.

1. 足配置周辺の点群を抽出し, 候補点群とする.
2. 候補点群から平面を推定する.
3. 推定された平面に対し, 足配置を鉛直に射影する. $(\mathrm{proj}(N_{2d}, P))$
4. 射影された足配置が点群によって支持されているか確認する.

RANSAC は方程式で表現されるモデルに関して範囲を設定することが難しい. 足配置周辺の平面領域を推定するために, 前処理として足配置周辺の候補点群を抽出する. 候補点群を高速に抽出するため, 予め 3 次元点群に対し 2 次元インデックス $h(i,j) \in H(E)$ を作成しておく (**Fig.** 4.9). この 2 次元インデックスを利用することでまず粗く足配置近傍の点を抽出する. i, j はインデックスの画素座標, $h(i,j)$ は画素 i, j に含まれる 3 次元点の集合である. 2 次元上のインデックスを用意しておくことで, 足配置 N_{2d} の近傍にある点群を 2 次元画像の塗りつぶしアルゴリズムによって高速に抽出可能である. 粗く近傍の点を抽出した後に, 精度高く近傍点を抽出する.

3 次元点群モデルに対するローカル平面認識

抽出された候補点に対して平面を推定するためには, RANSAC を利用する [93]. RANSAC により推定するモデルは平面の方程式である.

$$ax + by + cz + d = 0$$

候補点を予め抽出していることにより, ローカル平面認識は環境全体から推定するよりも高速に動作する. というのも, RANSAC の計算量は, inlier 点数 n に対して $O(nm)$ となる. ただしここで m は繰り返し計算回数である.

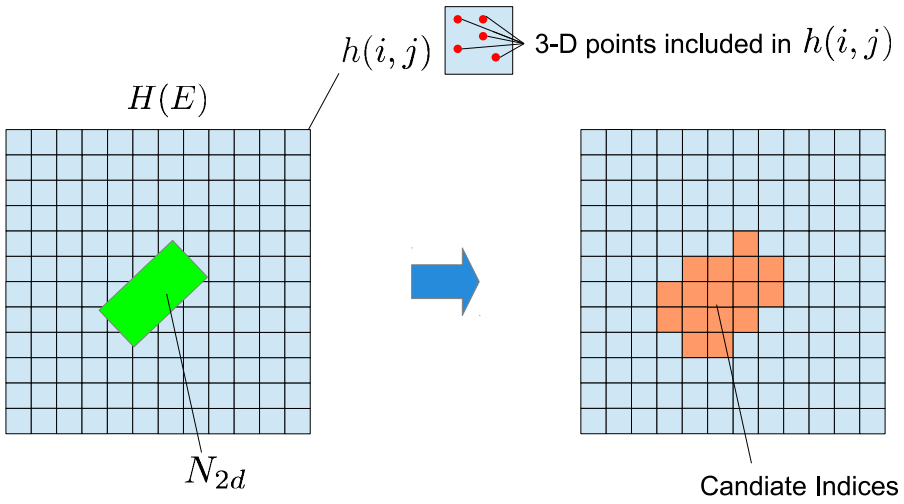


Fig4.9: Fast Approximated Neighbor Search using 2-D Grid Indices

3 次元点群に対する足配置の支持確認

RANSAC による平面認識では, 範囲を考慮することが難しい. そのため, 足配置を射影した後にそれが点群上に乗っているかを確認する. 足配置 N 上の点を n 個適当に選び, それを $\boldsymbol{q}_i$ とする. この時, すべての $\boldsymbol{q}_i$ に関して, 一定の距離 δr 以内にある点 p が存在するとき, 足配置は点群モデルによって支持されていると判断できる (**Fig.** 4.10). ただし 点 p は 3 次元点群によって表現された環境モデル E の一点である.

$$|\boldsymbol{q}_i - \boldsymbol{p}| < \delta r \tag{4.26}$$

$$^{\exists}\boldsymbol{p} \in E \tag{4.27}$$

$$^{\forall}\boldsymbol{q}_i \in (\boldsymbol{q}_0, \boldsymbol{q}_1, \cdots, \boldsymbol{q}_n) \tag{4.28}$$

半径 δr 以内の点を探索するためには, kd-tree を用いる. サンプリング点 $\boldsymbol{q}_i$ は多ければ多いほど, 支持確認の確からしさが向上するが, それは計算コストとのトレードオフとなる. そのため, サンプリング点 $\boldsymbol{q}_i$ に関しては長方形形状の足配置 N の各頂点と重心座標を用いる.

4.4.2　環境平面仮説による優先度計算の近似に従った認識遅延

Alg. 3 に従うと, 足配置の射影後に最も始めに足配置座標が利用されるのは, ノードをオープンリストへの追加時に必要となる優先度の計算である. 優先度を正確な値とするには, オープンリストへの追加前に足配置の射影を行う必要がある. しかし, 環境平面仮説に従うと, 射影操作によって足配置が移動する場合はまれである. そのため, オープンリストで用いる優先

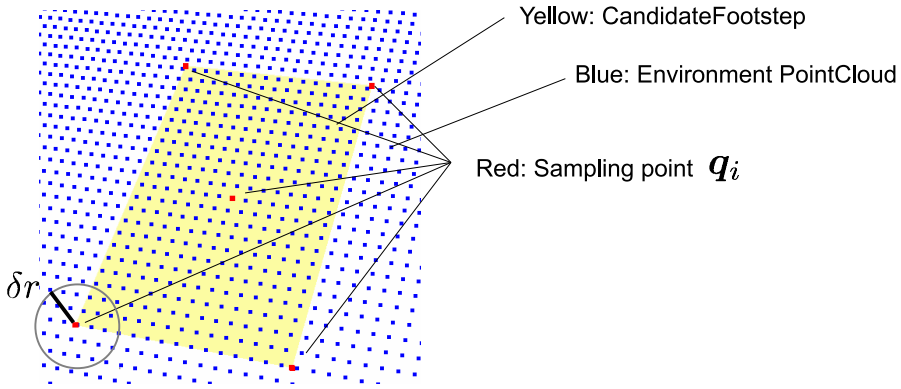


Fig4.10: Confirmation of Footstep on PointCloud Model

度を足配置の射影操作前の座標を利用することにする．この近似の導入により，足配置の射影はオープンリストからノードが選択された直後まで評価を遅延させることが可能となる．**Alg. 5** に射影操作を遅延させたアルゴリズムを擬似コードで示す．この認識遅延により，計画で必

Algorithm 5 3-D Footstep Planning with Lazy Local Plane Detection

$O \leftarrow \{S\}$
$C \leftarrow \emptyset$
while O **do**
　$N \leftarrow pop(O)$
　$coef \leftarrow estimatePlane(N, P)$
　$N' \leftarrow project(N, coef)$
　if N' is supported by P **then**
　　if N' is G **then**
　　　return $path(N)$
　　end if
　end if
　$push(expand(N), O)$
　$C \leftarrow N + C$
end while

要な部分のみ，認識が行われるようになる．

4.4.3　環境平面仮説による平面認識処理の削減

本小節では，環境の複雑さに応じて認識処理を省略する機能を足配置計画器に導入する．足配置の認識処理は以下のような手順であった．

1. 足配置周辺の点群を抽出し, 候補点群とする.
2. 候補点群から平面を推定する.
3. 推定された平面に対し, 足配置を鉛直に射影する.
4. 射影された足配置が点群によって支持されているか確認する.

この中で, 4 つめの点群による支持確認は, 射影した足配置が実際に平面上に乗っているかを確認するために必要となる. 例えば, この支持確認によって足配置の前方部分しか平面上に乗っていない場合などを足配置候補から棄却することができる.

一方, 環境平面仮説に従うと, 一般に環境の多くは平面で構成される. 3 次元足配置計画では, 射影操作がない場合はノードを展開する元の足配置に対して同一平面上でノードが展開されていく. したがってその多くのノードでは射影操作による足配置の移動は少ないと推定され, この時の認識処理は必要ないと言える. 環境平面仮説導入すると, 足配置の認識処理は以下のように支持確認を先頭に持っていくことで平面認識処理を省略することができる.

1. 足配置が点群によって支持されているか確認する.
2. 支持されていなかった場合, 認識処理へと移る
3. 足配置周辺の点群を抽出し, 候補点群とする.
4. 候補点群から平面を推定する.
5. 推定された平面に対し, 足配置を鉛直に射影する.
6. 射影された足配置が点群によって支持されているか確認する.

というのも, 平面認識処理を行わずとも, 2 次元上に展開した足配置ノードが環境に対して支持されている可能性が高いからである. このような認識処理の削減は, 多くが平面で構成されるような環境に対しては劇的に計算コストを下げることができるが, 複雑な環境ではかえって計算コストが増大してしまう可能性がある. というのも, 一つの足配置を考慮する中で点群による支持確認処理を最悪 2 回行う必要があるためである.

Alg. 6 に楽観的足配置平面認識を導入した 3 次元足配置計画のアルゴリズムを擬似コードで示す.

Algorithm 6 3-D Footstep Planning with Lazy Optimistic Local Plane Detection and Local Movement

$O \leftarrow \{S\}$
$C \leftarrow \emptyset$
while O **do**
  $N \leftarrow pop(O)$
  if N is supported by P **then**
    $n' \leftarrow N$
  else
    $coef \leftarrow estimatePlane(N, P)$
    $N' \leftarrow project(N, coef)$
    if N' is supported by P **then**
      $n' \leftarrow N'$
    else if N' is partially supported by P **then**
      $push(locallyMove(N'), O)$
    end if
  end if
  if N' is G **then**
    return $path(N)$
  end if
  $push(expand(N), O)$
  $C \leftarrow N + C$
end while

4.5　SLAM によるロボット周囲の詳細点群モデリング

本節では提案した足配置計画法のための密な 3 次元点群モデルを生成するための手法を述べる. **Fig.** 4.11 に SLAM によるロボット周囲の詳細点群モデリングの全体構成を示す.

本章で議論してきた足配置計画では密な点群モデルが必要とされる. というのも, 足配置が点群上に接地可能であるかを検査するためには, 点群の密度がその精度に大きく影響を与えるからである. 密度の低い点群を環境モデルとして利用すると, 足配置が点群上に接地不可能であると判断される場合, それが点群の密度の低さに由来するのかそれとも実際の環境中に障害物や欠損が存在するのかを切り分けるのが困難である.

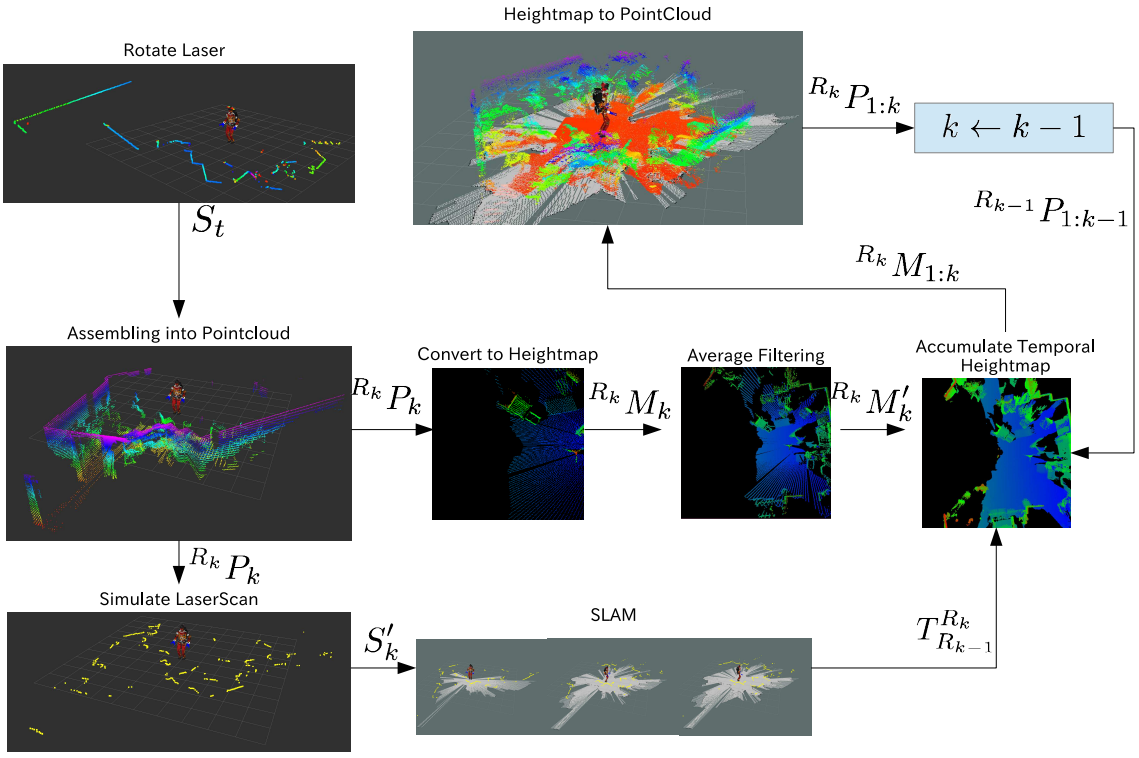


Fig4.11: Overview of Environment Modeling based on SLAM and Heightmap Representation

4.5.1 Heightmap による点群データのモデル化

すでに足配置計画において, 本論文では射影操作 $\mathrm{proj}(N_{2d}, E)$ を導入している. これは特に鉛直方向に関して複数の射影が存在しうる可能性を無視することであった. この仮説に従い, 密な 3 次元点群モデルを生成するために, 点群データを Heightmap と呼ばれる 2.5 次元データ構造にして環境モデルを生成する. **Fig.** 4.12 に 3 次元点群を Heightmap とし, 高さに応じて色を変化させたものを示す. Heightmap は画像のような 2 次元配列の形をしており, 各画素データに点群の高さが保持されているようなデータ構造である. Heightmap を利用することによる利点は以下のようなものである.

1. 3 次元空間を直接グリッドにするよりも 2 次元データ構造なのでメモリ効率が良い.
2. 3 次元点群を 2 次元画像処理によって処理できる.

前者に関しては, メモリ効率の向上により, 粒度の小さい点群データを利用できることにつながる. 一方後者に関しては, 後に述べる 2 次元画像処理を利用した補間によって欠損部を補充

できることにつながる.

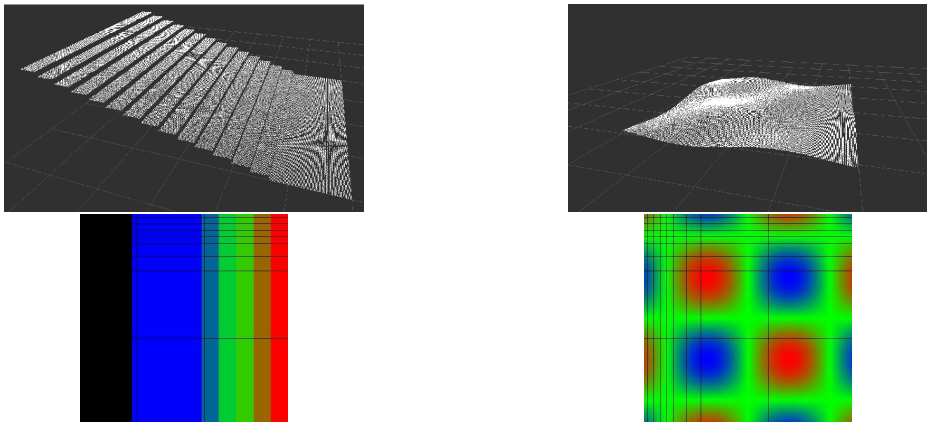


Fig4.12: Heightmap Model: Left) Simulated 3-D PointCloud Model.Right) Heightmap Generated from Left PointCloud Model. Upper) Stair Environment. Lower) Hills Environment.

一般の 3 次元点群は同じ x, y 座標に対し異なる複数の z を持つ 3 次元点が含まれうる. Heightmap 導入による欠点は, Heightmap を生成する過程でこれらの複数の 3 次元点は無視されることである. しかしながら足配置計画に応用することを考慮すると, これらの複数 3 次元点の中で最も大きな z 座標を持つ 3 次元点のみを Heightmap に登録することでこの欠点を少なくすることができる. というのも, 低い 3 次元点に対し足配置を考慮する可能性は衝突の可能性から無視できるからである. さらにロボットよりも高い位置の点群はあらかじめ考慮しないことによって, 考慮しなくて良い天井などの点群を除去する. **Fig.** 4.13 に示すように, 本論文ではロボットの肩周辺よりも高い位置にある点群は考慮しないとした.

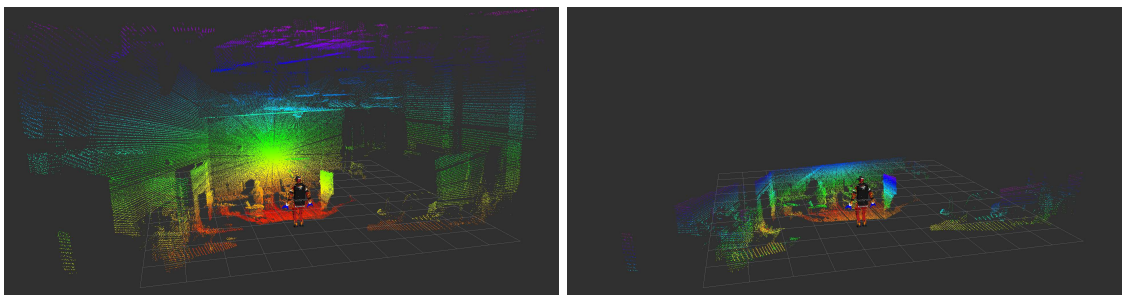
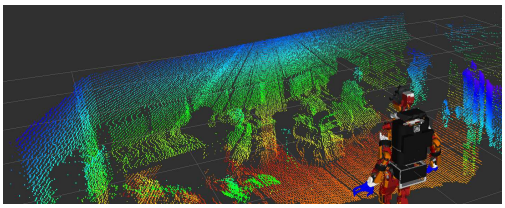


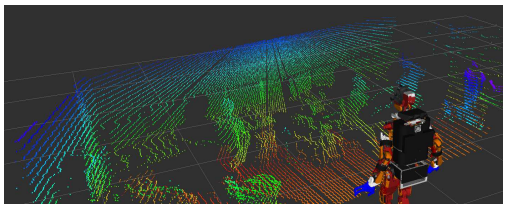
Fig4.13: Input PointCloud to Construct Heightmap. Left) Original PointCloud. Right) PointCloud Cut Upper Region of Robot.

4.5.2 Heightmap による点群データの空間方向での補間

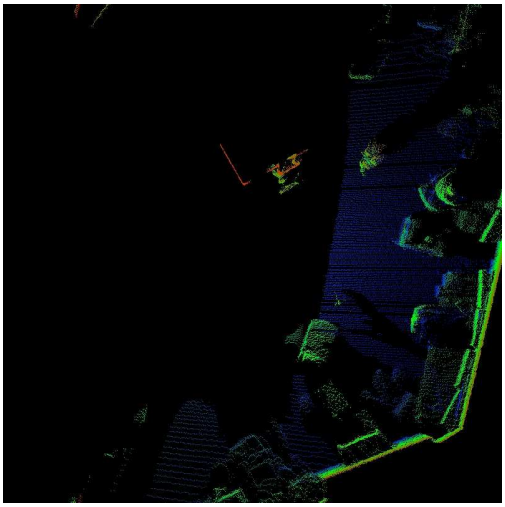
3 次元点群の取得のためのセンサはステレオカメラ, ToF カメラ, レーザなど複数の手法が考えられるが, 本論文では 2 次元レーザを回転機構によって回転させ, 3 次元点群を得るものとする. 2 次元レーザを回転させて 3 次元点群を取得する場合, 距離に応じてサンプリング間隔が粗くなる. したがって密な点群を得るためには点群の補間が必要となる. **Fig.** 4.14 に実際に 2 次元レーザを回転数を変化させて回転させて得た 3 次元点群および点群を Heightmap に変換したものを示す. Heightmap に変換することで, 3 次元点群がどの程度サンプリング間隔によって欠損しているかが可視化されている.



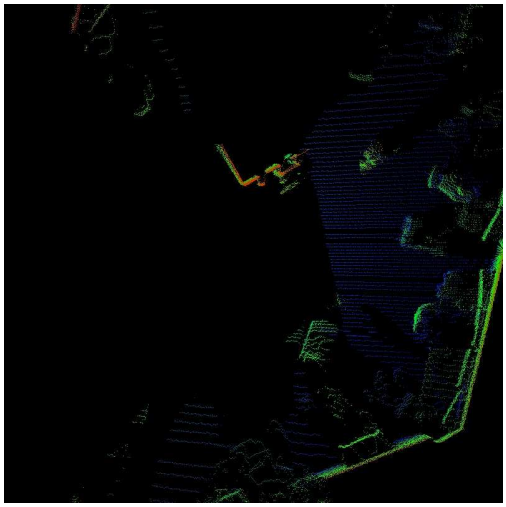
(a) Input PointCloud with $\dot{\theta} = 0.5$ rad/sec



(b) Input PointCloud with $\dot{\theta} = 1.0$ rad/sec



(c) Input PointCloud with $\dot{\theta} = 0.5$ rad/sec



(d) Input PointCloud with $\dot{\theta} = 1.0$ rad/sec

Fig4.14: PointCloud and HeightMap from Rotating Laser with Different Rotation Velocity

Heightmap 上に存在する欠損をごま塩ノイズとしてみなし, これを平均フィルタによって補間する. 画素 i, j に対する画像カーネル (**Fig.** 4.15) を $k(m, n) \in K(i, j)$ とすると, Heightmap $M(i, j)$ を平均フィルタによって補間した Heightmap $M'(i, j)$ は **eq.** 4.29 のよ

うになる.

$$M'(i,j) = \frac{\sum_{M(m,n)\neq\emptyset} M(m.n)}{N(M(k,l)\neq\emptyset)} \tag{4.29}$$

ここで, $N(M(k,l)\neq\emptyset)$ は画像カーネル $k(m,n)$ なかで, 欠損していない画素数である. 平均

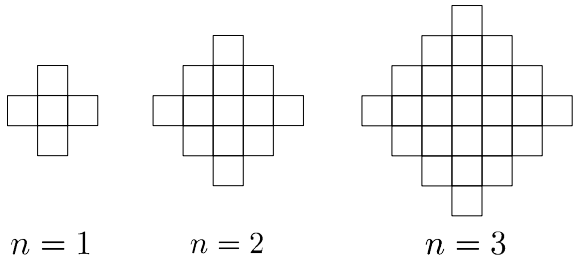


Fig4.15: Kernel Patches with Different Sizes

フィルタはカーネルの中で欠損となっていない画素の平均値として機能する. これは, 環境は連続な平面で構成されているという仮定に従った補間方法であり, 元の環境が単一の平面で構成されている場合は完全な点群が復元可能である. しかしながら, 高さ方向にエッジを持つような領域で欠損が生じた場合, 補間によってノイズが発生する. **Fig.** 4.16 に階段モデルに欠損を生じさせ, 平均フィルタによって Heightmap 補間を行い, 補間された Heightmap から点群を再構成したものを示す. このノイズは, すでに述べた連続な平面で構成されているという仮定に従わない領域で発生する. また, このようなノイズはカーネルサイズ n が大きくなるほど乗りやすい傾向にある. そのため, n は大きすぎる値を設定することはできない. **Fig.** 4.17,

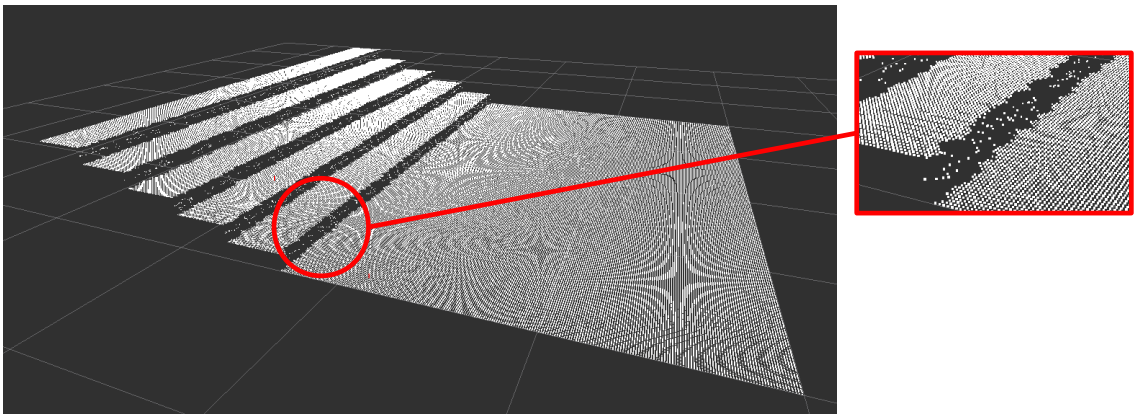


Fig4.16: Reconstructed Stair PointCloud from Completed Heightmap. Zoomed Region Shows Shadow Noise because of Average Filtering.

Fig. 4.18 にランダムに欠損を生じさせたシミュレーションによる点群補間実験を示す. 黒で

塗られた画素が欠損させた画素である. ここで r は欠損ノイズ率である.

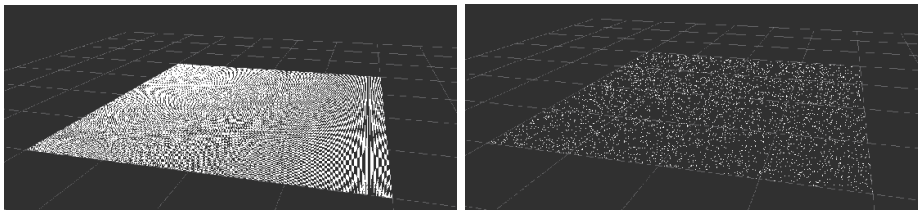
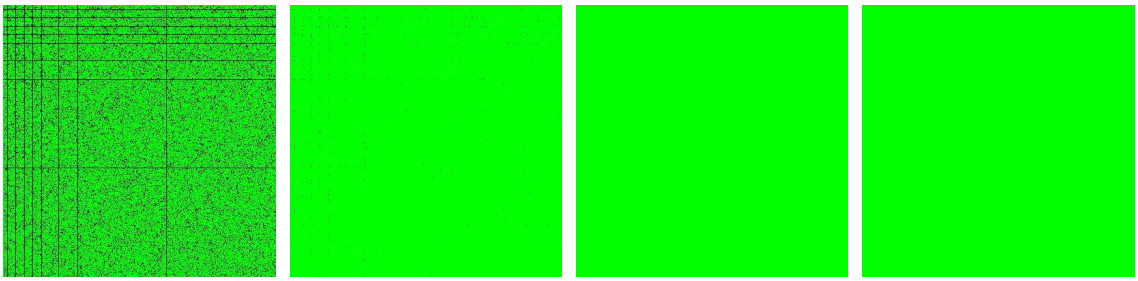
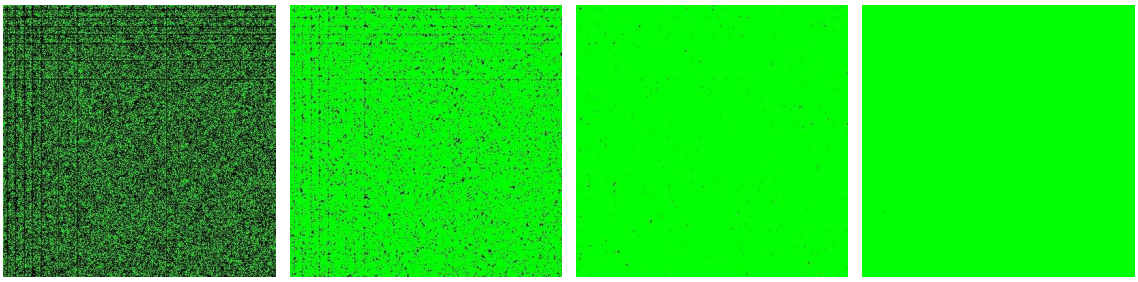


Fig4.17: Left) Flat PointCloud Environment. Right) Environment with $r = 0.8$ Hole Noise Simulation.



(a) Input PointCloud with $r = 0.2$ (b) $r = 0.2, n = 1$ (c) $r = 0.2, n = 2$ (d) $r = 0.2, n = 3$



(e) Input PointCloud with $r = 0.6$ (f) $r = 0.6, n = 1$ (g) $r = 0.6, n = 2$ (h) $r = 0.6, n = 3$

Fig4.18: Heightmap Completion of Flat PointCloud (**Fig.** 4.17) by Average Filtering with Different Noise Ratios r and Kernel Sizes n

Fig. 4.21 に実際に 2 次元レーザをいくつかの回転速度で回転させた場合の補間について実験した結果を示す. レーザの回転速度 $\dot{\theta}$ が小さいほど, 3 次元環境を細かな粒度でサンプリングできるため, 密な点群が入力としてとれる (**Fig.** 4.22). そのため, Heightmap の空間的な補間においても, $\dot{\theta}$ が小さいほど平均フィルタが機能していることがわかる (**Fig.** 4.21). しかしながら, $\dot{\theta}$ が小さいほど 3 次元環境をスキャンするのに時間がかかるため, 環境の変化を検知するために時間がかかってしまう. このトレードオフのため, 本論文では実験的に $\dot{\theta} = 0.5$

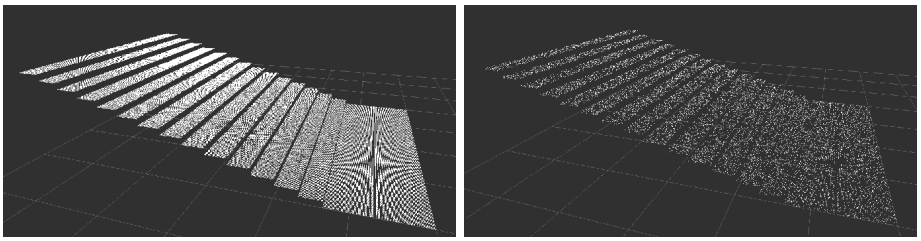
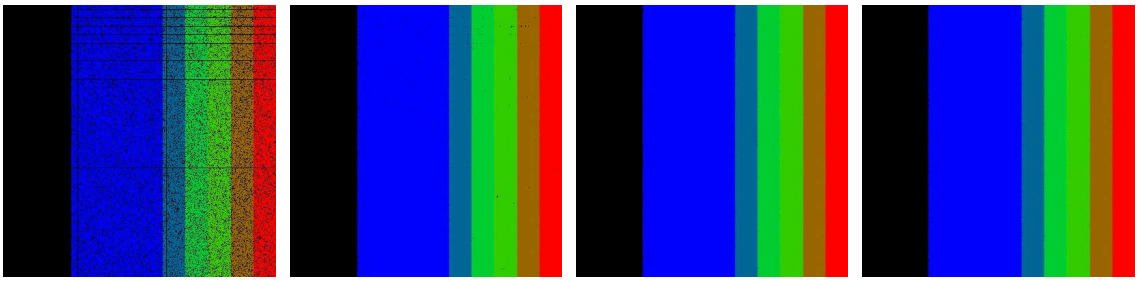
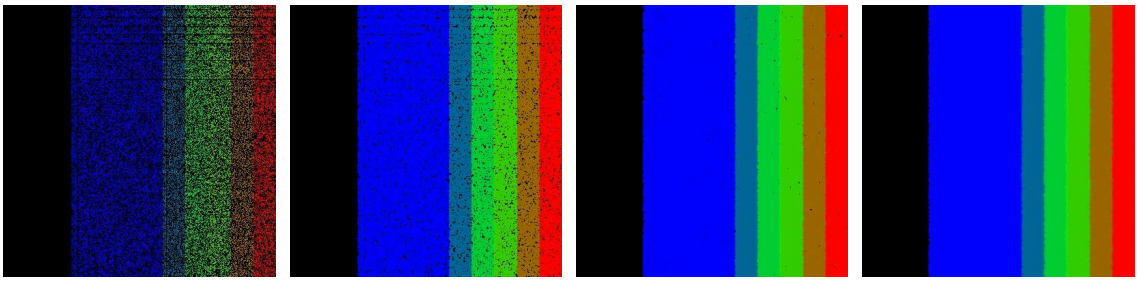


Fig4.19: Left) Stairs PointCloud Environment. Right) Environment with $r = 0.8$ Hole Noise Simulation.



(a) Input PointCloud with $r = 0.2$　(b) $r = 0.2, n = 1$　(c) $r = 0.2, n = 2$　(d) $r = 0.2, n = 3$



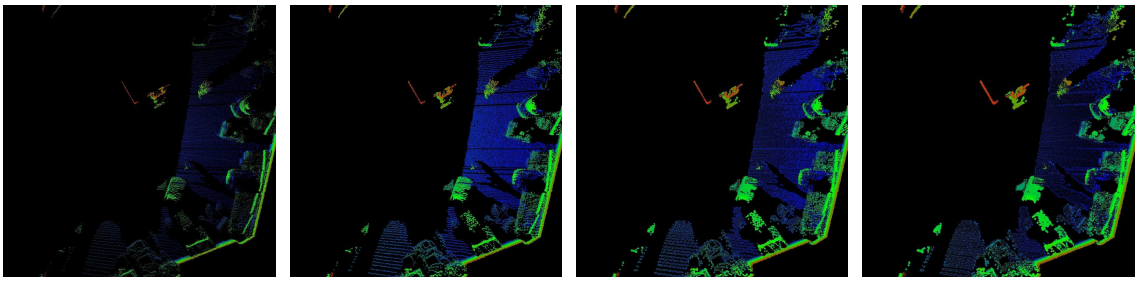
(e) Input PointCloud with $r = 0.6$　(f) $r = 0.6, n = 1$　(g) $r = 0.6, n = 2$　(h) $r = 0.6, n = 3$

Fig4.20: Heightmap Completion of Stairs PointCloud by Average Filtering with Different Noise Ratios r and Kernel Sizess n

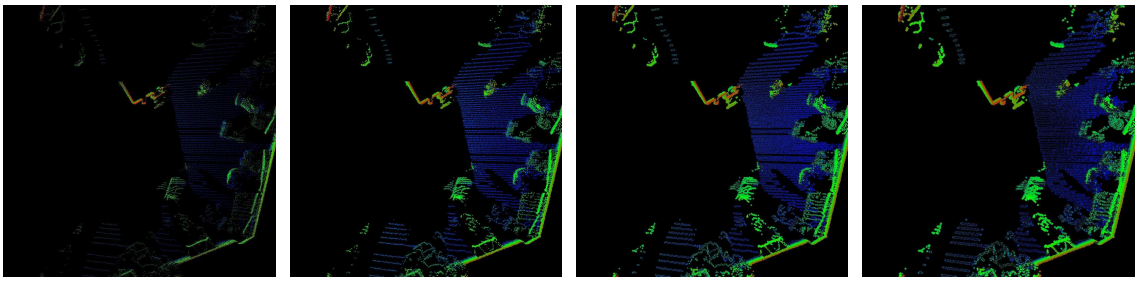
rad/sec, $n = 2$ というパラメータを選択した.

4.5.3　Heightmap による時系列方向での点群データ統合

第 4.5.2 節において, Heightmap 表現を利用した 3 次元点群の空間方向の補間方法を示した. 空間方向の補間によってある程度は密な点群を生成可能であるが, 以下のような観点からこれだけでは不十分である.

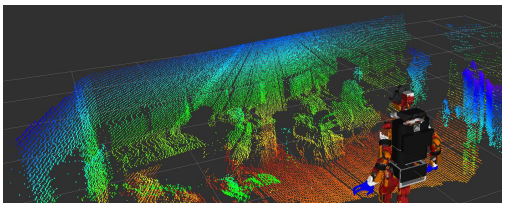


(a) Input PointCloud with $\dot{\theta} = 0.5$ rad/sec (b) $\dot{\theta} = 0.5$ rad/sec, $n = 1$ (c) $\dot{\theta} = 0.5$ rad/sec, $n = 2$ (d) $\dot{\theta} = 0.5$ rad/sec, $n = 3$

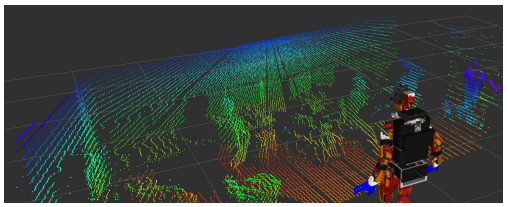


(e) Input PointCloud with $\dot{\theta} = 1.0$ rad/sec (f) $\dot{\theta} = 1.0$ rad/sec, $n = 1$ (g) $\dot{\theta} = 1.0$ rad/sec, $n = 2$ (h) $\dot{\theta} = 1.0$ rad/sec, $n = 3$

Fig4.21: Heightmap Completion with Actual Sensor with $\dot{\theta} = [0.5, 1.0]$ rad/sec



(a) Input PointCloud with $\dot{\theta} = 0.5$ rad/sec



(b) Input PointCloud with $\dot{\theta} = 1.0$ rad/sec

Fig4.22: Input PointCloud of **Fig.** 4.21, Visualized as 3-D PointCloud

1. オクルージョンによる 3 次元点群の欠損
2. サンプリング間隔が大きく 増大してしまう 領域の補間

そこで, 本小節では Heightmap を 時系列方向に統合することでこれら 2 つの問題を 解決する. この時, 以下の優先度に従って点群モデルを 統合する.

1. 最新 (時刻 k) の点群

（ a）高い位置にある 点群

（ b）低い位置にある点群

2. 過去 (時刻 $1:k-1$) の点群

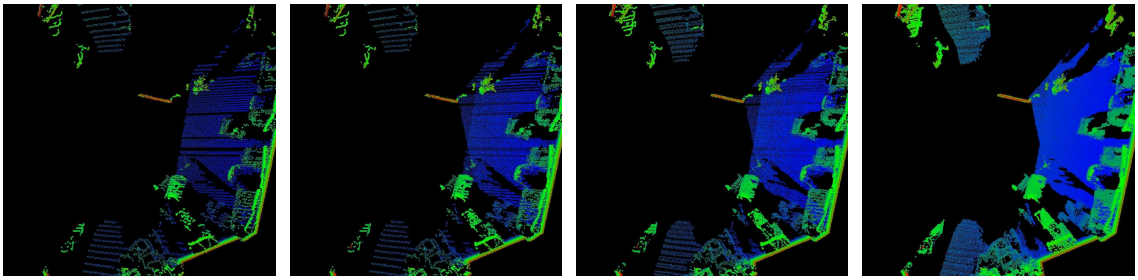
時刻 k の点群を高さの優先度に従って統合する手法についてはすでに第 4.5.2 節にて述べた.

時刻 k におけるロボットの位置を R_k, 座標系 R からの相対座標で表現される Heightmap と点群 ${}^{R}M_k, {}^{R}P_k$ と, 時刻 1 から $k-1$ までのデータが統合された Heightmap と点群 ${}^{R}M_{1:k-1}, {}^{R}P_{1:k-1}$ とすると, **Alg.** 7 のように ${}^{R_k}M_k$ の中で欠損となっている画素を過去の ${}^{R_k}M_{1:k-1}$ で補間することで時系列方向の統合が可能である. ここで, ${}^{R_k}_{R_{k-1}}T$ は R_{k-1} から R_k

Algorithm 7 Integrate Heightmap at step k with Heightmap from step 1 to $k-1$

1: ${}^{R_k}M_{1:k} \leftarrow {}^{R_k}M_k$
2: ${}^{R_k}P_{1:k-1} \leftarrow {}^{R_k}_{R_{k-1}}TP_{1:k-1}$
3: ${}^{R_k}M_{1:k-1} \leftarrow \text{HeightmapToPointCloud}({}^{R_k}P_{1:k-1})$
4: **for** $m(i,j) in {}^{R_k}M_{1:k}$ **do**
5:　**if** $m(i,j) = \emptyset$ **then**
6:　　$m(i,j) \leftarrow {}^{R_k}m_{1:k-1}(i,j)$
7:　**end if**
8: **end for**

への変換行列であり, これは自己位置推定の結果から得られる. **Fig.** 4.23 に時系列方向での点群データ統合を行った Heightmap の時間経過を示す. この実験では簡単のため, SLAM による自己位置推定を除外するためロボットは静止させている. 時間ステップ k が進むに連れて, 点群の欠損が補間されて密な Heightmap となっていることがわかる.



(a) $k=0$　(b) $k=1$　(c) $k=2$　(d) $k=10$

Fig4.23: Heightmap Temporal Completion with Static Robot Location

4.5.4 SLAM によるロボットの自己位置推定

第 4.5.3 節で述べた, ロボットの移動量 ${}^{R_k}_{R_{k-1}}T$ を得るために, SLAM(Simulteneous Localization And Mapping) と呼ばれる地図生成と自己位置推定を同時問題として解く手法を用いる. ロボットに搭載されたエンコーダや IMU などの内部センサのみでは, ロボットは目標値通りに足配置を実現することは難しい. 特にヒューマノイドロボットでは歩行時に片足にすべての体重がかかり , 減速機構のたわみが無視できない.

特に 2 次元レーザを利用した SLAM は台車型ロボットにおいて成功的な研究が多くなされている. 台車型ロボットではセンサ位置はロボット原点に対して一定位置にあり , 一方ヒューマノイドでは高さを含めたロボットの全身の姿勢が歩行中に動くという点が異なる. そこで, 床面から高さ一定の水平面を考え, そこに仮想的な 2 次元レーザを設置することで, すでに台車型ロボットで利用されている 2 次元 SLAM ソフトウェアを流用する. **Fig.** 4.24 に仮想レーザ平面を図示する. このように, ロボットの姿勢に依存せず高さ一定な平面を仮想レーザ平面とする.

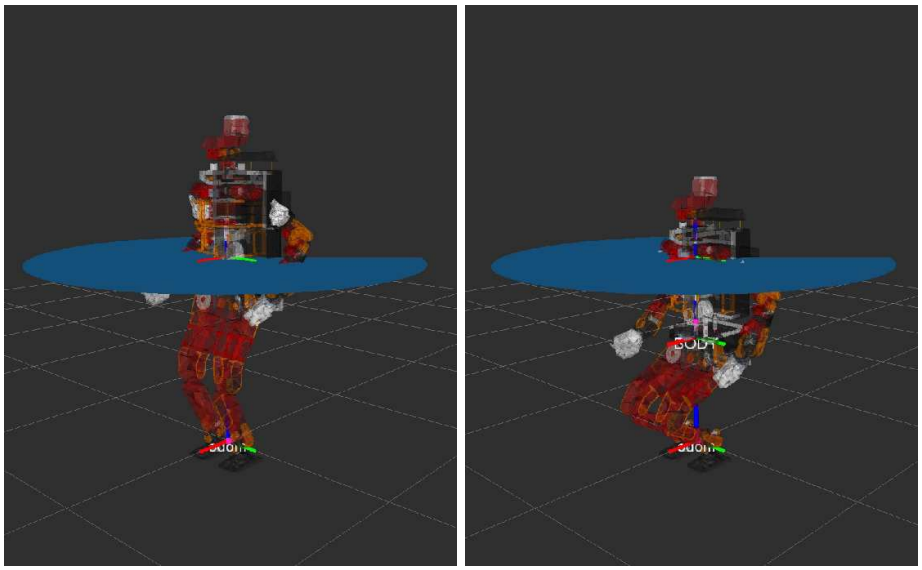


Fig4.24: Virtual Laser Scan Plane with Constant Height from Ground

台車型ロボットにおいて利用される SLAM は 2 次元 SLAM と呼ばれ, x, y および θ を推定する問題である. 一方でヒューマノイドの場合は 3 次元 SLAM となり , $x, y, z, \phi, \psi, \theta$ の 6 パラメータを推定する問題となる. ここで ϕ, ψ, θ はそれぞれ x, y, z 軸周りの回転を意味する. しかしながらこの 6 パラメータすべてを SLAM によって推定するのはヒューマノイドの足配

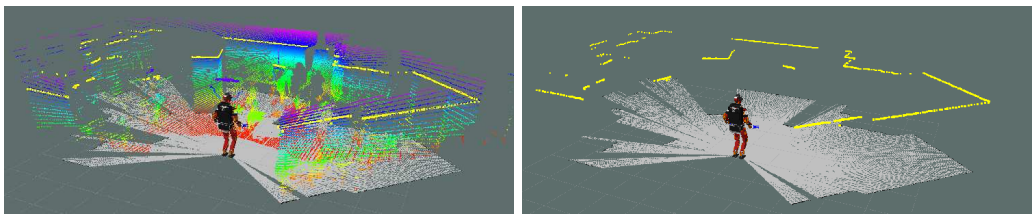


Fig4.25: Virtual 2-D Scan Geneted from 3-D PointClooud from Rotating Laser. Upper) Input 3-D PointCloud. Lower) Simulated 2-D Scan Geneted from Input 3-D PointClooud.

置計画に応用することを考慮するとなじまない. というのも, 特に z, ϕ, ψ は自己位置推定によって値が変動することで制御器に致命的な影響を与えうるからである. そこで z は制御目標値, ϕ, ψ に関しては制御機内で利用されている IMU からのカルマンフィルタによる状態推定結果をそのまま用いる. 後者に関しては重力方向を考慮することで, 値をドリフトすることなく推定することができる.

このような構成によって SLAM が動作できることをシミュレーションによって制度評価した. シミュレーションにおける環境は **Fig.** 4.26 のような直線的な環境である. この環境中でロボットを歩行させ, 歩行のオドメトリにノイズとして x, y 方向の正規分布のノイズを加えた. **Fig.** 4.28 にシミュレータから得られる真値, ノイズを加えたオドメトリ, そして自己位置推定によって修正した軌道をグラフにした.

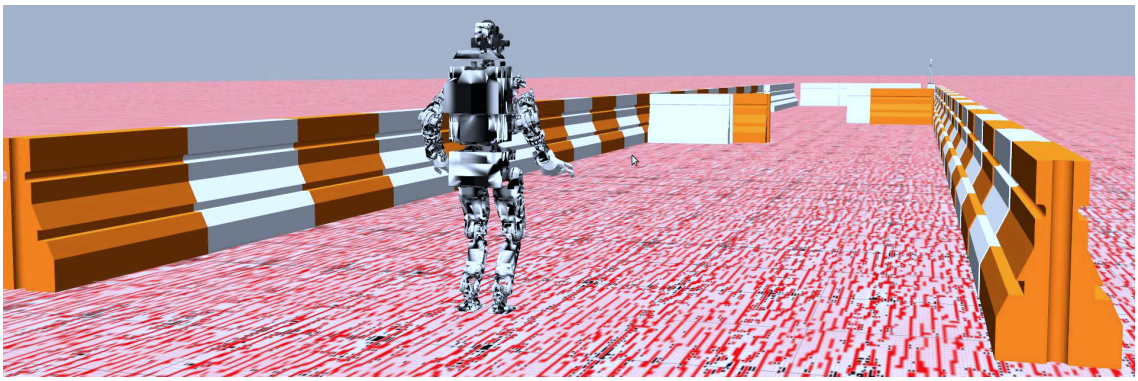
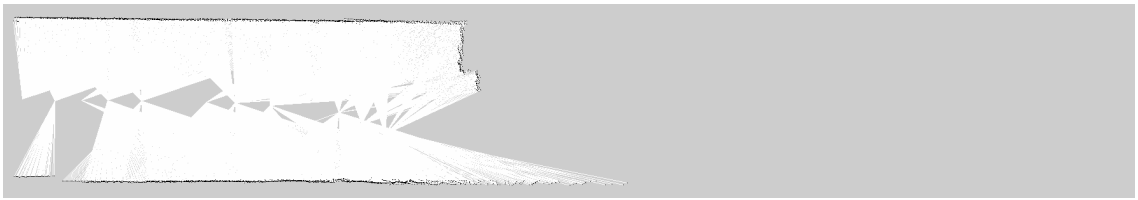


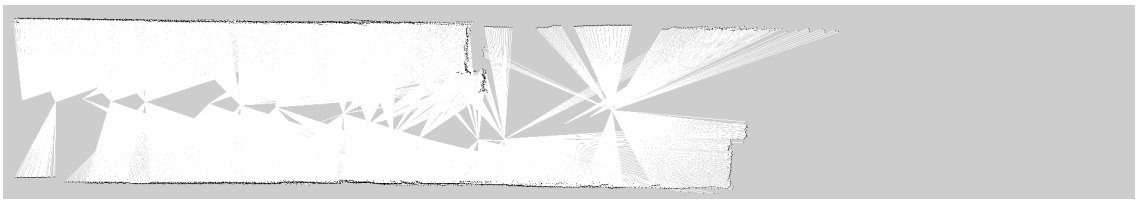
Fig4.26: Simulation Environment to Evaluate SLAM Integration

Fig. 4.29 に実環境で生成した 2 次元地図を示す. また, その上に Heightmap を重ねたものを **Fig.** 4.30 に示す. 自己位置推定結果を利用して密な環境の点群モデルが生成されている.

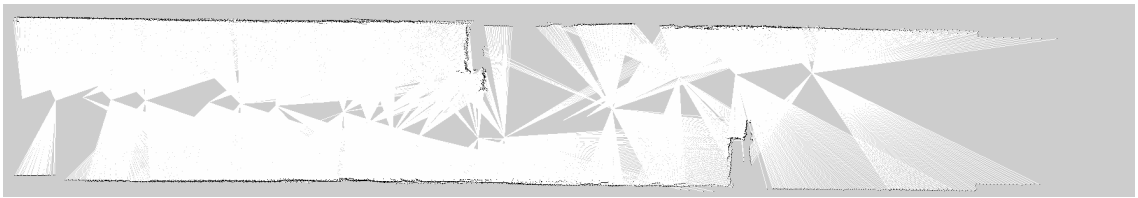
(a) $t = 100$ sec



(b) $t = 200$ sec



(c) $t = 300$ sec



(d) $t = 400$ sec

Fig4.27: Built Global Map with Different Frames in Simulated Environment (**Fig.** 4.26)

4.6 足配置計画法の評価実験

環境モデルとなる 3 次元点群をシミュレーションによって与えた場合の足配置計画に必要となる計算時間によって評価実験を行う. この実験では 3 種類の環境モデルを与えた. それらは平面環境, 階段環境, 斜面環境である. **Fig.** 4.31, **Fig.** 4.32, **Fig.** 4.33 にそれぞれ 3 次元点群として可視化したものと, Heatmap として可視化したものを示す. これらの環境中で足配置計画した結果生成されたプランを **Fig.** 4.34 に示す. これらのプランを生成するために必要となった時間を環境平面仮説に基づく近似を行った場合, 行わなかった場合について **Table** 4.1 に示す.

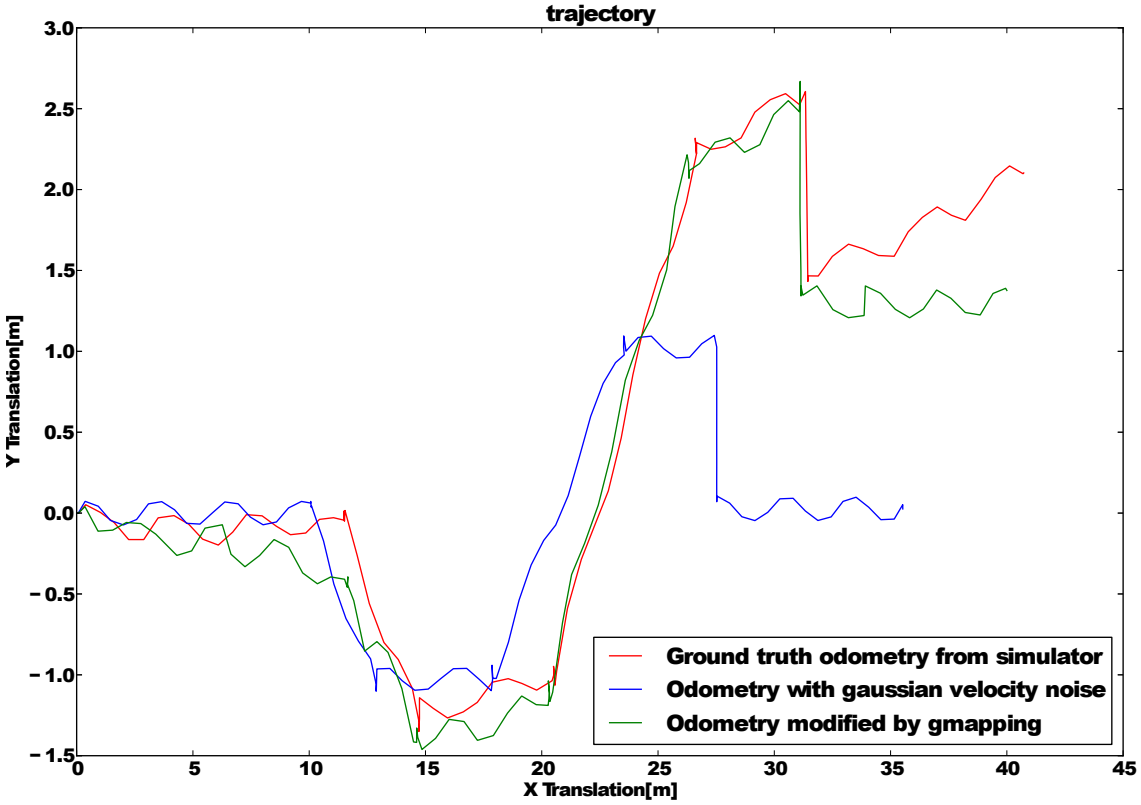
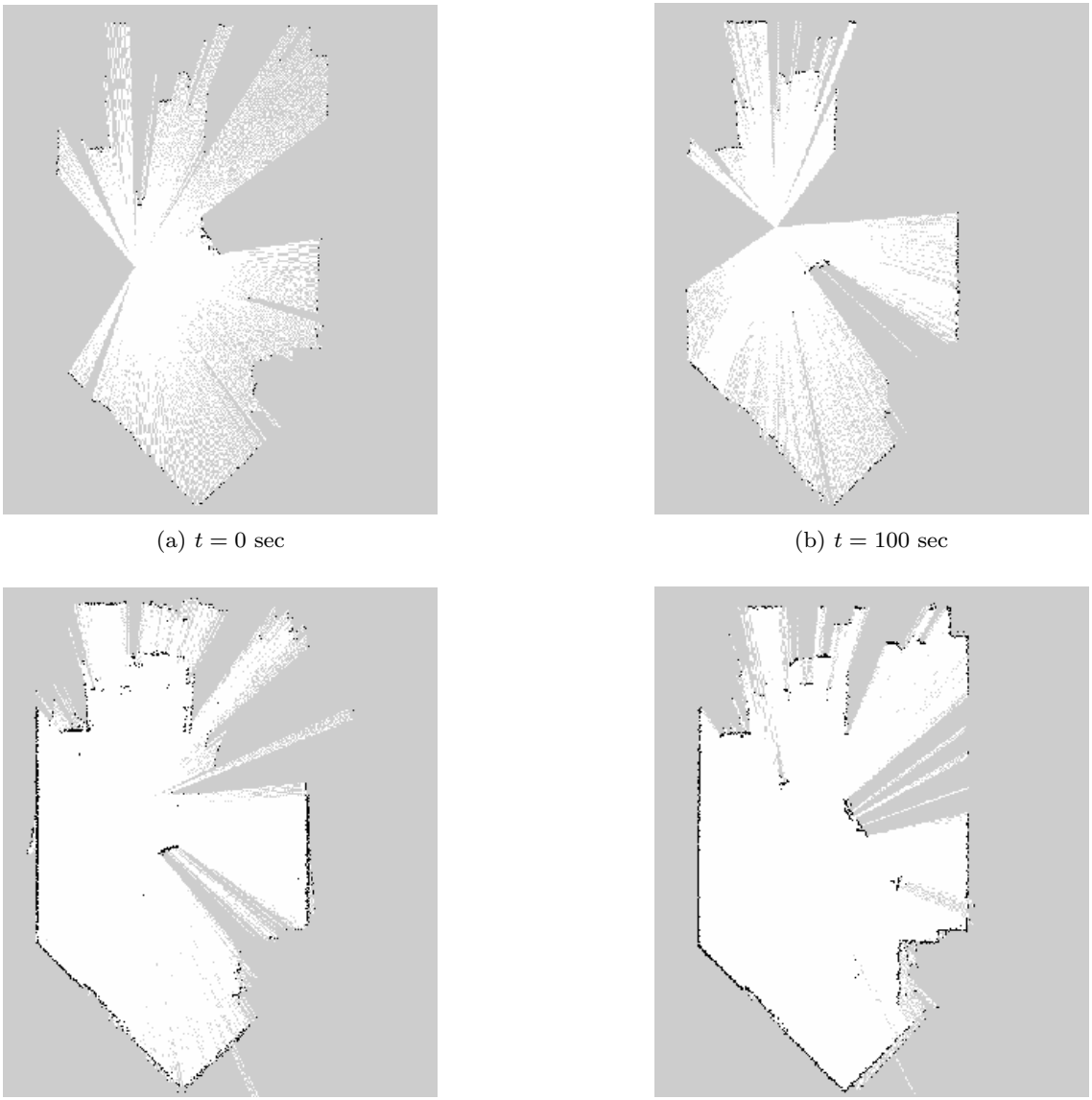


Fig4.28: Evaluation of SLAM Localization in Simulated Environment (**Fig.** 4.26)

Table4.1: Experimental Result: Effect of Approximation Based on Continuous Planar Environment Assumption with Different Environment Models

Environment	Continuous Planar Environment Assumption	Time to Plan	Number of Open-list	Number of Closed-list
Flat	True	0.096 sec	4332	306
	False	0.24 sec	4332	306
Stairs	True	0.43 sec	6162	662
	False	0.97 sec	2590	662
Hills	True	0.078 sec	2825	177
	False	0.21 sec	2705	171



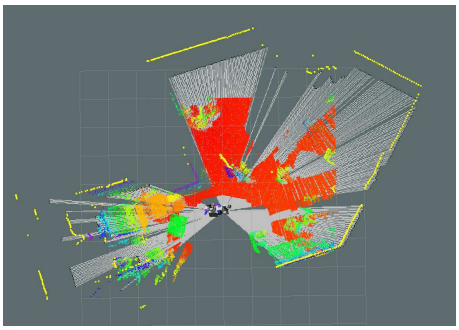
(a) $t = 0$ sec (b) $t = 100$ sec

(c) $t = 200$ sec (d) $t = 300$ sec

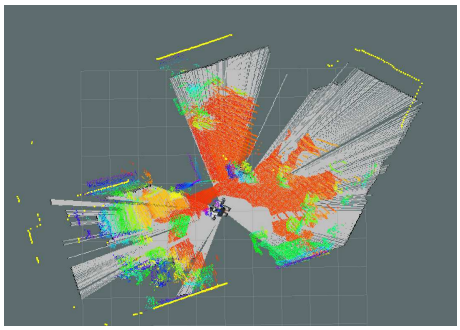
Fig4.29: Built Global Map in Actual Environment

Table 4.1 の実験結果から環境平面仮説に基づく近似を導入することで探索時間が約 2 倍程度短縮されている. 特に階段環境 (**Fig.** 4.32) における実験結果では, プランニング終了時の Open List に残っていたノード数に違いが見られる. これは第 4.3.5 節において述べたローカル足位置修正によって増加した足配置候補のうち, 認識処理が遅延されたことによって環境に設置できない足配置が Open List に残ったまま評価されずにプランニングが終了していることを意味する.

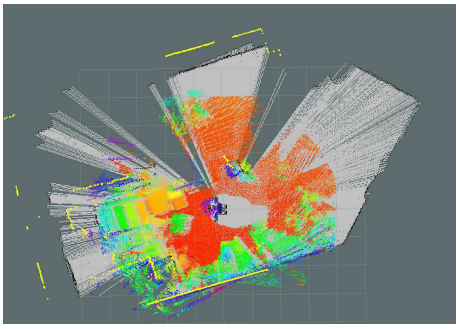
また, これらのシミュレーション環境に対して, 平面認識を環境全体に実行することで必要



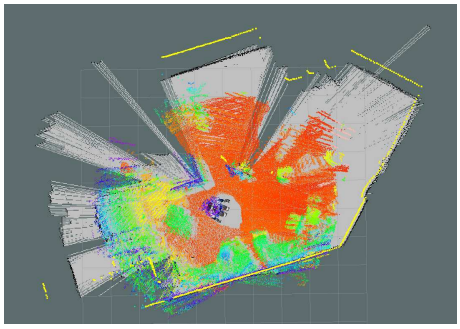
(a) $t = 0$ sec



(b) $t = 100$ sec



(c) $t = 200$ sec



(d) $t = 300$ sec

Fig4.30: Built Heightmap with SLAM Localization in Actual Environment

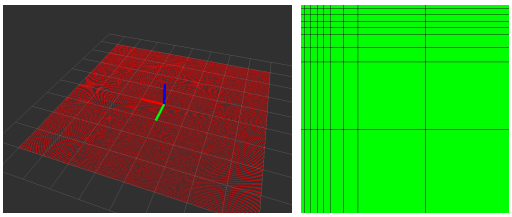


Fig4.31: Flat PointCloud Environment for Simulation Experiments

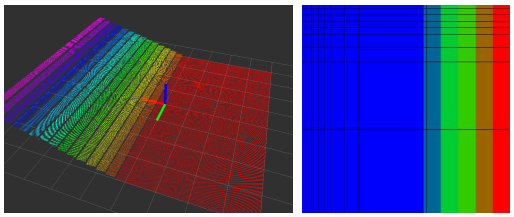


Fig4.32: Stairs PointCloud Environment for Simulation Experiments

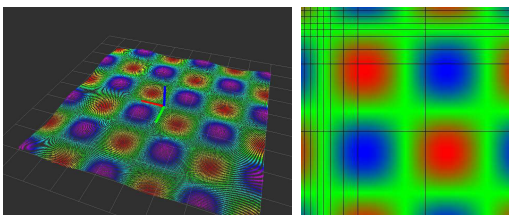
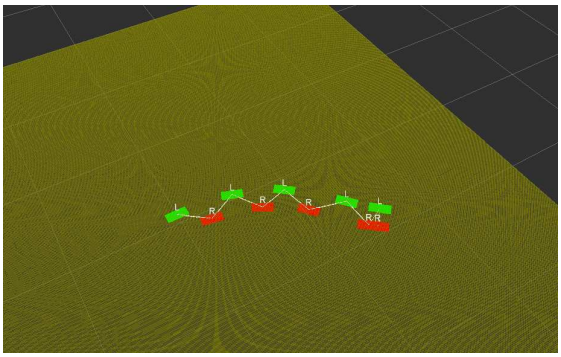
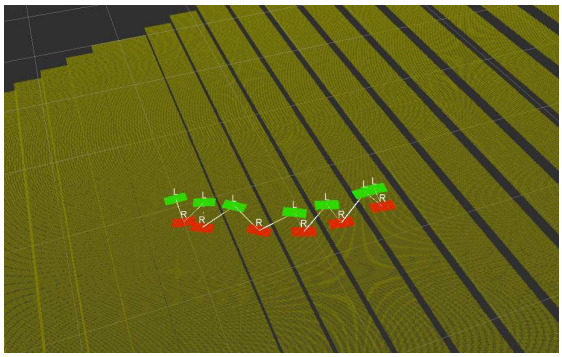


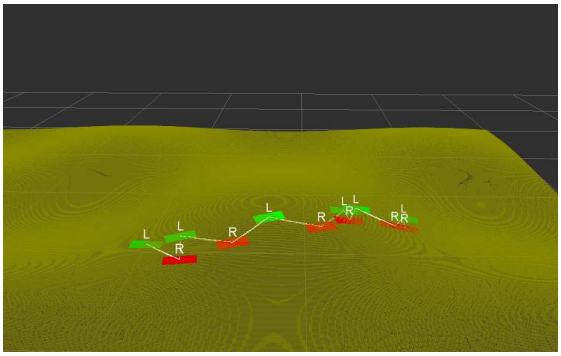
Fig4.33: Hills PointCloud Environment for Simulation Experiments



(a) Planned Footsteps in Flat PointCloud Environment



(b) Planned Footsteps in Stairs PointCloud Environment



(c) Planned Footsteps in Hills PointCloud Environment

Fig4.34: Result of Footstep Planning in Simulated Environment

となる時間を計測した (**Table** 4.2). 複数平面認識には Region Growing[94] を用いた. 本章で提案した計画時に領域に依存した要求駆動型の認識器を走らせることで **Table** 4.2 で必要としているような環境をモデル化するための認識時間を無視することができる. また, Hills 環境においては, 環境が曲面から構成されるため, このような平面認識では環境をモデル化することができない. このような環境でも足配置計画を行うことができるのは本章で提案した手法の大きな利点である.

これらの結果を第 2 章でのべた計算時間 t の関係として整理する (**Table** 4.3).

Table4.2: Experimental Result: Time to Model Simulated Environment by Multi-Plane Detection based on Region-Growing.

Environment	Normal Estimation	Region Growing
Flat	0.22 sec	8.99 sec
Stairs	0.24 sec	7.75 sec
Hills	0.22 sec	N/A

Table4.3: Experimental Result: Comparison of Duration of Perception and Motion Planning between Collaborative and Separative Scheduling Approach

Scheduling Approach	Environment Model	t_P	t_M	$t_P + t_M$
Collaborative Scheduling	Flat	0.018	0.082	0.1
	Stairs	0.09	0.09	0.18
	Hills	0.038	0.071	0.11
Separative Scheduling	Flat	9.21	0.082	9.3
	Stairs	7.99	0.09	8.1

4.7　おわりに

本章では, 足配置計画法を題材として, 動作計画と認識器を協調的に動作させることで高速に動作計画を行う手法について提案した. 提案手法の特徴は通常環境構造を認識しモデル化した後に動作計画を行うところを, 動作計画と認識処理を協調的に実行させることで動作計画の中で必要となる領域のみを認識・モデル化することであった. 認識器は動作計画器から手がかりとして認識すべき場所の情報を得ることで, 動作計画器が必要とする領域のみに認識処理を走らせることができ, 高速化が可能であった. 必要となる認識回数を削減し高速化するため, 計画アルゴリズムに環境が連続した平面から構成されるという仮説に従った近似を導入した. また, 事前認識処理として SLAM を利用した密な 3 次元点群による環境モデルの構築法を示した. 最後に実験を通してこれらの近似や協調的サブスケジューリングによる高速化の有効性について確認した.

第 5 章

継続的認識機能のための 3 次元点群追跡による物体認識器

5.1 はじめに

本章では継続的認識機能を実現するため, パーティクルフィルタを用いた 3 次元点群追跡手法について述べる (**Fig.** 5.1). 継続的認識機能は第 2 章において述べた移動中認識実行モデル

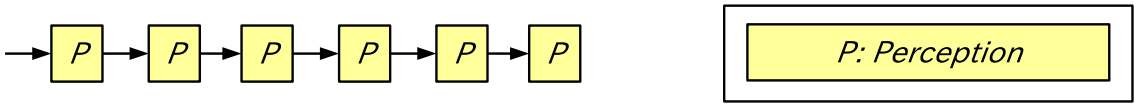


Fig5.1: Scheduling Overview of Continuous Perception

(第 2.5.3 節) および環境の監視機能 (第 2.3.4 節) の実現のために重要な認識機能である.

3 次元点群の位置推定アルゴリズムとしてはパーティクルフィルタを利用する. これは, 観測回数が増えるにしたがって認識精度が向上し, 時系列フィルタリングによって高速な実装が可能であるためである. 本章では, 3 次元センサによって取得したセンサデータから参照点群を作成し, その点群の位置姿勢の変化を 6 自由度で追跡する手法を示す (**Fig.** 5.2). アルゴリズムの概要は以下のような流れに従う.

1. 参照となる 3 次元点群モデルを取得する
2. 2 種類の動作モデルを利用してパーティクルの状態ベクトルを更新する.
3. 2 点間の距離に基づいた利用してパーティクルの重みを計算する
4. パーティクルの重みを正規化することで, 確率密度関数を得る

トラッキングにおいて 3 次元点群によって表現された物体モデルを利用する難しさは, その

データの大きさによって生じる探索や座標変換の計算コストの高さである．実時間でのトラッキングを実現するために, 本章では点群の 2 点間の距離に基づいた観測モデルを提案し, 実時間処理可能な最適化によって計算コストを削減する．

評価制御機構を備えたロボットシステムの中では, 精度と計算速度をパラメータによって制御可能な認識器として位置づけられる．これらのパラメータに関して, 点群追跡の高速化の中で計算量と関係して言及する．代表的なパラメータは 3 次元点群の低解像度化のための立方体格子の大きさ, パーティクル数である．

5.2　関連研究

物体の位置と姿勢の 6 自由度推定問題は画像処理の分野で広く研究されている．物体の位置と姿勢の 6 自由度を得ることはマニピュレーションのようなタスクにおいて認識結果を利用する上で基本となる情報である．それに加えて, 物体の位置姿勢をトラッキングすることは重要である．トラッキングによって可能となる応用としては, ビジュアルサーボや環境の監視などがある．

2 次元画像処理を用いた物体認識として, エッジ特徴量に基づいた手法が研究されている．Lowe は直線エッジを利用した物体認識手法を提案している [95, 96]．Vacchetti らは局所的なバンドル調整の問題としてトラッキング問題を定式化し, エッジ特徴を利用した高速なトラッキング手法を提案している [97, 98]．[99] においては, 任意の 3 次元形状に関してエッジ特徴量を利用したトラッキング手法が提案されている．任意形状を対象とするため, 自由表面形状の表現には NURBS(Non-Uniform Rational Basis Spline) が利用される．また, 3 次元形状のモデルから 実時間でエッジ画像を生成するため GPU の利用が大きく貢献している．しかしながら, このような 3 次元形状モデルを利用した 2 次元画像を入力とする推定手法は高い精度の 3 次元形状モデルが必要となる．

SIFT や SURF といった 2 次元画像の局所特徴量を利用することで対象となるテクスチャの位置姿勢の 6 自由度を推定することができる [100, 101]．2 次元画像の局所特徴量における難しさは, 物体の見え方が変化しても特徴量が変化しない, Affine 不変な特徴量を作ることである．Fern[102] は特徴点周辺の画像パッチを Affine 変換させ学習データを生成し, 特徴量をランダムフォレストを利用して学習させた局所特徴量である．学習データを Affine 変換によって生成しているため, 生成された特徴量は Affine 変換に対して頑強な特徴量が生成される．ASIFT[103] は SIFT 特徴量を Affine 変換に対して頑強になるように修正した特徴量であ

る. SIFT 特徴量計算の中で, 特徴点周囲の画像を拡大縮小することでスケール不変な特徴を得るように, 特徴点周囲の画像をカメラの z 軸を傾けた画像も考慮することで Affine 変換に対して頑強な特徴を得る. しかしながら, このような局所特徴量を利用して物体の位置姿勢の 6 自由度を推定するためには, 事前にテクスチャと特徴点の 3 次元位置の対応付けが必要となる. この問題を解決するため, Feature Harvesting 法が提案されている [104]. Feature Harvesting 法では局所特徴点を追跡すると同時にその見えと幾何情報を同時に学習する. 特徴点追跡によって様々な視点からみた画像を取得することが可能であり, それらの画像を使って見え方の変化を含んだ特徴量を学習することができる. Feature Harvesting 法では事前の 3 次元知識は必要ではないが, 十分な数の特徴点を検出するためのテクスチャが存在する物体にのみ適用可能な手法である. また, SIFT や SURF といった 2 次元局所特徴量の算出は計算コストが高く, トラッキングに用いるのが難しい.

これら 2 次元画像を入力とした位置姿勢の推定手法の一方で, 3 次元点群を利用した物体位置推定手法も提案されている. ICP (Iterative Closest Point) は 2 つの 3 次元点点群が与えられた時に, それらの 3 次元点点群同士の距離が最も小さくなるように位置合わせをする手法である [105]. 3 次元点点群の位置のみではなく, 色も考慮することで位置合わせの精度を向上させる手法も提案されている [106]. Kinect Fusion は深度画像全体の ICP を GPU を利用することで実時間で実装することを可能としている [107]. ICP は反復的なアルゴリズムであるため, 並列化が難しくマルチコアの恩恵を受けることが難しい. また, ICP では事前に精度良い初期位置が必要であり, 位置合わせの探索手法が局所的であるため, ICP をトラッキングに用いるためにはトラッキング中には常に ICP による位置合わせが精度よく成功していなくてはいけない. また, GPU での最適化による制約のため, kinect fusion では入力となる 3 次元点群は Organized PointCloud と呼ばれる画像と同じデータ構造を持つ 3 次元点群データに限られる.

2 次元画像の局所特徴量と同様に, 3 次元点群の局所特徴量も提案されている [108, 109, 110, 111]. spherical harmonic invariants[108] や integral volume descriptors[109] は回転と並進に関して不変な 3 次元特徴であるが, 深度カメラの 3 次元点群はノイズが大きく, これらの特徴の利用が難しい. Point Feature Histgrams (PFH)[110] はノイズに頑強な特徴量として考案されている. しかしながら, PFH は密な点群と高い計算コストを必要とするため, トラッキングや実時間の認識には不向きである.

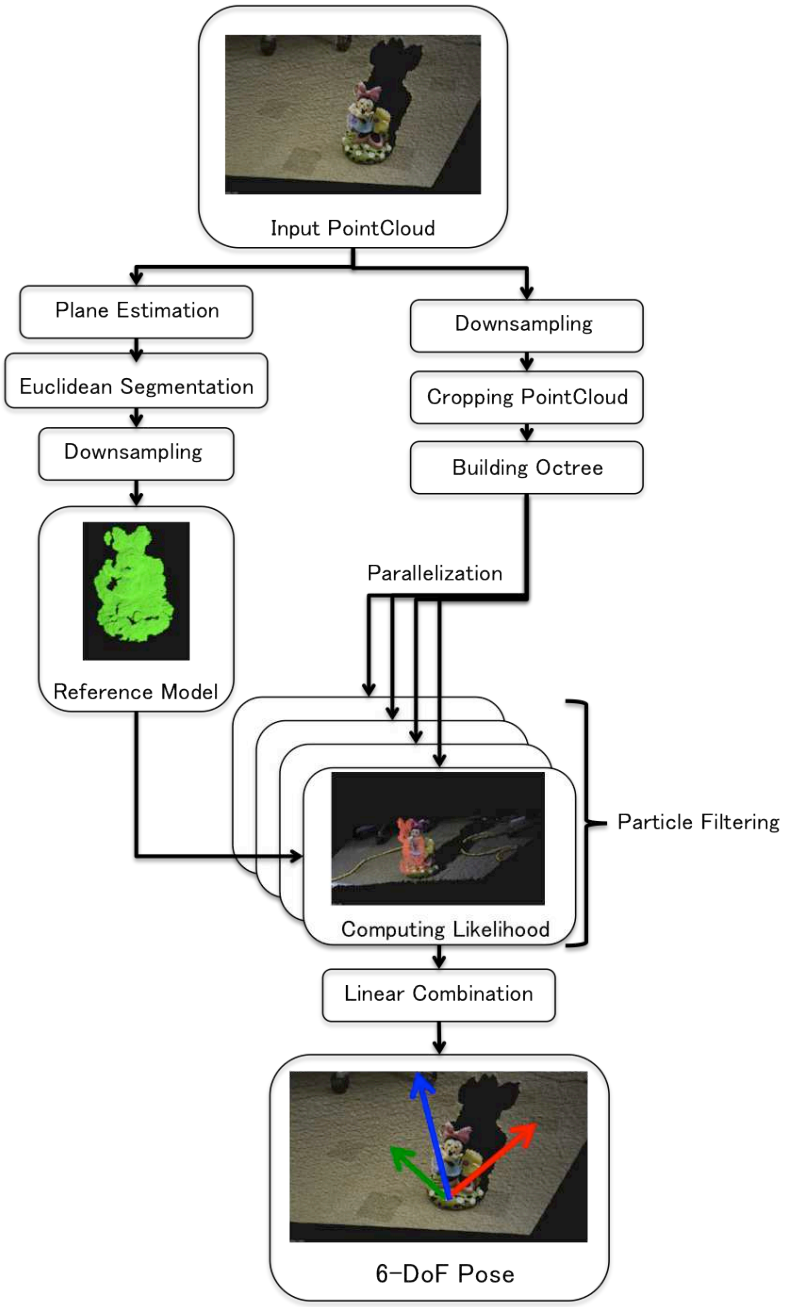


Fig5.2: Overview of Tracking Objects using PointCloud

5.3 パーティクルフィルタを用いた物体状態推定

本章で述べる 3 次元点群追跡は再帰ベイズ推定手法の一つであるパーティクルフィルタを利用する [112]. 再帰ベイズ推定は観測ステップと予測ステップを繰り返すことで確率的な記述に従って状態量を推定していく手法である. そのため時間に依存した機能の実現が容易である.

推定すべき確率密度を $p(\boldsymbol{x}_k|z_{1:k})$ としたとき, 予測ステップとは $p(\boldsymbol{x}_k|z_{1:k-1})$, 観測ステップとは $p(\boldsymbol{x}_k|z_{1:k})$ を推定することにほかならない. ここで, $\boldsymbol{x}_k$ は時刻 k における推定したい対象となる状態量, z_k は時刻 k におけるセンサによる観測データである. 予測ステップはベイズの条件付き確率により以下のように記述される.

$$p(\boldsymbol{x}_k|z_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|z_{1:k-1})d\boldsymbol{x}_{k-1} \tag{5.1}$$

$p(\boldsymbol{x}_k|z_{1:k-1})$ は時刻 1 から $k-1$ までのセンサデータをもとに予測された時刻 k における状態量 $\boldsymbol{x}$ の確率密度である. これはひとつ前の推定結果 $p(\boldsymbol{x}_{k-1}|z_{1:k-1})$ とモーションモデルと呼ばれる $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ に従う. $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ は予め推定対象に従って定義される予測モデルである.

一方観測ステップは以下のように記述される.

$$p(\boldsymbol{x}_k|z_{1:k}) = \frac{p(z_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|z_{1:k-1})}{\int p(z_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|z_{1:k-1})d\boldsymbol{x}_k} \tag{5.2}$$

$p(\boldsymbol{x}_k|z_{1:k})$ は時刻 1 から k までのセンサデータをもとに予測された時刻 k における状態量 $\boldsymbol{x}$ の確率密度である.

パーティクルフィルタはモンテカルロシミュレーションにより再帰ベイズ推定を解く手法である (**Fig.** 5.3). パーティクルフィルタでは, 確率密度をパーティクルと呼ばれる状態量の仮説の集合として表現する.

$$p(\boldsymbol{x}_k) = \{\boldsymbol{p}_{k,i}\}, 0 \leq i \leq N \tag{5.3}$$

$$\boldsymbol{p}_{k,i} = \begin{pmatrix} \boldsymbol{x}_{k,i} \\ w_{k,i} \end{pmatrix} \tag{5.4}$$

パーティクル $\boldsymbol{p}_{k,i}$ は状態量 $\boldsymbol{x}_{k,i}$ と重み係数 $w_{k,i}$ からなるベクトルである. パーティクルを用いることで予測ステップを簡潔に記述することができる.

$$\boldsymbol{p}_{k,i} = f(\boldsymbol{p}'_{k-1,i}) + \boldsymbol{w}_k \tag{5.5}$$

f はモーションモデルであり, 特に静止モデルを利用する場合 **eq.** 5.6 となる.

$$\boldsymbol{p}_{k,i} = \boldsymbol{p}'_{k-1,i} + \boldsymbol{\sigma}_k \tag{5.6}$$

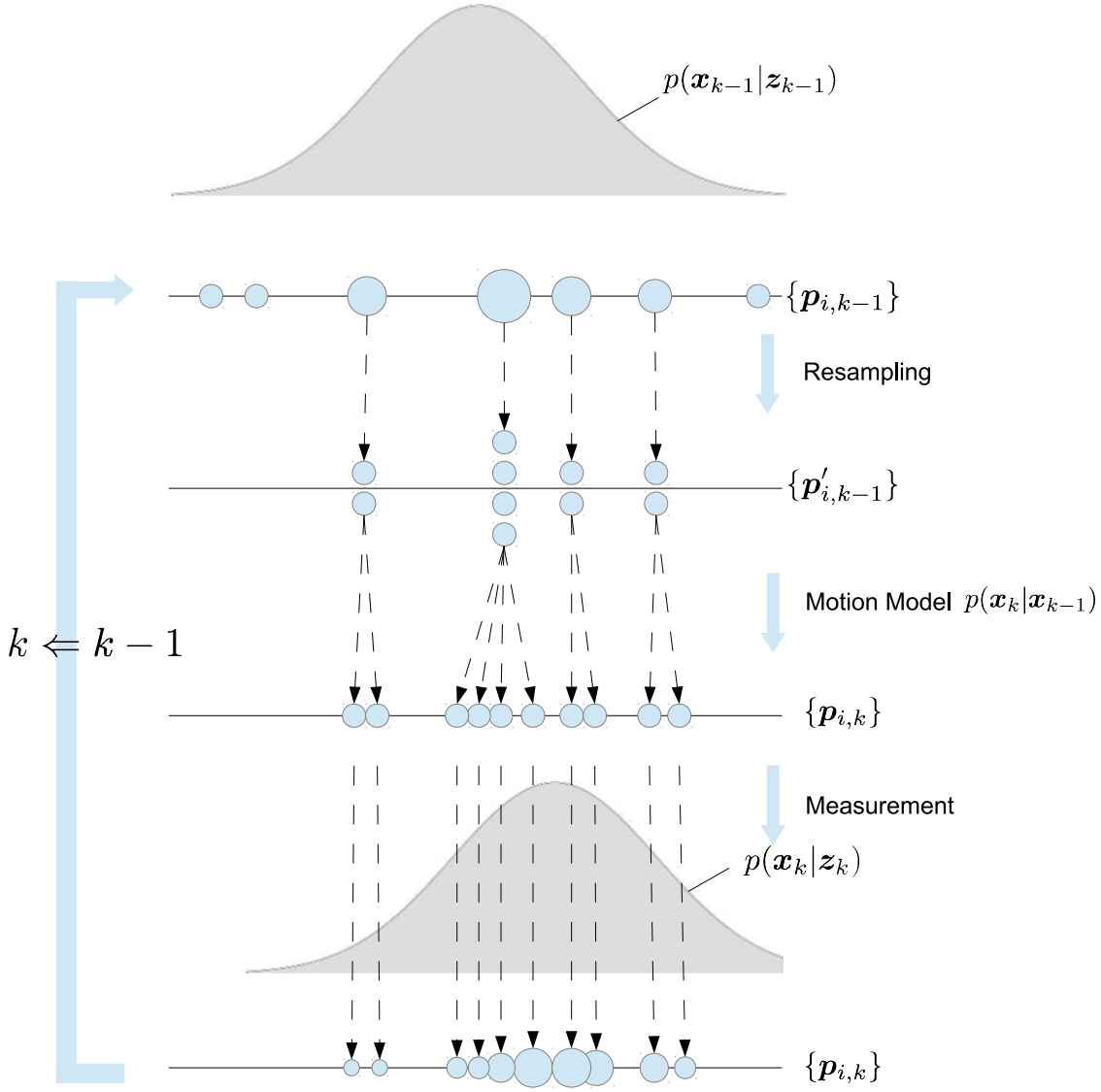


Fig5.3: Overview of Particle Filtering

$\boldsymbol{\sigma}_k$ はノイズパラメータであり , 白色ノイズが用いられる .

$$\boldsymbol{\sigma}_k \sim N(\boldsymbol{\mu}, \Sigma) \tag{5.7}$$

$\boldsymbol{p}'_{k-1,i}$ は非復元抽出に従って $\boldsymbol{p}_{k-1,i}$ から生成されたものである . このとき , $\boldsymbol{p}'_{k-1,i}$ と $\boldsymbol{p}_{k-1,i}$ の対応は重み係数 $w_{k-1,i}$ に従って決定される . この作業をパーティクルフィルタでは resampling と呼ぶ.

観測ステップでは重み係数 $w_{k,i}$ を **eq.** 5.8 に従って計算する .

$$w_{k,i} = \frac{l(\boldsymbol{p}_{k,i}|z_k)}{\sum l(\boldsymbol{p}_{k,i}|z_k)} \tag{5.8}$$

$l(\boldsymbol{p}_{k,i}|z_k)$ はパーティクル $\boldsymbol{p}_{k,i}$ のセンサデータ z_k に対する尤度関数である．観測するセンサデータが複数ある場合, 観測ステップは **eq.** 5.9 のようになる．

$$w_{k,i} = \frac{l(\boldsymbol{p}_{k,i}|z_k, y_k)}{\sum l(\boldsymbol{p}_{k,i}|z_k, y_k)} \tag{5.9}$$

ここでは簡単のため 2 つの観測データ z_k, y_k のみがあるとした．

5.4　幾何的な関係性を考慮した初期分布記述

物体位置を推定する場合, 環境との幾何的な拘束関係を考慮することで, 推定を容易にすることができる．これは Tabletop Manipulation で特によく使われる手法である．Tabletop Manipulation では机上に物体が存在することが前提条件となるため, 机面の高さが既知であったりセンサから机面を推定することで, 机上の物体を検出することができる．

このような幾何的な関係性を考慮するため, 本節ではセンサモデルを拡張して複数の認識器を統合する．ロボットに搭載された n 個のセンサデータを $Z_k = \{z_{k,1}, z_{k,2}, \cdots, z_{k,n}\}$ とする．推定する状態量を $\boldsymbol{x}_k$ とすると, **eq.** 5.9 より

$$w_{k,i} = \frac{l(\boldsymbol{p}_{k,i}|Z_k)}{\sum l(\boldsymbol{p}_{k,i}|Z_k)} \tag{5.10}$$

となる．さらにすでに既知である他の状態量を $Y_k = \{y_{k,1}, y_{k,2}, \cdots, y_{k,n}\}$ とすると,

$$w_{k,i} = \frac{l(\boldsymbol{p}_{k,i}|Z_k, Y_k)}{\sum l(\boldsymbol{p}_{k,i}|Z_k, Y_k)} \tag{5.11}$$

のように他の状態量もセンサデータの一種として統合可能である．

Fig. 5.4 に幾何的な関係性の具体例を示す．図中では Door, Floor, Chair, Desk, Can, Remote-Controller, Carton の 7 種類の物体を対象に矢印は幾何的な関係を意味している．それぞれの幾何的な関係の意味を **Table** 5.1 に示す．ここでは, Door の位置は部屋の原点と同じ意味で利用している．これらの幾何的な関係性はいくつかの種類に分類可能である．

1. 床面からの距離 (1-2)
2. 物体同士の接触状態 (3-6)
3. 環境における物体位置 (7-12)

ロボットによって環境を認識することを考えるとさらにこれらの関係性にロボットの位置を考慮しなくてはいけない．この分類に従うと, 1 および 2 の違いは些細である．3 番目の環境における物体位置は利用するためには以下の 3 点を満たす必要がある．

Table5.1: Geometric Relationship in **Fig.** 5.4

Arrow Number	From Object	To Object	Description
1	Floor	Desk	Height of desk from floor is static and can be pre-defined.
3	Floor	Chair	Height of chair from floor is static and can be pre-defined.
3	Desk	Remote-Controller	Remote-Controller is located on Desk.
4	Desk	Tray	Tray is located on desk.
5	Tray	Can	Can is located on tray.
6	Desk	Carton	Carton is located on Desk.
7	Door	Chair	Location of chair can be defined from door if location of chair is static.
8	Door	Desk	Location of desk can be defined from door if location of desk is static.
9	Door	Tray	Location of tray can be defined from door if location of tray is static.
10	Door	Can	Location of can can be defined from door if location of can is static.
11	Door	Carton	Location of carton can be defined from door if location of carton is static.
12	Door	Remote-Controller	Location of remote-controller can be defined from door if location of remote-controller is static.

- 対象物の環境中の位置が既知であること
- 対象物が環境中で静的であること
- ロボットの環境中での自己位置が精度よく推定できていること

これら 3 点が満足できる場合, 環境における物体位置は良く機能する. しかし, これらの信頼性が低い場合, 物体位置の候補を示す確率分布を広くしなくては物体の発見は難しくなる. 確率密度の初期分布を広くしていくと, 推定結果に False-Positive を含みやすくなってしまう. このような状況において, 他の物体との位置関係を利用できると, 確率密度の初期分布を効果的に定義可能である.

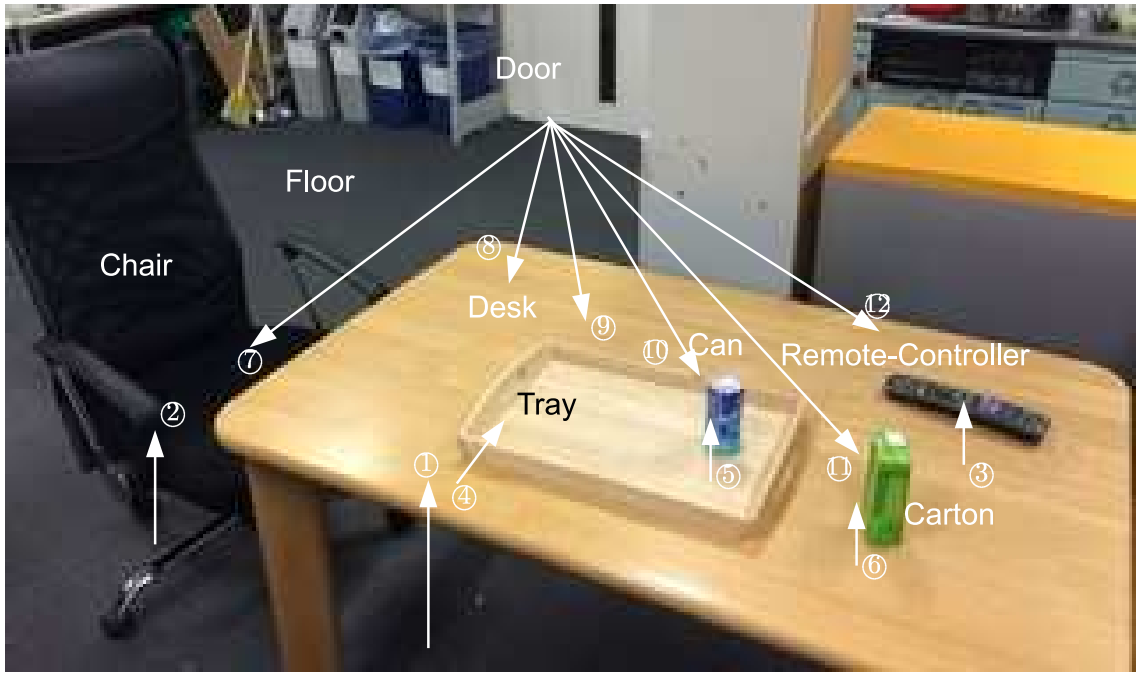


Fig5.4: Typical Environmental Setup and Geometric Relationship

5.4.1　環境の平面構造に着目した幾何的な関係性を用いた探索範囲の設定

環境中の平面構造に着目することで, 物体認識を有利にすすめることができる. 先に述べたように, 環境中に存在する物体は特殊な場合を除き, 環境から何らかの支持を受ける. そのような支持には以下のようなものが考えられる.

1. 壁面に貼りつけられた物体
2. 机や床などの水平な平面上に設置された物体

3. 人によって把持されている物体

このように, 平面から支持を受けているという対象物が環境中では多くを占めると考えられる.

3 次元点群から平面を推定する手法としては様々な手法が提案されている. 本論文では物体認識を補佐するための平面構造の認識手法として Region Growing [94] による複数平面認識を利用する. Region Growing によるセグメンテーションの結果として, 平面を構成すると考えられる点群のクラスタが得られる. 各クラスタ点群に対して RANSAC[93] によって平面モデルを当てはめることによって最終的な認識結果とする. 3 次元点群が構成する平面を推定する手法として RANSAC による平面推定は最もよく使われる手法である. しかしながら, 3 次元点群が複数の平面を構成する場合, 期待する推定結果が得られない. ここまでの議論で Region Growing によって複数の平面含む 3 次元点群はそれぞれの平面を構成する 3 次元点群のクラスタとして分割している. したがってこれらのクラスタから平面のモデルを推定するためには RANSAC による平面モデルの当てはめが機能する. **Fig.** 5.5 に実際に実環境中の平面構造を回転機構レーザによって取得した 3 次元点群から推定した結果を示す.

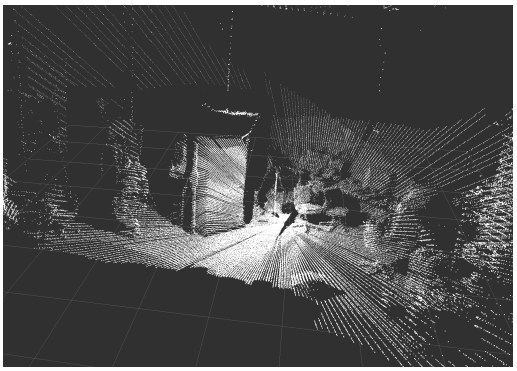
5.4.2　パーティクルフィルタでの幾何的な関係性の利用

パーティクルフィルタを利用した物体認識を設計する上で, 主に変更しうる点は以下の 3 つである.

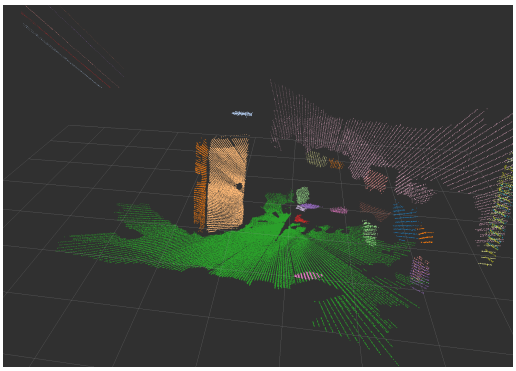
1. パーティクルが持つ状態ベクトル設計
2. パーティクルの初期分布設計
3. センサデータに対するパーティクルの重みを計算するための観測モデル

推定対象が決定すると状態ベクトルの設計はそれに従って決定される. 幾何的な関係性を利用することで変わりうるのは特にパーティクルの初期分布設計と観測モデル設計の 2 つである.

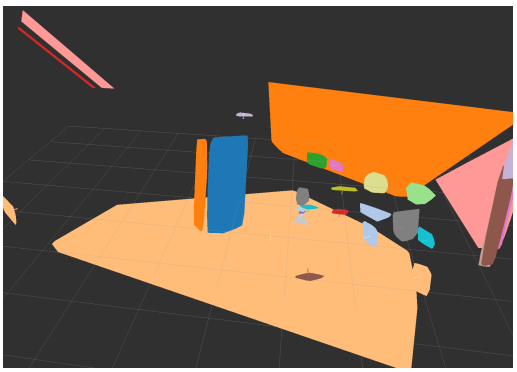
パーティクルフィルタによって探索する範囲が広い場合, パーティクルの初期分布を適切に設定することが必要不可欠である. マニピュレーション対象物となる物体の位置を推定する場合, その座標を用いて初期分布を設定することが最も容易な方法である. その場合, 対象物のおおよその位置が既知であることが必要である. ロボットが活動する環境の地図が事前に与えられている場合, その環境地図内で事前位置が既知であれば良い. ロボットの位置は SLAM に代表される位置推定手法によって得ることができる. しかしながら, ロボットの環境に対する位置推定精度が低かったり, 環境地図が既知でない場合, 環境に対する位置を利用した初期分布



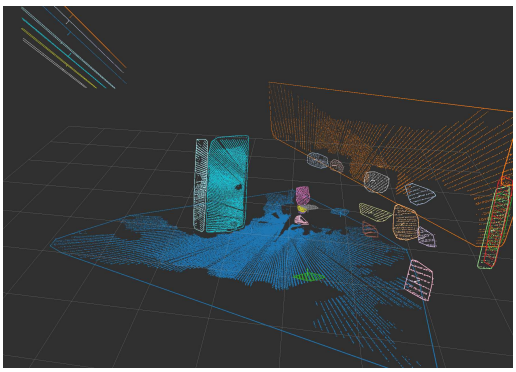
(a) Input PointCloud



(b) Clustering Result by Region Growing



(c) Estimated Plane Models by RANSAC Refinement



(d) Estimated Planes and Segmented PointCloud

Fig5.5: Region Growing Based Segmentation and Plane Estimation

設計は機能しない.

その一方で, 環境地図は既知ではないが, 環境のおおよその構造が事前に明らかであるという場合が考えられる. 特に遠隔操縦によるタスク遂行ではこのような条件があてはまる. 例えば先に述べた DRC 競技会においては, 事前に **Fig.** 5.6 のような環境でタスクを遂行しなくてはいけないことは明らかであった. **Fig.** 5.6 から事前に得られる知識としては以下のようなものが考えられる:

- ドアを通り抜けた後, 右側の壁近くにバルブが設置してある
- バルブの更に先にドリルが壁から水平に突出した段上に設置してある
- ドリルに対してドリルによって切削する壁は右側に設置してある

また, これらのおおよその距離もわかると考えられる. これらの知識を元に, 先に述べたように環境に対する位置を利用した初期分布設計を行った場合, 正確な対象物の位置が得られていな

Fig5.6: Overview of Environment in DRC Competition

いため, 分布を大きくする必要があり , 多くの False-Positive な推定結果を生んでしまう .

このような問題を解決するため, パーティクルの初期分布を平面の幾何的な関係性を利用して決定する . 平面の幾何的な関係性は以下の 2 点で利用する :

- パーティクルの分布範囲の制限
- 平面ごとの評価値を考慮し , パーティクルが属する平面の決定

パーティクルの分布範囲を決定するにあたって, 平面拘束を導入することで, 平面に束縛された対象物を探索するために適したパーティクルの初期分布を設計することができる . **Fig.** 5.7 に対象物を支持する平面が与えられた時に, パーティクルが存在可能な領域を図示した . これを実際のセンサデータから推定された平面に基づき分布させると , **Fig.** 5.8 のようになる .

支持平面上にパーティクルを分布させるには以下の手順に従う .

1. (x, y) に関して, 支持平面を包括するバウンディングボックスの大きさに従って一様分布から決定する .
2. (x, y) が支持平面内に入っていなければ 1 へ戻る .

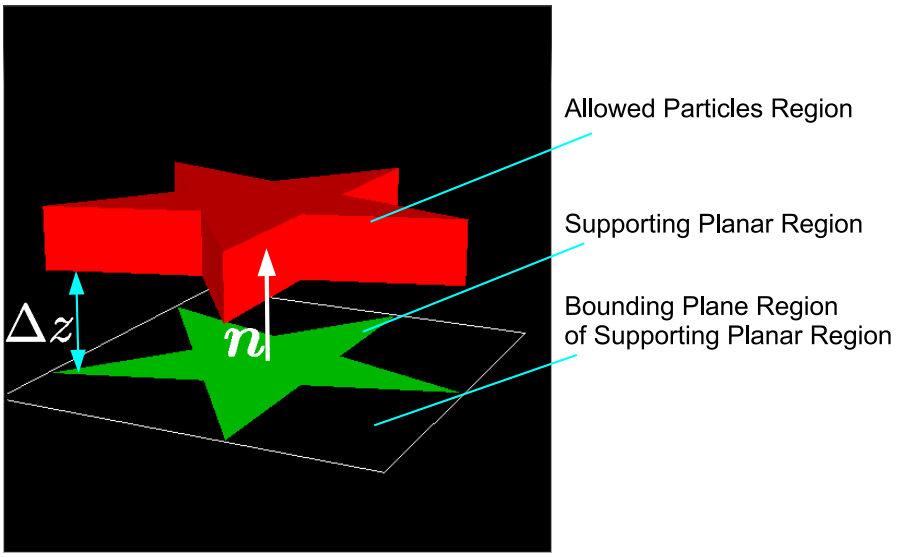


Fig5.7: Allowed Spatial Region for Particles

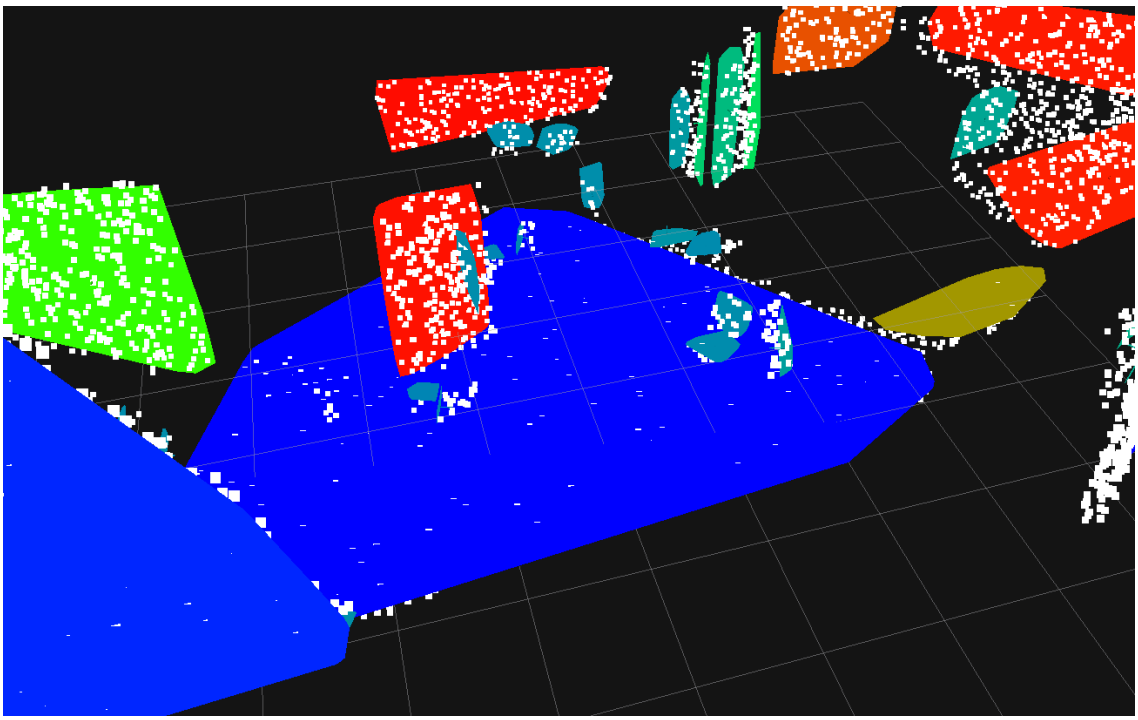


Fig5.8: Initialized Particles on Estimated Planes

3. z に関しては支持平面座標系において相対的な範囲を与えておき, その範囲内で一様分布から決定する
4. 姿勢に関しては, 支持平面の法線ベクトル $\boldsymbol{n}$ に対して x, y 軸周りの回転角に対して相対的に決定する.

また, 複数の平面が環境中に存在する場合, どの平面に対して初期パーティクルを分布させるかという問題が生じる. 複数の平面に関して優先度を定義するために, 以下のような指標に関して評価値を計算する.

1. 平面の向き $l_{\text{orientation}}$

   平面の法線ベクトルを $\boldsymbol{n}$ とした時に, 与えられた方向ベクトル $\boldsymbol{m}$ に対して, $\theta = \arccos(\boldsymbol{m} \cdot \boldsymbol{n})$ を求める. 予め θ_{offset} を与えておき, 評価値は以下の式によって決定する.

$$l_{\text{orientation}} = \frac{1}{1 + (\theta - \theta_{\text{offset}})^2} \tag{5.12}$$

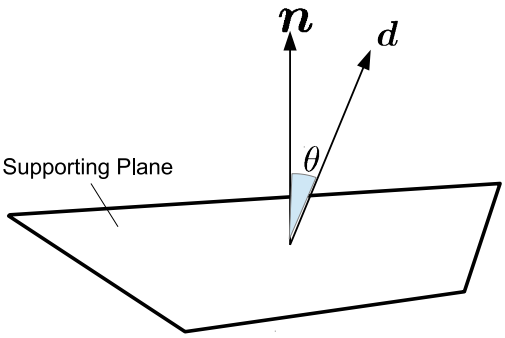


Fig5.9: Angular Difference θ

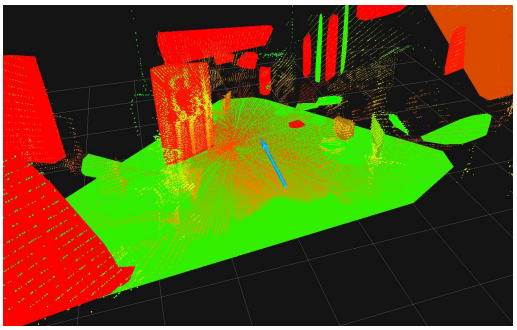


Fig5.10: Orientation Based Likelihood on Actual Sensor Input

2. 平面までの距離 l_{distance}

   与えられた点 $\boldsymbol{p}$ から平面上で最短距離を与える点までの距離 d を求める. 予め d_{offset} を与えておき, 評価値は以下の式によって決定する.

$$l_{\text{distance}} = \frac{1}{1 + (d - d_{\text{offset}})^2} \tag{5.13}$$

3. 平面の大きさ l_{area}

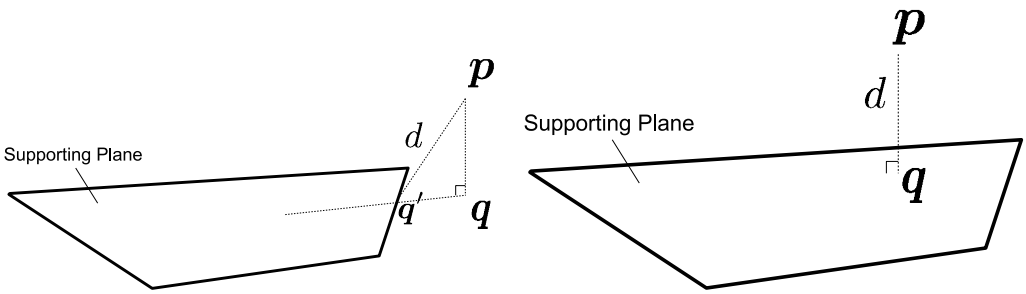


Fig5.11: 3-D Point to Polygon Distance d Definition

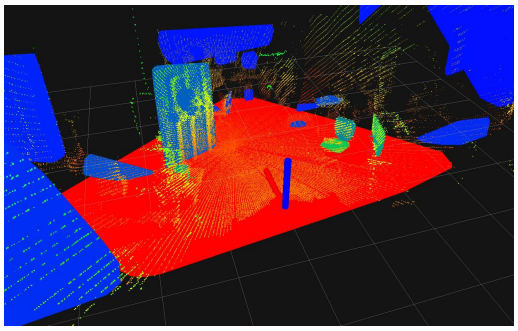


Fig5.12: Distance Based Evaluation Value on Actual Sensor Input

与えられた面積 a と, 平面の面積 r に対して, **eq.** 5.14 に従って評価値を決定する.

$$l_{\text{area}} = \frac{1}{1 + (a - r)^2} \tag{5.14}$$

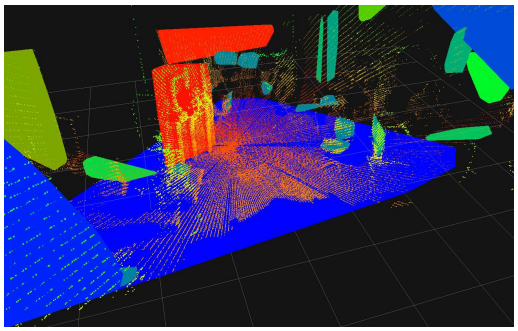


Fig5.13: Area Based Likelihood on Actual Sensor Input

4. 平面の方向 $l_{\text{direction}}$

点 $\boldsymbol{p}$ と方向ベクトル $\boldsymbol{d}$ が与えられた時, その方向に平面が存在しているかに従って評価

値を決定する. 点 $\boldsymbol{p}$ を通り 方向ベクト ルが $\boldsymbol{d}$ である直線 l が平面と 交わる時, $\theta = 0$ であるとする (**Fig.** 5.14 左). 交わらない場合, 直線 l とその平面上で直線 l と 最近傍となる点 $\boldsymbol{q}$ に関して, 直線 l と点 $\boldsymbol{p}, \boldsymbol{q}$ を通る直線のなす角を θ とする (**Fig.** 5.14 右). 予め θ_{offset} を与えておき, 評価値は以下の式によって決定する.

$$l_{\mathrm{direction}} = \frac{1}{1 + (\theta - \theta_{\mathrm{offset}})^2} \tag{5.15}$$

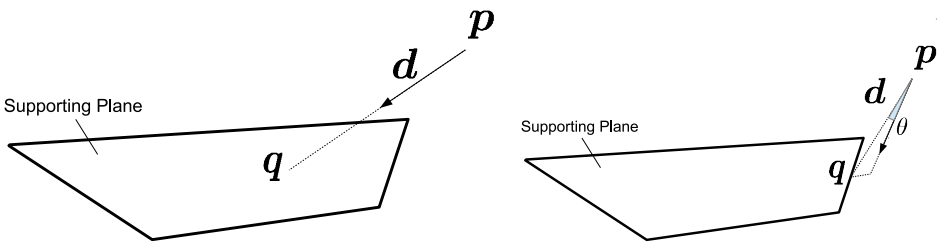


Fig5.14: Polygon Direction Evaluation Value

これらの平面評価値は複数の指標を 利用するため, 重ね合わせることで最終的な評価値とする.

5.5　パーティ クルフィ ルタによる 3 次元点群を用いた 6 自由度物体位置追跡

5.5.1　パーティ クルフィ ルタによる 3 次元点群追跡

時刻 k における各パーティクルの状態ベクト ルを $\boldsymbol{p}_{k,i}$, 重みを $w_{k,i}$ とする. パーティクルフィルタではパーティクルの重みが確立密度関数となるという 前提を 置くため (**eq.** 5.8), 各パーティクルの和は 1 となる (**eq.** 5.16).

$$\sum_i w_{k,i} = 1 \tag{5.16}$$

i 番目のパーティクルの状態ベクト ル $\boldsymbol{p}_{k,i}$ は, 位置と 姿勢の 6 パラメータから 構成される.

$$\boldsymbol{p}_{k,i} = \begin{pmatrix} \mathbf{t} \\ \mathbf{r} \end{pmatrix} \tag{5.17}$$

$\mathbf{t}$ は位置の 3 次元ベクト ル, $\mathbf{r}$ は姿勢の 3 次元ベクト ルである. ここでは, オイラー角による表現を $\mathbf{r}$ に利用する.

時刻 k における推定結果の代表値 $\boldsymbol{r}_k$ としては, 各パーティクルの重みを 利用した線形結合の値を代表値として用いる.

$$\mathbf{r}_k = \sum_i w_{k,i} \boldsymbol{p}_{k,i} \tag{5.18}$$

5.5.2 状態ベクトルの更新と動作モデル

状態ベクトルを更新する際には, モーションモデル (**eq.** 5.5) に従うが, モーションモデルとして静止モデル (**eq.** 5.6) に加えて等速運動モデルも同時に利用する.

$$\boldsymbol{p}_{k,i} = \begin{cases} \boldsymbol{p}'_{k-1,i} + \boldsymbol{w}_k \\ or \\ \mathbf{p}_{k-1,i} + \mathbf{v}_{k-1} + \boldsymbol{w}_k \end{cases} \tag{5.19}$$

速度ベクトル $\mathbf{v}_k$ は代表値 $\boldsymbol{r}_t$ より計算する.

$$\mathbf{v}_t = \mathbf{r}_t - \mathbf{r}_{t-1} \tag{5.20}$$

モーションモデルを静止モデルのみで構成すると, パーティクルは正規分布のみに従って広がっていく. これは素早く動く対象物を追跡するためには白色ノイズの正規分布の共分散行列の要素の値を大きく設定しなくてはいけないことを意味する. 追跡の信頼性を上げるには, 真値周辺に十分な密度でパーティクルを分布させる必要がある. そのため, パーティクルの数を増やす必要が出てきてしまう.

その一方で, 等速運動モデルのみを使った場合も追跡が不安定になる. 等速運動モデルを利用した場合, 等速に直線で運動する物体の追跡に適している. しかしながら, 一定でない速度で運動する物体の追跡においては, 容易に追跡に失敗してしまう. 例えば, 等速に直線で運動する物体を追跡している最中に物体が急停止した場合, パーティクルの集合は速度を維持して物体よりも先まで移動してしまう. このような状況で追跡を継続させるためには白色ノイズの正規分布の共分散行列の要素の値を大きく設定しなくてはいけない.

静止モデルと等速運動モデルと組み合わせることで, パーティクルの数を増やすことなく両方のモデルの利点を得ることができる. 各パーティクルは予測ステップでどちらかのモーションモデルをランダムに選択する (**eq.** 5.19). どちらのモデルが選択されるかはあらかじめ選択比率を与えておく. モデル選択比率は実験を通して, 静止モデル : 等速運動モデル $= 0.75 : 0.25$ とした.

5.5.3 3 次元点群の仮説評価のための観測モデル

観測モデル $p(z_k|\boldsymbol{x}_k)$ を関数 L を用いて以下のように定義する.

$$p(z_k|\boldsymbol{x}_k = \boldsymbol{p}_i) \propto L(\boldsymbol{p}_i|Z_k) \tag{5.21}$$

パーティクルフィルタにおける各パーティクルの重み w_i は観測モデル $p(z_k|\boldsymbol{x}_k)$ に比例する関数 $L(\boldsymbol{p}_i|Z_k)$ を用いて以下のように定める.

1. 参照 3 次元点群 P_{ref} を i 番目のパーティクルの状態ベクトル $\boldsymbol{p}_i$ が表現する姿勢に移動させる. P_i は移動させれた仮説 3 次元点群とする (**Fig.** 5.15).

$$P_i = \mathrm{transformPointCloud}(P_{\mathrm{ref}}, \boldsymbol{p}_i)$$

2. 仮説 3 次元点群 P_i の全ての点に関して, センサ 3 次元点群 Z_k に対して最近傍点を探索する. 仮説 3 次元点群 P_i の点を p_j としたとき, q_j をセンサ 3 次元点群 Z_k における p_j の最近傍点とする (**Fig.** 5.16).

$$q_j = \mathrm{nearestPoint}(p_j, Z_k) \tag{5.22}$$
$$p_j \in P_i \tag{5.23}$$
$$q_j \in Z_k \tag{5.24}$$

3. 点 p_j に関する近傍点間の類似度 l_j を, 点 p_j と点 q_j の間の距離に従って計算する. 特徴はユークリッド空間と色空間の距離の重ね合わせとして計算する.

$$l_j = l_{\mathrm{euc}}(p_j, q_j) l_{\mathrm{color}}(p_j, q_j) \tag{5.25}$$
$$l_{\mathrm{euc}}(p_j, q_j) = \frac{1}{1 + \alpha \left| p_j - q_j \right|^2} \tag{5.26}$$
$$l_{\mathrm{color}}(p_j, q_j) = \frac{1}{1 + \beta \left| p_{j,hsv} - q_{j,hsv} \right|^2} \tag{5.27}$$

点 p_j と点 q_j の距離がしきい値 thr_e よりも大きい場合, 類似度 l_j を 0 とする. 外れ値とみなされる点を類似度計算で考慮しない事で, 対象物体に属さない点を無視する.

4. 近傍点間の類似度 l_j を足し合わせることで, 関数 L およびパーティクル P_i の重み w_i とする.

$$L(\boldsymbol{p}_i|Z_k) = \sum_j l_j \tag{5.28}$$
$$w_i = \frac{L(\boldsymbol{p}_i|Z_k)}{\sum_i L(\boldsymbol{p}_i|Z_k)} \tag{5.29}$$

仮説が表す姿勢に応じて, 観測可能な点の数は変化する. 観測可能な点数に依存する項を正規化するためには, この点数に従って正規化するのが直接的である. 観測可能な点数は点群の法線ベクトルとカメラ位置を考慮することで計算することができる. しかしながら, このような正規化は特定の姿勢に依存して不安定な挙動を示す. 正規化の結果,

観測可能な点数が少ない姿勢ほど類似度が増加してしまい, 姿勢推定の結果は観測可能な点数が少ない姿勢に偏向してしまう. 例えば, ある姿勢において観測可能な点数が 1 点のみであるとすると, 類似度は大きく増加し, その姿勢周辺にパーティクルが引き寄せられる. そのため, パーティクルの相対的正規化 [113] を利用することでこのような問題を回避する.

$$w'_i = -w_i \tag{5.30}$$
$$w_{min} = \min_i w'_i \tag{5.31}$$
$$w_{max} = \max_i w'_i \tag{5.32}$$
$$\pi'_i = \exp\left(1 - \gamma \frac{w'_i - w_{min}}{w_{max} - w_{min}}\right) \tag{5.33}$$
$$\pi_i = \frac{\pi'_i}{\sum_i \pi'_i} \tag{5.34}$$

正規化された後の重み (**eq.** 5.34) は確率密度関数となるため **eq.** 5.16 を満足する必要がある. また, パラメータ γ は実験から $\gamma = 15$ とする.

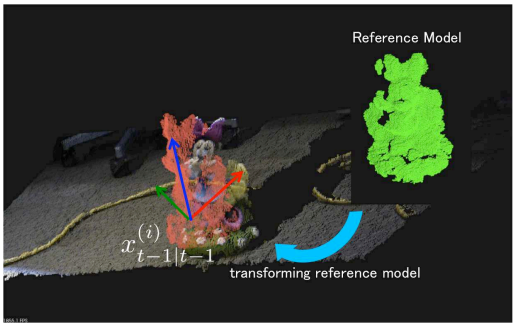


Fig5.15: Transformed Reference Point Cloud. (Red point cloud is the transformed point cloud by hypothesis state vector.)

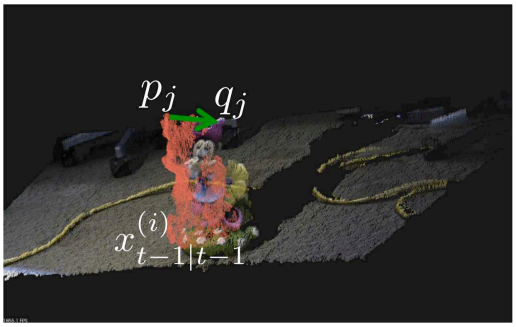


Fig5.16: Nearest Neighbor Pair of Hypothesis and Input Point Cloud

5.6　実時間物体追跡のための高速化

ここまでに述べたパーティクルフィルタを利用した 3 次元点群のトラッキングアルゴリズムの計算量は $O(NM\log(n))$ である. ここで, N は参照 3 次元点群の点数, M はパーティクルの数, n は入力 3 次元点群の数である. $\log(n)$ は最近傍探索の計算コストから与えられる. 3 次元点群をセンサ入力, 参照モデルとして用いることの欠点は, これらの点群に関する近傍探索や移動の計算のコストが高いことである. 実時間で 3 次元点群のトラッキングを実現するた

め, 高い計算コストを下げるためのアルゴリズムの最適化を行う. **Alg.** 8 にその概要を擬似コードとして示す.

Algorithm 8 Pseudo Code of Point Cloud Tracking

$S_t, X_t, B \leftarrow \emptyset$
$k, n \leftarrow 0$
repeat
  Sample an index j with replacement from the distribution given by the weights in S_{t-1}
  $\mathbf{x}_t^n \leftarrow \mathbf{x}_{t-1}^j + \omega$
  if $rand() < R_{motion}$ **then**
    $\mathbf{x}_t^n \leftarrow \mathbf{x}_t^n + \mathbf{v}_{t-1}$
  end if
  $X_t \leftarrow X_t \cup \mathbf{x}_t^n$
  if x_t^n falls into empty bin b **then**
    $k \leftarrow k + 1$
    Check b as non-empty
  end if
  $n \leftarrow n + 1$
until $k < 2 \cap n < M_{\max} \cap n < M(k)$ (**eq.** 5.36)
$P_{t,h} \leftarrow$ empty Point Cloud
for $\mathbf{x}_t^i$ in X_t **do**
  $P_{t,h} \leftarrow P_{t,h} \cup P_{x_t^i}$
end for
$tree \leftarrow$ OcTree(cropByBoundingBox($P_t, P_{t,h}$))
for $\mathbf{x}_t^i$ in X_t **do**
  $w_i \leftarrow$ computeSimilarity($P_{x_t^i}, tree$)
  $S_t \leftarrow S_t \cup \{\mathbf{x}_t^i, w_i\}$
end for

5.6.1　複数 CPU コアを利用した並列計算

パーティクルフィルタを利用する大きな利点の一つは, 並列化の容易性にある. 各仮説 p_i に関する重み w_i の計算は互いに独立であるため, 複数 CPU コアを利用して並列に計算可能である. 並列スレッド数を N_{thread} とすると, N_{thread} が増えるに従って計算時間は減少する. 並列計算による高速化は, 精度に影響を与えないという利点が大きい. その一方で, 計算機資源を

多く利用するため, ロボットシステムに統合する際は利用できる計算機資源を考慮すると共に, 分散処理によってシステム全体で利用できる計算機を増やすことが重要となる.

5.6.2 近似最近傍探索の利用

最も計算コストが高い要素は関数 $L(\boldsymbol{p}_i|Z_k)$ における最近傍点の探索である (**eq.** 5.22). 厳密な最近傍点探索はコストが高いため, 8 分木による近似最近傍点探索を利用する.

5.6.3 低解像度化による点数の削減

参照 3 次元点群の点数 N を削減するため, 立方体格子を利用した点群の低解像度化を利用する. このときの格子サイズを d_{voxel} とする. d_{voxel} を大きくすると計算コストを大きく下げることができるが, 同時に精度も低下させてしまう. そのため, 適した voxel gird の解像度パラメータ d_{voxel} を設定することが必要である.

5.6.4 Adaptive Particle Filtering によるパーティクル数の削減

パーティクルの数 M はすでに述べたように, 計算コストの重要な項を占める. しかしながら少ないパーティクルでは十分な姿勢推定結果の精度, トラッキングの追従性の両方が難しくなる. 一方で, パーティクルの数 M は環境やトラッキングの状況に応じて減らすことができる. パーティクルの数 M を動的に適した数に決定するため, KLD-sampling[114] を用いる. KLD-sampling のアイデアは, 状態空間においてパーティクルが狭い空間に集中しているときは少ないパーティクルを利用し, 逆にパーティクルが広い空間に広がっている場合は不確実性が高いとみなし, 多いパーティクルを利用するというものである. KLD-sampling においてパーティクル数 M は以下のように決定する:

$$M(k) = \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \tag{5.35}$$

$$\doteq \frac{k-1}{2\epsilon}\left\{1 - \frac{2}{9\,(k-1)} + \sqrt{\frac{2}{9\,(k-1)}}z_{1-\delta}\right\}^3 \tag{5.36}$$

ここで, $z_{1-\delta}$ は $N(0,1)$ の正規分布における上部の $1-\delta$ 分位点である.

5.6.5 センサ点群の領域選択による高速化

計算量 $O(NM\log(n))$ の $\log(n)$ は octree や kdtree の探索コストである. n はセンサ点群の点数であったが, これをセンサ点群において 3 次元的な注視領域を適用することで, n を減

少させる. パーティクルフィルタによる物体追跡では, 時刻 k の計算に先立って, 時刻 $k-1$ のパーティクルの分布が既知である. パーティクルは十分に対象物の真値周辺に分布しているという仮定を置くことで注視領域を設定する.

1. 時刻 $k-1$ の仮説点群 P_i を全てのパーティクル i に関して加算する (**Fig.** 5.17).
2. 加算された点群のバウンディングボックスを計算する (**Fig.** 5.18 左).
3. 時刻 k のセンサ点群を時刻 $k-1$ のパーティクルから算出したバウンディングボックスを注視領域として, 内部にある点群のみを利用する (**Fig.** 5.18 右).

このようにして切り出された入力点群をセンサ出力の代わりに octree や kdtree で利用する. センサ点群の注視領域設定による高速化は対象物が高速に動きパーティクルの分布から完全に対象物が外れてしまった場合, 注視領域となるバウンディングボックスの外側に対象物が移動してしまうため, トラッカは対象物を見失ってしまう. しかしながら, そのような場合, 注視領域の設定を行わなくてもトラッカは対象物の追跡に失敗している可能性が高いと考えられる. そのため, このような仮定に従ったセンサ点群の領域選択による高速化はパーティクルフィルタの精度に影響を与えにくいと言える.

Fig5.17: Building Hypothesis Point Cloud

5.7　3 次元点群追跡の継続的認識評価実験

5.7.1　3 次元点群追跡の速度評価実験

計算機としては, 2.8 GHz Intel Core i7 CPU の 4 物理コアの計算機を用いた. 並列計算の実装には OpenMP を用いた. センサとしては Asus Xtion Pro を利用した. **Fig.** 5.19 ではジョッキの 3 次元点群を追跡している. これはシンプルな形状を簡単な環境でトラッキングし

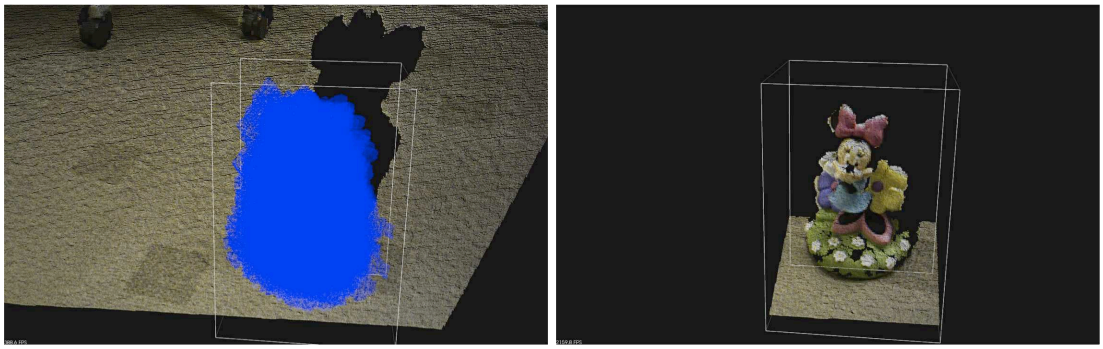


Fig5.18: Left) Bounding Box of Hypothesis Point Cloud Right) Cropping Input Point Cloud using Bounding Box

た様子である．この実験におけるトラッキングのうち，パーティクルフィルタの計算速度は約 26fps であるが, 立方体格子による点群の低解像度化と合わせると 15fps 程度である．**Fig.** 5.20 では **Fig.** 5.19 と同じ環境で, より複雑な形状である人形をトラッキングしている．この実験におけるパーティクルフィルタの計算速度は約 25fps であるが, 立方体格子による点群の低解像度化と合わせると 15fps 程度である．これは **Fig.** 5.19 とほとんど計算速度は変化していない．**Fig.** 5.21 ではヘッドフォンをトラッキングしており , より煩雑な環境でのトラッキング実験となっている．この実験におけるパーティクルフィルタの計算速度は約 15fps であるが, 立方体格子による点群の低解像度化と合わせると 10fps 程度である．これは **Fig.** 5.20, **Fig.** 5.19 の実験と比べると低速なものになっている．その理由は環境が複雑であるため, パーティクル数を KLD-Sampling によって削減することが出来ず, 計算コストが高くなってしまったことである．これらの実験において, 青い点群はトラッキング結果を示しており , 画像と同じ色が付いている点群はセンサ入力点群である．速度の計測結果は **Table** 5.2 にまとまた．

Table5.2: Experimental Result: Speed of PointCloud Tracking with Different Targets

Target	Particle Filtering	Downsampling	Tracking Speed
Beer	26fps	40 fps	14.8 fps
Doll	26 fps	42 fps	14.8 fps
Headphones	15.2 fps	36.5 fps	10.0 fps

Fig5.19: Tracking of Beer Mug

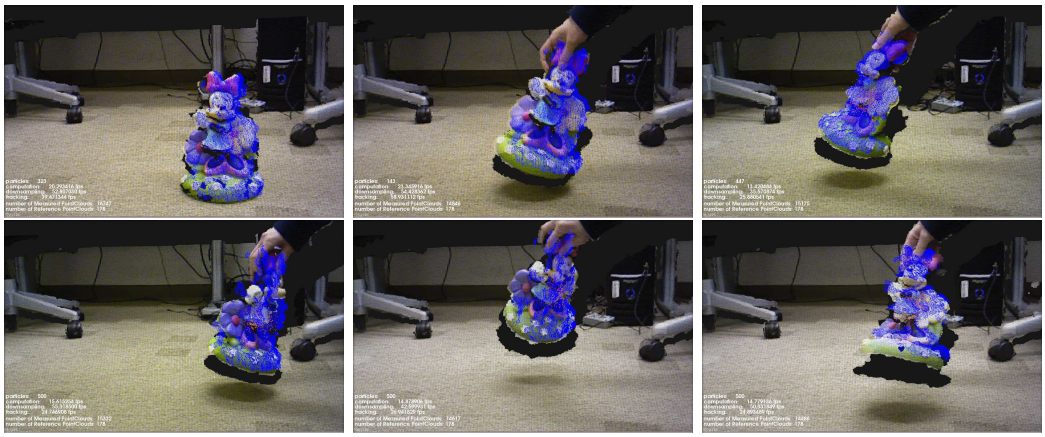


Fig5.20: Tracking of Doll

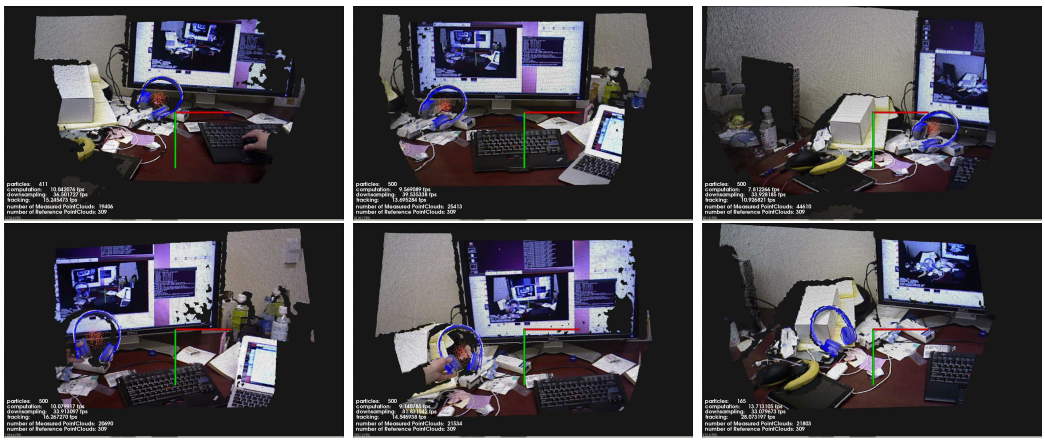


Fig5.21: Tracking of Headphones in Cluttered Scene

Table5.3: Comparison of proposed tracker. The alues are 6-DOF pose RMS Errors between the tracker and the ground truth pose.

Object	**RMS Errors**					
	X(m)	Y(m)	Z(m)	Roll(deg)	Pitch(deg)	Yaw(deg)
Gallon	0.024	0.026	0.011	5.15	8.12	5.08
Cookie Box	0.018	0.020	0.014	3.10	3.23	4.85
Spray	0.103	0.089	0.039	10.82	4.01	17.80
Juice Box	0.033	0.028	0.079	12.31	9.97	10.14
Drill	0.015	0.020	0.020	4.37	5.57	3.37

5.7.2　3 次元点群追跡の精度評価

点群追跡による精度の評価実験を行った．実験では真値を測定するためにはモーションキャプチャシステムである optitrack[*1]を利用した．**Table** 5.3 に精度の評価結果を示す．

(a) Cup　(b) Cookie box　(c) Gallon　(d) Juice Box　(e) Drill　(f) Spray

Fig5.22: Objects to Evaluate Tracking

また, 他のアルゴリズムとの比較のため, 代表的な 2 次元画像によるトラッキング手法として STRUCK[115], CMT[116] の 2 つのアルゴリズムとの比較を行ったものを **Fig.** 5.23 に示す．本章で提案した 3 次元点群による点群追跡手法が 2 次元画像によるトラッキング手法よりも精度良い挙動を示している．

5.7.3　静的な環境下における高速化パラメータの精度への影響評価実験

本小節では, 高速化のための導入したパラメータを変化させることで, 計算速度および精度にどの程度影響があるかを評価するための実験を行う．本小節での評価を用いて, 第 7 章にお

[*1] http://www.optitrack.com/

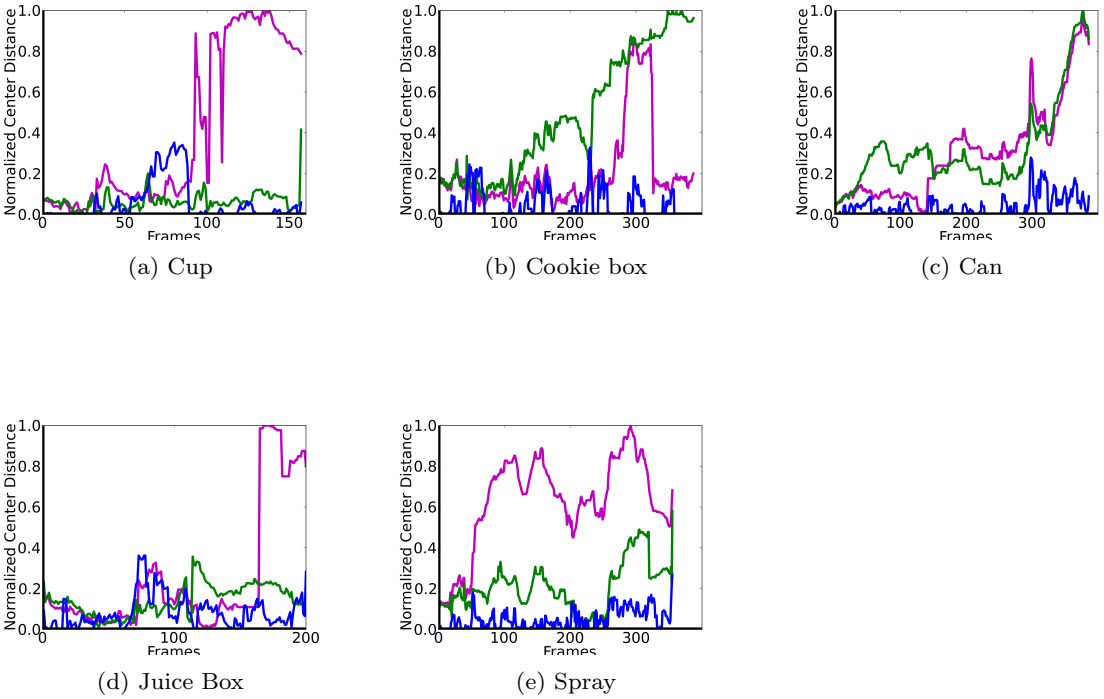


Fig5.23: Normalized center error Euclidean distance between the trackers and the ground truth. Each graph is normalized by the maximum and the lower value represents the best tracker.

いて品質-時間テーブルを構築する.

実験には第 3 章で述べた災害対応タスクの *Valve* タスクにおけるバルブを利用した (**Fig.** 5.24). また, トラッキング行う際の環境は静的なものとしている. 本実験で変化させるパラメータは以下の 3 つである.

d_{voxel}　立方体格子における格子サイズ (第 5.6.3 節)

N_{thread}　並列スレッド数 (第 5.6.1 節)

$N_{\mathrm{particles}}$　パーティクル数 (第 5.6.4 節)

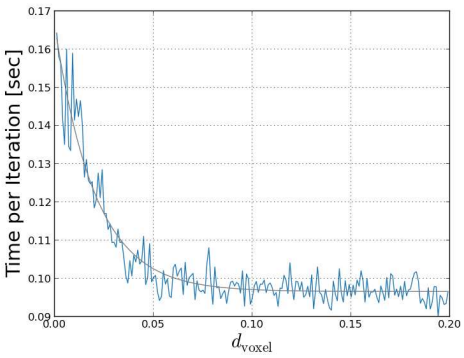
ここで, パーティクル数 $N_{\mathrm{particles}}$ は第 5.6.4 節で述べたように, Adaptive Particle Filtering によって動的にパーティクル数を決定するのが望ましいが, KLD-sampling によるパーティクル数の決定は実験環境に大きな影響を受けるため, 本実験では直接パーティクル数を指定した. それぞれのパラメータを変化させた時の計算時間, および位置と姿勢の精度に関して **Fig.**

Fig5.24: Experimental Setup of Valve Tracking for Evaluating Time and Accuracy. Blue Points Shows Result of Tracking.

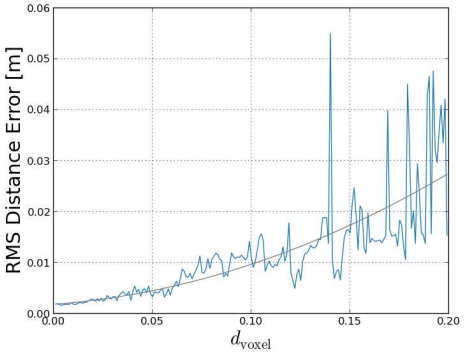
5.25, **Fig.** 5.26, **Fig.** 5.27 に示す. 並列スレッド数 N_{thread} に関しては, 静的な環境では精度に影響を与えないため, 計算時間のみに関して実験結果を示した. 立方体格子における格子サイズの評価 (**Fig.** 5.25) に関しては計測のノイズが大きかったため, 計算時間, 位置と姿勢の精度それぞれに関して灰色で指数関数, 二次関数の最小二乗法によるフィッティング結果を示している.

それぞれのパラメータ変化の影響に関して考察を行う. 立方体格子における格子サイズ d_{voxel} は格子サイズが大きくなるほど計算時間は短くなるが, 精度は低下してしまう. 特に姿勢に関する誤差は影響が大きい. これは, 格子サイズが大きくなることで小さな 3 次元特徴が観測できなくなることが原因だと考えられる. パーティクル数 $N_{\mathrm{particles}}$ は値が小さい時 ($N_{\mathrm{particles}} < 500$), 精度に大きな影響を与えている. パーティクルが一定数以上であれば誤差は収束しているが, この収束値はパーティクルに与える白色ノイズの大きさに依存している. また, 本実験では静止した環境を用いているため, パーティクル数が増えることによる追従性の向上といった影響は見られていない. 並列スレッド数 N_{thread} は静止環境においては精度に影響を与えないため, 計算機資源に余裕があるかぎりは大きな値を利用することが望ましい. 並列スレッド数が一定以上 ($N_{\mathrm{thread}} > 20$) では高速化の効果が見られにくくなっている原因

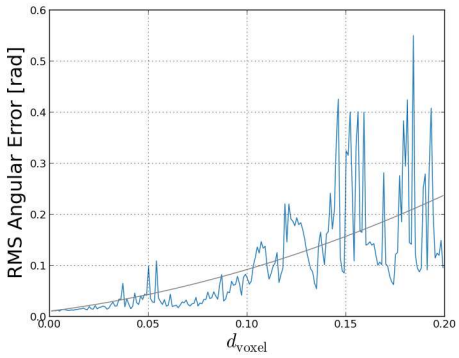
としては, 各スレッドが担う計算負荷が十分小さくなったためであると考えられる.



(a) Time to Track Pointcloud with 1 Iteration with Different d_{voxel}. Gray Line is Fitted Exponential Function $0.072\exp(-49d_{\mathrm{voxel}}-0.037)+0.097$



(b) Effect of Down-Sampling with d_{voxel} Voxel Grid to Translation Accuracy. Gray Line is Fitted Quadraric Function $0.50d^2_{\mathrm{voxel}}+0.50d_{\mathrm{voxel}}+0.0019$



(c) Effect of Down-Sampling with d_{voxel} Voxel Grid to Rotation Accuracy. Gray Line is Fitted Quadraric Function $3.2d^2_{\mathrm{voxel}}+0.50d_{\mathrm{voxel}}+0.011$

Fig5.25: Effect of Down-Sampling with d_{voxel} Voxel Grid

5.8　おわりに

本章では, 3 次元点群を利用した物体の位置姿勢の 6 自由度追跡手法について示した. 物体追跡手法にはパーティクルフィルタを用い, そのための観測モデルを定義した. 仮説の観測モデルは 3 次元点群の座標や色特徴を利用したものであった. 評価値計算は参照点群モデルとセンサ入力点群との間の最近傍点ペアでの類似度を元としたものであった. 計算量を削減するた

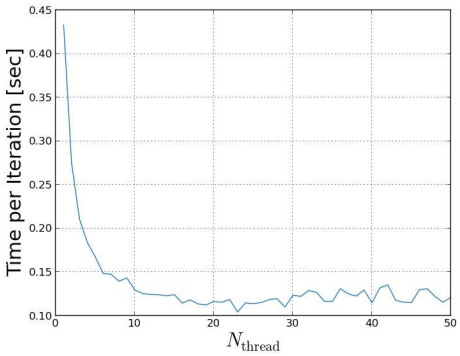
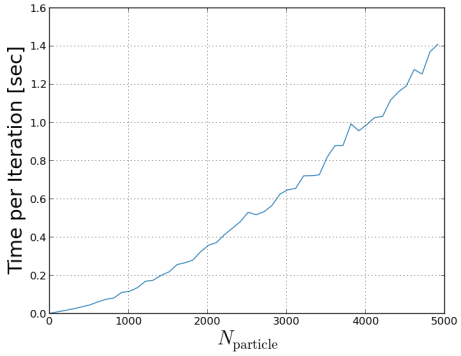
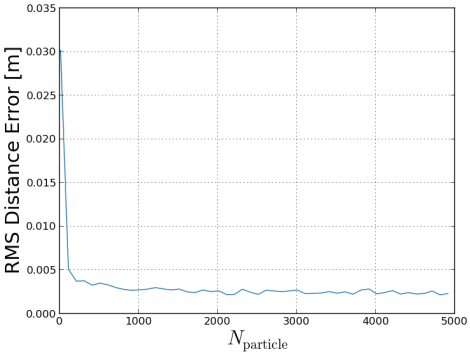


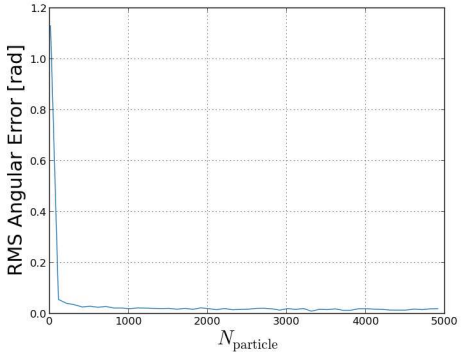
Fig5.26: Time to Track Pointcloud with 1 Iteration with Different Number of Parallel Thread N_{thread}



(a) Time to Track Pointcloud with 1 Iteration with Different Number of Particles $N_{\text{particles}}$



(b) Effect of the Number of Particles ($N_{\text{particles}}$) to Translation Accuracy



(c) Effect of the Number of Particles ($N_{\text{particles}}$) to Rotation Accuracy

Fig5.27: Effect of the Number of Particles ($N_{\text{particles}}$)

め, 複数の最適化手法を導入した. 精度に影響を与えない高速化としては, 複数の CPU コアを利用した並列計算を導入した. 点群数を削減するために立方体格子を利用して点群解像度を削減することは実時間処理を実現するために非常に大きな効果があった. しかしながら, 点群解像度を粗くすると姿勢推定の精度が下がってしまうため, 点群解像度は注意深く決定する必要がある. さらに, 仮説の分布に基づいて入力点群に対して 3 次元的な注視領域を設定することで点群数を削減することが可能であった.

提案した 3 次元点群追跡の精度や速度に関して, 評価実験を行った. また, 高速化の中で精度や速度に影響を与えるものに関して, その影響を実験を通じて評価した. これは第 2.3.3 節で述べた品質-時間テーブル構築のために利用される.

点群に関する特徴のみを使った姿勢追跡では, 3 次元的な特徴に乏しい物体に関しては追跡精度が落ちてしまう. そのような物体に関しては 2 次元画像特徴量を同時に利用することで追跡精度を向上させることが可能だと考えられる.

本章で述べた 3 次元点群追跡のソフトウェアは 3 次元点群処理ライブラリ Point Cloud Library (PCL)[*2]の一部としてオープンソースで公開されている.

[*2] http://pointclouds.org/

第 6 章

環境およびロボットの状態監視のための常時稼働型ソフトウェア

6.1 はじめに

本章では, 環境やロボットに関してタスク実行時に状態を監視することで, 動的な環境の変化や予期しない環境との接触に対応するための常時稼働型のソフトウェア (第 2.3.4 節) について述べる (**Fig.** 6.1).

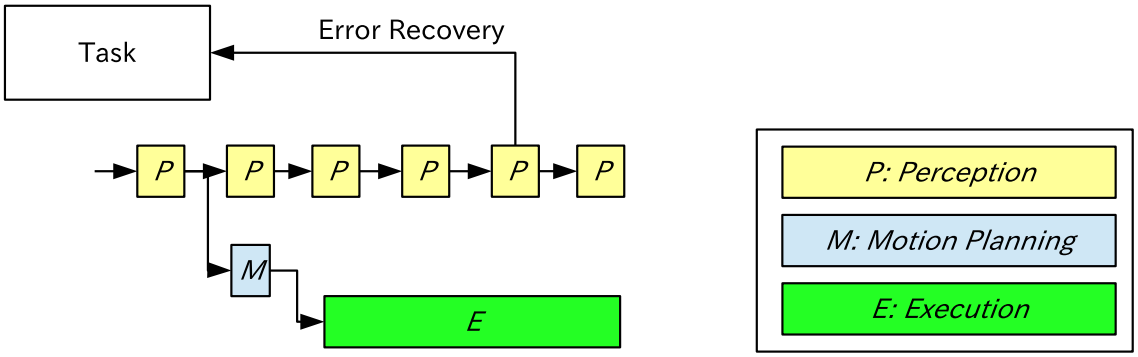


Fig6.1: Overview of Monitoring of Environment and Robot during Execution

環境およびロボットの状態を常時監視することによる利点は以下のようなものである.

- 環境が動的に変化する場合, その変化量をエラーとして解消することで動的な環境への即時性の向上が見込める点
- 認識処理やセンサの視覚の制限, さらには動作計画の不備などで環境との予期しない衝突やそれに伴う転倒が発生する可能性がある時にロボットの動作を緊急停止させることで転倒を回避することが可能である点

その一方で, タスク実行中の監視のための計算処理, 特に環境を監視するための認識処理は計算機資源を必要とする. しかし, 特にロボットミドルウェアによる複数の計算機を利用した分散処理, および計算機の CPU コア数の増大によって, タスク遂行のために利用できる CPU コア数には余裕が増えつつある. そのため, 動作計画処理が終了した後でも引き続き認識処理を継続することは難しくなくなってきている. また, このような認識処理を継続していくためには物体追跡による認識機能が重要である. このために本章では第 5 章において述べた 3 次元点群追跡による物体認識器を利用する.

状態を監視することでエラーを検知した場合はロボットを停止させることはもちろん, 動作計画器に対して変化量を入力することでビジュアルフィードバックによる動く対象物への追従動作であったり, 上位のタスクプランニング機能にフィードバックすることで大域的な行動の修正, 遠隔操縦インタフェイスを通じてオペレータへの注意喚起といった応用を行うことができる.

タスクプランニング機能は第 2.6 節において遠隔操縦インタフェイスと共に上位に位置し, タスク遂行のための意思決定を行う. タスクプランニングを利用するためにはシンボリックな記述によってタスクおよび環境を表現する必要がある. これはまさに古典的な AI として知られるような推論システムである. シンボリックな記述を利用することで, 実行時にタスクをロボットが想定しているように実行できているかをセンサ入力から確認することが可能である.

本章ではこの確認機能を第 5 章で述べた認識機能を用いることで常時確認可能なように拡張し, システムの環境の変化への即応性を向上させるシステム構成法を古典的 AI であるタスクプランニングと連携する形で述べる.

6.2　関連研究

ロボットのタスク管理, タスクプランニングについては今まで多くの議論がなされている. ロボットのタスクプランニングにおいては, タスクを状態 (state) とその状態間の移動 (operator) によって記述するシンボリックなプランニングがよく用いられる. Fikes らは STRIPS[2] と呼ばれるロボットのタスク記述言語を提案した. STRIPS におけるオペレータは動作 (action), それを行うために必要な前提条件 (precondition), その動作による副作用 (effect) の三つを一つにまとめたものが用いられる. 小倉らはこの STRIPS による表現を用いて高度なヒューマノイドによるタスクマネージメントシステムを構築した [36]. しかし小倉らのシステムにおいては失敗からの復帰については考慮されていなかった. Okada らは小倉らの

システムを拡張し, 画像処理による認識機能を統合することで行動の前後で想定しているシンボリックな記述と現在の環境から推定される記述との差分を検知することで失敗復帰可能なシステムを構築した [38]. しかしながら行動の成否確認を行うのは行動の前後のみに限られていた. 本章ではこれらの研究を発展させ, 行動の成否確認を行動の実行中においても行うシステムを構成する.

近年では PDDL[18] と呼ばれる STRIPS を拡張した自動計画言語を利用した行動計画の研究が行われている. PDDL は Problem Domain Description Language を意味し, IPC(International Planning Competition) のために開発された. PDDL にはいくつかのバージョンが存在し, PDDL 1.1[18], PDDL2.1[117], PDDL2.2[118], PDDL3.0[119] がある. STRIPS への拡張として, 例えば, 型情報を付与するための typing, 状態のとり得る値を真偽の 2 値から, 整数に拡張するための fluents などである. ロボットのプランニングに関しても問題記述に PDDL および問題解決に PDDL のソルバーが利用されている [5]. また, PRS[19, 20] と呼ばれる推論システムが開発されている. PRS は Procedural Reasoning System を意味する. 本論文では, STRIPS を拡張した自動計画言語 PDDL をロボットの行動記述に用いる.

その一方で, ロボットの行動シーケンスを有限要素機械による記述を提案しているものも多い. 金広らは StateNet[120] を提案し, 開発者が陽に失敗回復行動を実装せずに障害回復を可能とするシステムを提案した. しかし, StateNet による行動生成では物体操作を含むような行動を実現することは難しい. また, 反射的なモジュール群に基づき, ロボットの行動を決定する手法として, サブサンプションアーキテクチャが提案されている [121]. サブサンプションアーキテクチャでは単純なロボットの行動システムを構築することが効果的に行えるが, ヒューマノイドのような複雑な行動システムの記述は難しい.

6.3 シンボリックなタスク問題記述を利用した行動計画

本小節では, 代表的な自動計画言語であり, 本論文でも利用する STRIPS について [122] を参考に説明する.

STRIPS は STanford Research Institute Problem Solver の略である. STRIPS に限らず, タスクプランニングは一階述語論理 [122] かその部分である形式言語による記述を用いる. プランニングにおける状態と目標は文の集合となり, 行為は条件と結果の論理的な記述となる. 以下にタスク問題記述の主要な構成要素を述べる.

1. 状態の表現

   状態の表現は関数を含まない, 真であるリテラルの条件から構成される. 状態の表現には閉世界仮説が用いられるため, 言及されていない条件は常に偽である.

2. 目標状態の表現

   目標状態に達したと判断するために条件文によって表現される状態の部分集合 g である. ある状態 S が g に含まれるすべての条件を見たせばその状態 S が目標に達したとされる.

3. オペレータの表現

   行為記述の集合である.

STRIPS における行為記述は以下の 3 つの要素から構成される.

- 行為名およびパラメータ

  行為を表現するための名前, およびパラメータである.

- 前提条件

  行為を適用するために見たしてなくてはいけない条件である. この条件に含まれる変数はパラメータに含まれている必要がある.

- 副作用

  その行為を適用することで変化する状態を記述する条件である. この行為を適用した後に真となるべき条件群を P, 偽となるべき条件群を Q とすると, 状態 S に行為を適用することは, 状態 S に P を加え, Q を削除するということである.

これらに加え, 本章では行動中の成否判定を行うための条件を追加する.

- on 条件

  on 条件は実行中に特に常に成立すべき条件である. これはシンボリックなプランニングにおいては必要ではないが, 後に行為実行中の監視機能のために必要となる. 行為によって変化しない条件群は先の副作用に記述されていないもの全てとなる. 明示的に監視すべき条件として記述することで計算量が少なくなるのはもちろんだが, ロボットが関係する条件を限定できるため, 動作の安定性が向上すると考えられる. というのも, ロボットの行動中は対象となっている物体以外については視覚の隠れが生じたりすることで認識結果が不安定になるためである.

STRIPS によるプランニングでは, 与えられた初期状態 S_0 から g をみたす状態へと行為の副作用によって遷移していくことが目的となり, その軌道である行為列が結果として得られる. 以下に STRIPS によるハノイの塔の記述例を示す. 記述には S 式を用いた.

```
(defobjects peg1 peg2 peg3 disk1 disk2 disk2)
(defpredicate (on obj1 obj2))
(defpredicate (clear? obj))
(defpredicate (bigger? obj1 obj2)) ;; true when obj1 > obj2
(defaction move (target from to)
  :precondition (and (on target from)
                     (bigger? to target)
                     (clear? to))
  :effect (and (not (on target from))
               (not (clear? to))
               (on from to)))
(definitial (on disk3 peg1)
            (on disk2 disk3)
            (on disk1 disk2)
            (clear? disk1)
            (clear? peg2)
            (clear? peg3)
            ;; constant states
            (bigger disk2 disk1)
            (bigger disk3 disk2)
            (bigger disk3 disk1)
            (bigger disk1 peg1)
            (bigger disk2 peg1)
            (bigger disk3 peg1)
            (bigger disk1 peg2)
            (bigger disk2 peg2)
            (bigger disk3 peg2)
            (bigger disk1 peg3)
            (bigger disk2 peg3)
            (bigger disk3 peg3))
(defgoal (and (on disk3 peg3)
              (on disk2 disk3)
              (on disk1 disk2)))
```

この問題記述を元に, 行動列を計画すると, 以下のようになる.

```
1. (MOVE DISK1 DISK2 PEG3)
2. (MOVE DISK2 DISK3 PEG2)
3. (MOVE DISK1 PEG3 DISK2)
4. (MOVE DISK3 PEG1 PEG3)
5. (MOVE DISK1 DISK2 PEG1)
6. (MOVE DISK2 PEG2 DISK3)
7. (MOVE DISK1 PEG1 DISK2)
```

6.4　発生するエラーの分類

本節では, ロボットシステムが検知するエラーに関してその定義を述べ, それを 2 つに分類する.

本論文では, プランを生成する前に仮定していた記述とプランを生成した後に想定している記述が, センサから推定された記述と異なっていることをエラーと定義する. エラーは以下のような 2 つのレベルに分類可能である [123]. この分類は, それぞれのエラーを解決するためにロボットがどのような戦略を取るべきであるかという観点から分類している.

1. 大域的なエラー

   大域的なエラーは, 解消するためにロボットが新たな行動を取る必要があるものである. 新たな行動とは, 例えば物体を下に落としてしまった場合は床面から物体を拾い直さなくてはいけない, といったものである.
2. 局所的なエラー

   局所的なエラーは, 解消するためにロボットが行動を変化させるというよりは, 目標値を微小に修正することで解消することが可能なものである.

システムの高次のレベルではシンボリックな記述, 下位の動作計画などでは幾何的な記述が利用される (**第** 2.2 節). それぞれの記述に対応するように, エラーを以下のように分類する.

1. シンボリックな記述レベルでのエラー

   シンボリックな記述におけるプラン時に想定していたものと, 実行時に推定されたものとの差分である.
2. 幾何的な記述レベルでのエラー

   動作軌道生成のための動作計画時における幾何的な環境記述と, 動作実行中の幾何的な環境記述との差分.

大域的なエラーと局所的なエラーをシンボリックな記述レベルでのエラーと幾何的な記述レベルでのエラーにそれぞれ対応させる. その理由は, それぞれのエラーに対する復帰の手法による.

1. シンボリックなレベルでのエラー復帰 (Global Error Recovery)

シンボリックなレベルでのエラーから復帰するためには, タスクプランニングによる再計画を行い, 新たに行動列を再生成する必要がある. タスクプランナは記号接地によって得られた現在のシンボリックな環境記述に基づき, 目標状態を満たす行動列を生成する. このようにエラー解消に新たな行動列を必要とするため, シンボリックなレベルのエラーは大域的エラーとみなすことができる.

2. 幾何的なレベルでのエラー復帰 (Local Error Recovery)
   例として, 物体認識を考えてみる. 物体が移動されれば, それはエラーとして検知され, 動作計画によって生成される軌道は適時修正される. つまりビジュアルフィードバックによって幾何的なレベルでのエラーは解決される. このようにエラー解消に新たな行動列を必要としないため, 幾何的なレベルのエラーは局所的エラーとみなすことができる.

6.5 非同期監視モジュールによる状態監視

本節では, ロボットの動作実行とは非同期に駆動される監視モジュールによる状態監視について述べる.

6.5.1 状態監視モジュールの分類と機能

監視モジュールは以下の 2 種類に分類可能である.

1. シンボリックな記述監視のためのタスク監視モジュール
   シンボリックな状態記述を常に監視することで, シンボリックなエラーが発生した際にロボットを停止させ, タスクプランニングを再実行させる. シンボリックな記述監視のための監視モジュールはユニークな名前を持つシンボルを介して認識結果を反映する. **Fig.** 6.2 にシステム内でのタスク監視モジュールの位置づけを示す.
   - Sensor Capturing System
     シンボル状態層の入力となるロボットに搭載されたセンサ値の更新を行う.
   - Symbol Properties Update Modules
     シンボルはジオメトリックな情報を保持するために属性を持つ. ここではその属性値の更新をセンサ層の出力を入力として行う. 属性として最も利用されるのは, そのシンボルが表現する物体の座標値であり, 視覚ベースな物体認識器等はこの階層に位置される.

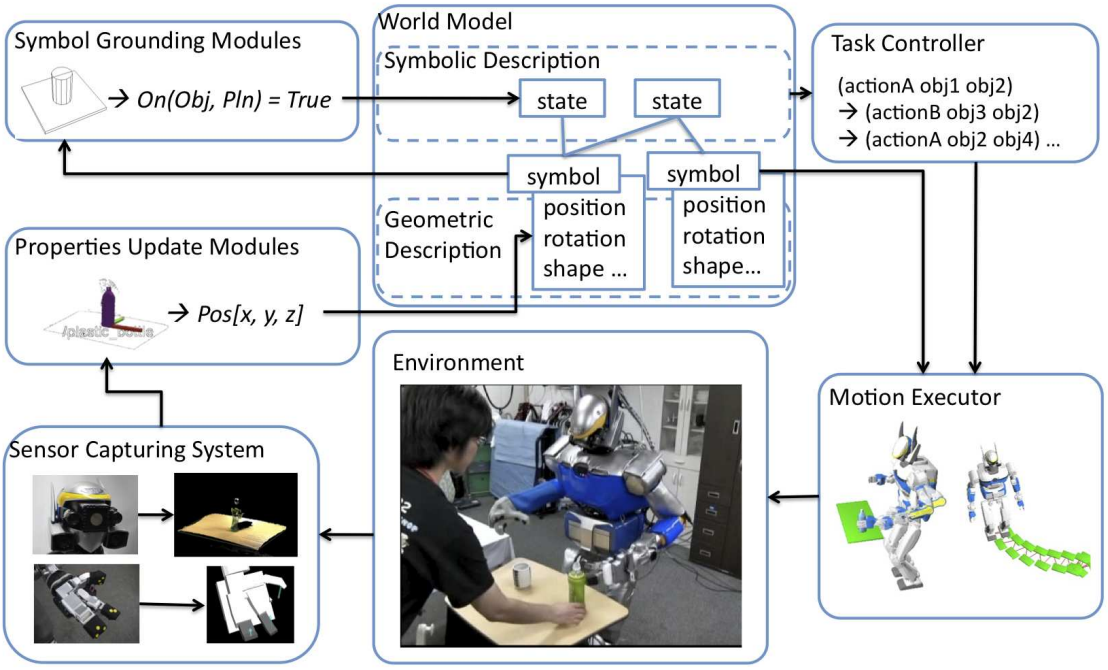


Fig6.2: Symbolic Description Monitor and Symbols

- Symbol Grounding Modules

  述語が引数に取るシンボルの属性値にしたがって, その述語の真偽値の推定をおこなう. この推定によって, ジオメトリックな表現からシンボリックな表現へと抽象化する.

- Motion Executor

  動作生成器の入力としては, 行為およびその引数であるシンボルが渡される. ここで, それらは座標値ではないことに注意されたい. シンボルを入力とすることで, シンボル属性の更新を適時動作に反映させることが可能となる.

- Task Controller

  タスク実行制御器はシステム全体の振る舞いを決定する調停器として動作する. タスク制御器は作業計画を行うため, タスクプランナをその内部に持つ. タスク実行制御器が担う主な役割は以下の 3 つである.

  – タスクプランニングによって行動列を生成する

  – Motion Executor へ目標値を指示する

  – シンボリックなレベルの記述を監視する

2. シンボリックな記述に依存せず動作の種類に応じて稼働する動作監視モジュール
   動作実行中に常時稼働する監視モジュールは, 上位の意思決定がタスクプランニングではなく遠隔操縦インタフェイスが利用された場合でも稼動している. これはタスクプランニングと連動するというよりは, 動作計画器と連動するものである. 動作計画器が想定していた状況と, センサ値から推定されるロボットと環境の関係が異なった場合に, 動作監視モジュールはタスクプランニングを再度実行もしくは遠隔操縦インタフェイスにオペレータの指示を仰ぐためのユーザインタフェイスを提示する. このような動作監視モジュールについては, 後の第 6.5.3 節, 第 6.5.4 節において詳しく述べる.

6.5.2 背景差分による環境監視機能の計算負荷低減

本章ではすでに述べたように, 環境の状態を監視するために第 5 章において述べた 3 次元点群追跡による物体認識器を利用する. 利用可能な計算機性能の向上によって, 常時複数の物体認識器を実行することが可能となってきている. しかしながら, 追跡対象となる対象物が増加することで計算負荷が増大したり, 通信制約のため複数計算機の利用が難しい状況が考えられる. 本小節では物体追跡による環境監視機能の計算負荷低減のため, 3 次元点群の背景差分を利用する.

対象物の位置姿勢を常時追跡する場合, 対象物が静止している場合は追跡のための計算を行う必要はない. 特に環境に固定されたカメラを用いた対象物のトラッキングでは, 背景差分画像を利用することで動的な対象物を追跡する手法が用いられている [124, 125]. 画像を用いた背景差分では, カメラがロボットに搭載されている場合はロボットの動作によって画像に差分が生じてしまう. そのため, 本研究では画像ではなく 3 次元点群を利用してフレーム間で差分を取ることにより, 環境の変化を検出する. 3 次元点群をそのまま利用して差分計算を行うことは難しいため, 8 分木を利用した 3 次元点群の背景差分法 [126] を利用する.

Fig. 6.3 に追跡器の挙動の概要を示す. トラッカは 2 つの状態を持つ.

1. Tracking 状態
   この状態では, 対象物に動きがあるとみなし, パーティクルフィルタによる追跡処理を行う.
2. Stable 状態
   この状態では, 対象物に動きがないとし, 物体追跡のための計算処理を行わない. 認識処

理結果として前フレームの追跡結果を出力する.

これらの状態の遷移条件は以下のようにした.

1. Tracking 状態から Stable 状態への遷移

   Tracking 状態から Stable 状態への遷移が発生する状況は, 追跡対象物が移動していない場合である. そのため, 一定時間対象物の移動速度が一定値 thr_{vel} 以下であるかによって判定する (**eq.** 6.1).

$$\|\boldsymbol{v}_i\| < thr_{\mathrm{vel}} \tag{6.1}$$

   ただし, 添字 i は直近の数フレームを意味する.

2. Stable 状態から Tracking 状態への遷移

   Tracking 状態から Stable 状態への遷移が発生する状況は, 追跡対象物が移動を開始した場合である. しかしながら, Stable 状態においては追跡処理をしていないため, 対象物が移動し始める時刻を知ることができない. そのため, 対象物近傍の点群に差分が生じたかを遷移条件として利用する.

$$\|\boldsymbol{p}_j - \boldsymbol{x}_t\| < thr_{\mathrm{dist}} \tag{6.2}$$

$$^{\exists}\boldsymbol{p}_j \in P_t \tag{6.3}$$

   ただし, 時刻 t および $t+1$ の間の差分 3 次元点群を P_t とし, 対象物の座標を $\boldsymbol{x}_t$ とする.

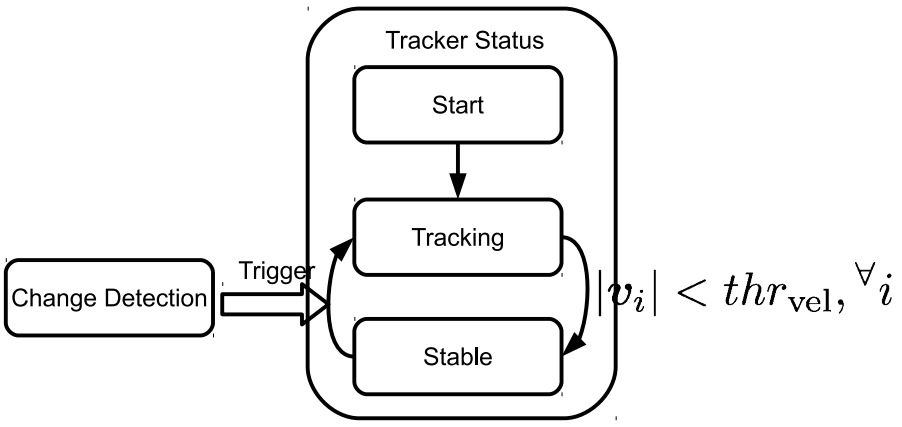
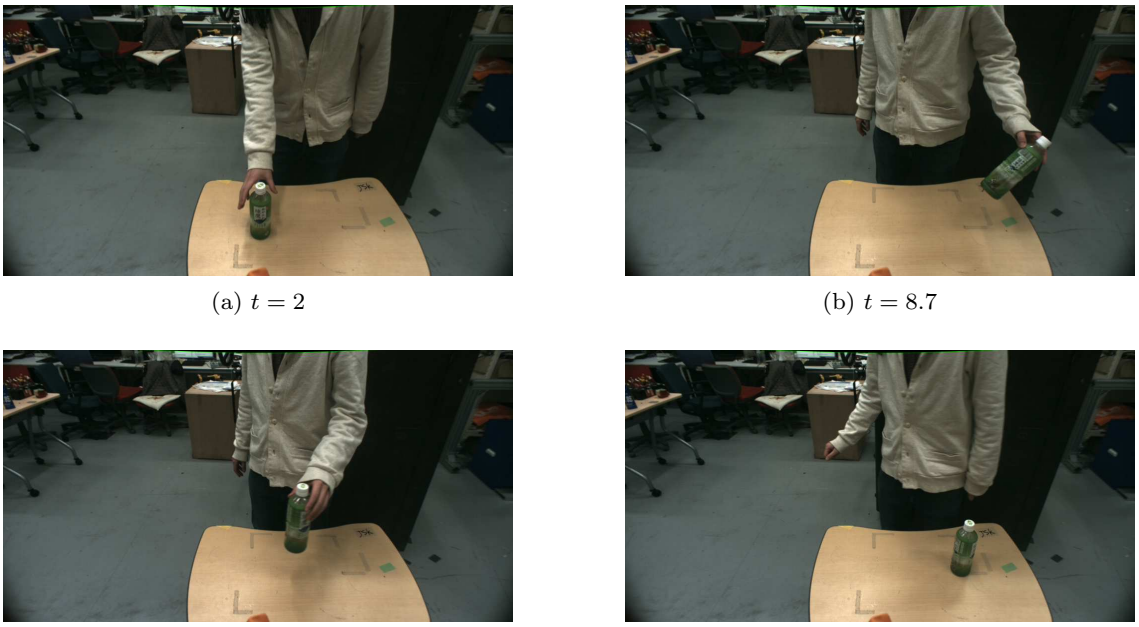


Fig6.3: Overview of Tracker Status to Skip Static Object Tracking

Fig. 6.4, **Fig.** 6.5, **Fig.** 6.6, **Fig.** 6.7 に背景差分による環境監視機能の計算負荷低減の実験結果を示す. この実験では $t = 2.5$ から $t = 17$ まで人が追跡対象物であるペットボトルを

把持している．**Fig.** 6.5 において, 赤で示されている点群が前フレームとの変化が発生している領域, 水色で示されている点群が追跡結果である．**Fig.** 6.7 において, 計算時間が 0 に近い領域では, 追跡対象物は静止しているとして追跡器は Stable 状態へと遷移しており , 計算負荷が軽減されていることがわかる．$10.5 < t < 11.5$ の領域では, 対象物を 人間が把持している状態であっても , 移動速度がしきい値以下であるとして Stable 状態へ遷移している．



(a) $t = 2$　(b) $t = 8.7$

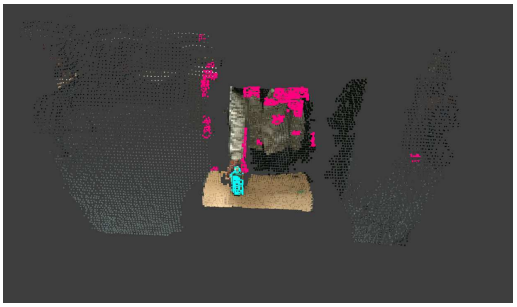
(c) $t = 12$　(d) $t = 16$

Fig6.4: Camera Views of Tracking Object with Change Detection Triggering

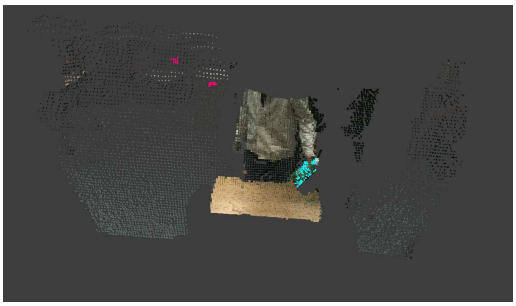
6.5.3　歩行計画実行時の監視機構

本小節では, 足配置計画によって計画された足配置の実行中に, その動作を監視し 実行器に割り 込みを 発生させるためのシステム構成について述べる．

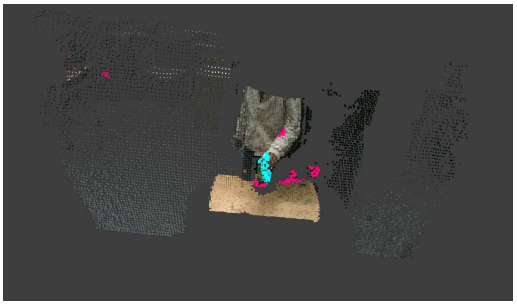
Fig. 6.8 に歩行実行器が時系列でどのようなスケジューリングを行うかについて示す．歩行中は両脚支持期・ 単脚支持期を 交互に繰り 返す．単脚支持期中は歩行実行器は割り 込み不可である．というのも , 動歩行中に動作を 中断してしまうと , バランスを 崩してしまいロボット は転倒してしまうためである．歩行を 停止するための割り 込みは両脚支持期中に行われる．単脚支持期中に歩行を 停止させるという 判断がなされた場合は, その割り 込み処理は次の両脚支持期まで先送り する．両脚支持期の停止タイミングは **Fig.** 6.8 において Breakpoint として図示している．



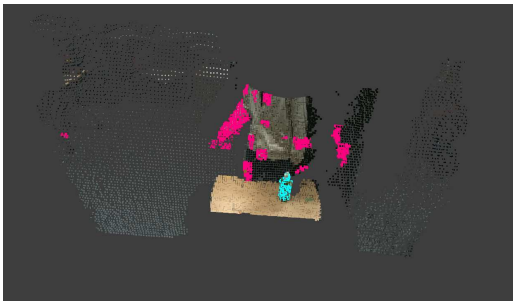
(a) $t = 2$



(b) $t = 8.7$



(c) $t = 12$



(d) $t = 16$

Fig6.5: Perception Result of Tracking Object with Change Detection Triggering. Red Points: Result of Change Detection. Blue Points: Result of Object Tracking.

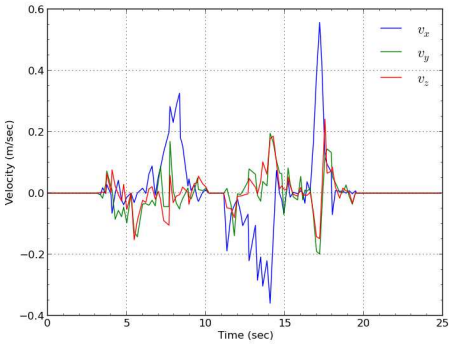


Fig6.6: Estimated Velocity of Tracking Object

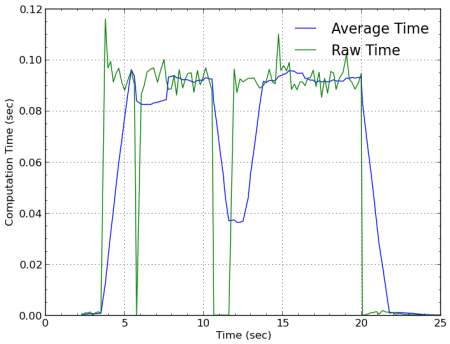


Fig6.7: Computation Time of Point-Cloud Tracking

床面状態監視器は歩行実行器はエンコーダおよび力センサを利用して床面状態の監視を両脚支持期中に行う．床面状態監視器は常に床面状態を確認しており，想定されている状態と異なっていた場合即座に歩行実行器に対して停止要求を送り，オペレータに対して判断を仰ぐためのユーザインタフェイスを立ち上げる (**Fig.** 6.9).

床面状態監視器は，計画時の足配置と実行中の足配置に関して足リンクの z 軸の傾きを評価することで，床面に想定していない障害物が存在するかを検知する (**eq.** 6.4).

$$acos(\mathbf{n}_{current} \cdot \mathbf{n}_{planned}) > thr_{foot} \tag{6.4}$$

Fig. 6.10, **Table** 6.1 に実際にいくつかの小さな物体の上でロボットを停止させた時の足リンクの z 方向とピッチ角の変化量を示す．このような力センサやエンコーダといった触覚に基づいた床面状態の確認は重要である．というのも非接触型のセンサであるステレオカメラやレーザレンジファインダを用いると小さな障害物を検知することは難しいからである．しかしながら，小さな障害物は制御器に大きな影響を与える可能性がある．また，歩行時にはこのような非接触型センサはオクルージョンによって床面状態を直接観測することは難しい．そのため，接触するリンクを用いて直接床面状態を推定することは有効である．

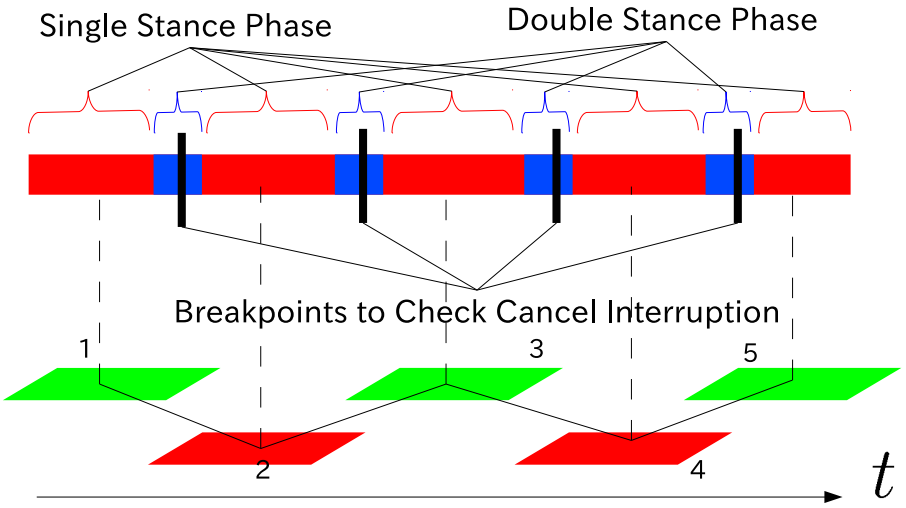


Fig6.8: Execution Timeline of Footstep Locomotion and Interruption

床面状態監視機能の効果を確認するため小さな障害物の上を歩行する実験を行った．障害物は直径 5mm の LAN ケーブルであり，その LAN ケーブルが床面上に置かれているという実験環境である (**Fig.** 6.11). この障害物は十分小さいため，ロボットは障害物を踏んでいても転倒せず制御することができる．しかしながら，ステレオカメラやレーザから取得される 3 次元点

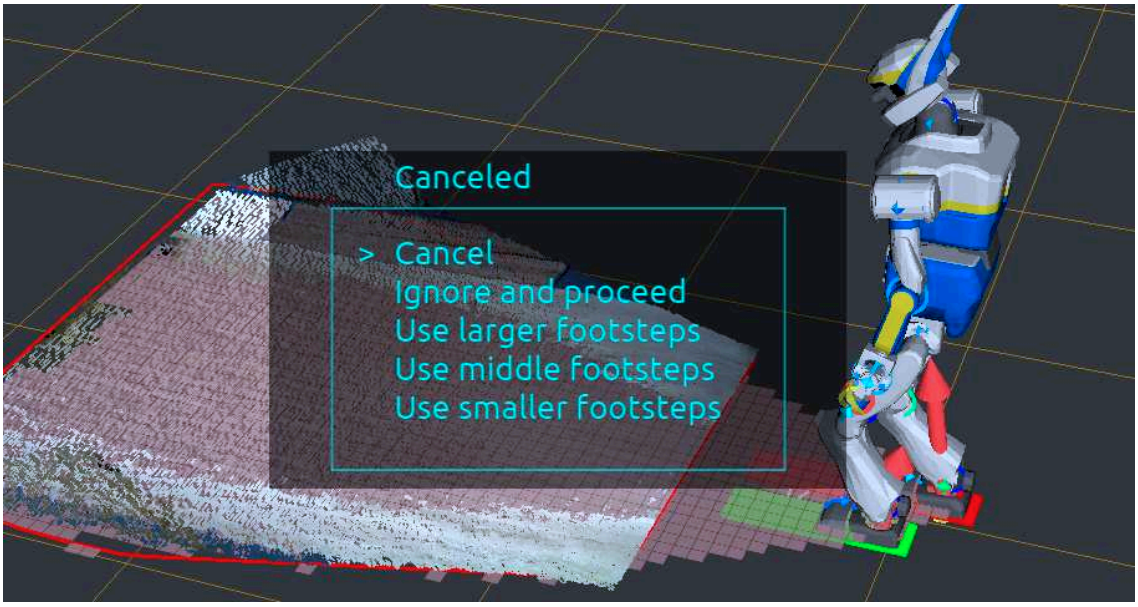


Fig6.9: Annotation Interface after Canceling Walking

(a) Flat Floor

(b) Headphone Cable (2mm)

(c) Pen (10mm)

(d) LAN Cable (5mm)

Fig6.10: Stepping over Small Obstacles

Table6.1: Z and Pitch Error of Leg End Effector when Stapping over Small Obstacles

Obstacle	Z Error (stddev)	Pitch Error (stddev)
Flat Floor (**Fig.** 6.10 (a))	0.010m (0.000016m)	0.0080 rad (0.00037 rad)
Headphone Cable (**Fig.** 6.10 (b))	0.017m (0.000019m)	0.016 rad (0.000042 rad)
Pen (**Fig.** 6.10 (c))	0.023m (0.000021m)	0.060 rad (0.00019 rad)
LAN Cable (**Fig.** 6.10 (d))	0.016m (0.000018m)	0.028 rad (0.00044 rad)

群ではでは検出が難しい大きさとなっている. ロボットがケーブルの上を通過するとき, 床面状態監視機能は計画時と実行時の誤差が大きいとしてケーブルを検知することができた (**Fig.** 6.11, **Fig.** 6.12). このときの計画時と実行時の足のエンドエフェクタの法線は 0.03 rad の差分があった. **Fig.** 6.12 における左上の赤い円は実行器が異常を検知したことを示している. 異常検知結果は遠隔操縦インタフェイスに送られ, オペレータはその場で足配置計画器のパラメータを変更することができる. **Fig.** 6.13 では, オペレータは”Use smaller footsteps”を選択し, ロボットを注意深くすすめるように指示している. 選択後, 新たなパラメータで計画された足配置は即座に反映される.

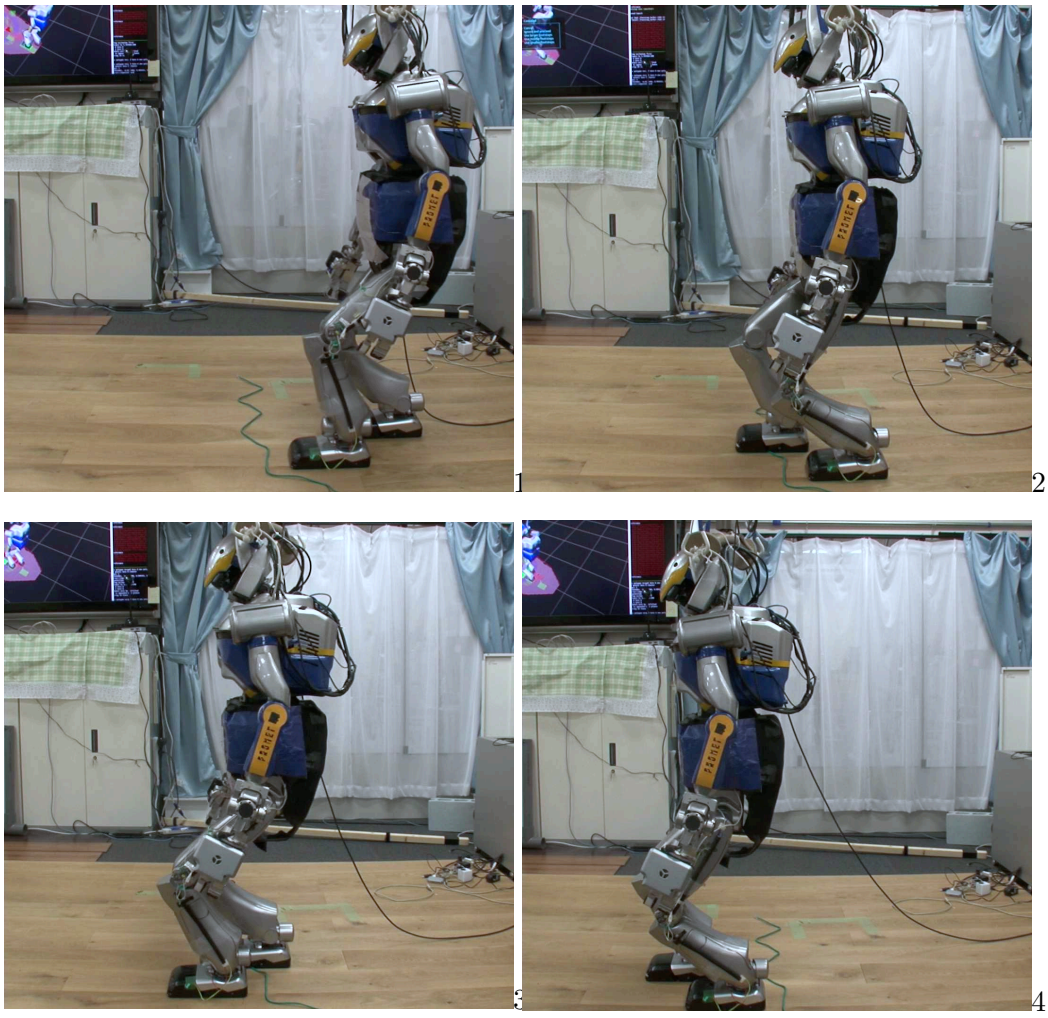


Fig6.11: An Experiment of Step on An Obstacle

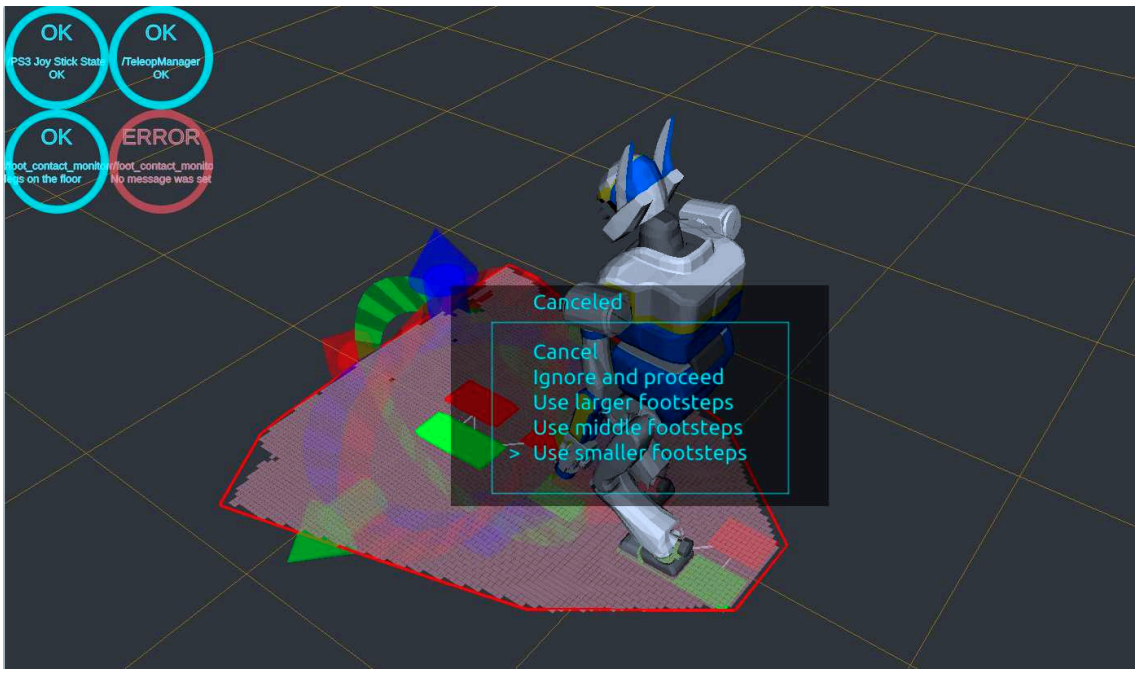


Fig6.12: Supervisor Display at The 2nd Picture of **Fig.** 6.11

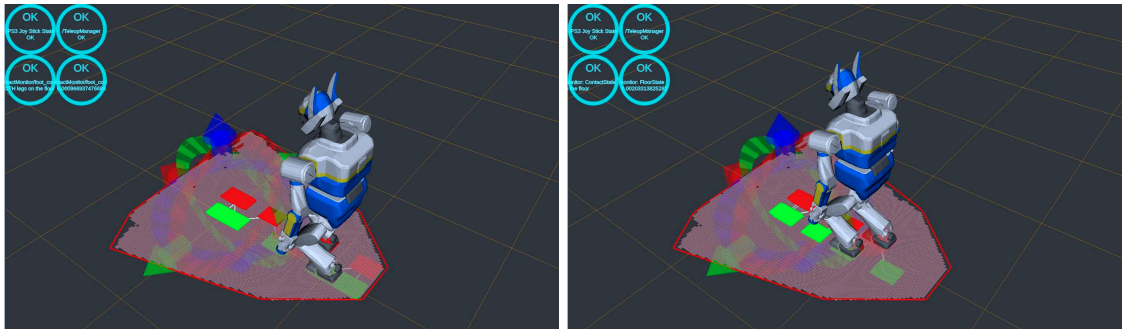


Fig6.13: Update of Parameters for Planner. Left) Using Default Footsteps. Right) Using Smaller Footsteps.

6.5.4　マニピュレーション計画実行時の監視機構

マニピュレーションは対象物との接触を伴う行動である．そのため, 環境との接触が生じる時刻に関して計画時と実行時の誤差に関する監視機能が必要となる．マニピュレーションの時系列での環境との接触状況は大きく分けて $t_{\mathrm{approach}}, t_{\mathrm{contact}}, t_{\mathrm{relase}}$ の 3 段階に分類できる．

t_{approach}　環境と接触せず, 対象物にマニピュレータを近づける段階

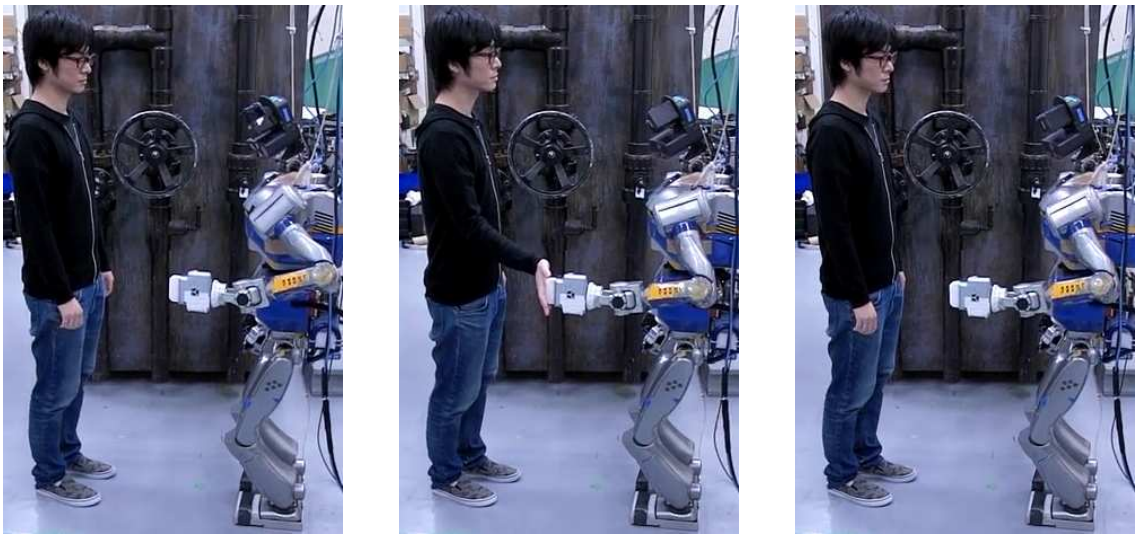
t_{grasp}　対象物を把持し, 操作を加える段階

t_{relase} 対象物からマニピュレータを離し, 環境との接触がなくなった段階

ある一定の時間間隔 Δt を設け, それぞれの段階でエンドエフェクタが環境と接触していないかを検知する. 環境との接触検知にはエンドエフェクタに搭載された力センサを利用する.

$$|\boldsymbol{f} - \boldsymbol{g}| < thr_{\mathrm{free}} \quad (0 \leq t \leq t_{\mathrm{approach}} - \Delta t) \tag{6.5}$$
$$|\boldsymbol{f} - \boldsymbol{g}| > thr_{\mathrm{contact}} (t_{\mathrm{approach}} + \Delta t \leq t \leq t_{\mathrm{contact}} - \Delta t) \tag{6.6}$$
$$|\boldsymbol{f} - \boldsymbol{g}| < thr_{\mathrm{free}} \quad (t_{\mathrm{contact}} + \Delta t \leq t \leq t_{\mathrm{release}} - \Delta t) \tag{6.7}$$

Fig. 6.14 環境との接触検知実験の結果を示す. 判定のしきい値である thr_{contact} は $thr_{\mathrm{contact}} = 15N$ とした. 接触時にはインピーダンス制御 [61] によってロボットの手先に検知される外力に対して手先位置をなじむように修正している. このような実験において力センサ値の変化は **Fig.** 6.16 のようであった. この力センサに基づく接触推定結果を **Fig.** 6.15 に示す. **Fig.** 6.15 では, 青色のリンクは非接触状態, 赤色のリンクは接触状態を示す.

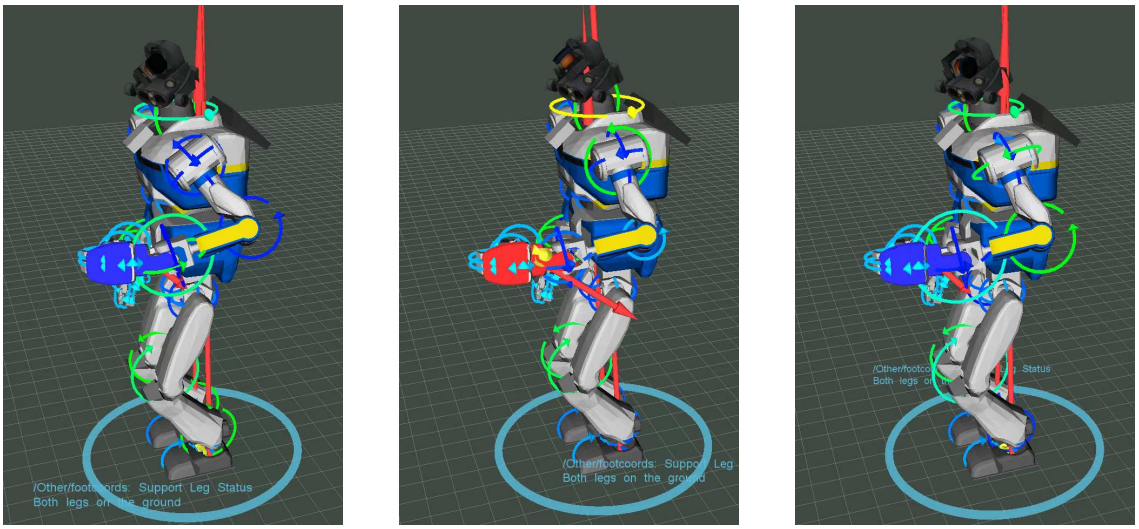


(a) $t = 2$sec (b) $t = 11$sec (c) $t = 15$sec

Fig6.14: Experiments of Contacting against Unexpected Environment under Impedance Control

また, 把持に関しては接触に加えハンドの関節角を利用して成否を判定する. ハンドの関節角を $\boldsymbol{\theta}$, 把持成功時のハンドの関節角を $\{\boldsymbol{\theta}_{\mathrm{success}}\}$, 把持失敗時のハンドの関節角を $\{\boldsymbol{\theta}_{\mathrm{fail}}\}$ とする. この時, ハンドの関節角度空間における距離を利用して把持の成功・失敗を分類する (**Fig.** 6.17).

$$\boldsymbol{\theta}_{\mathrm{min}} = \underset{\boldsymbol{\theta}_{\mathrm{sample}}}{\arg\min} |\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{sample}}| \tag{6.8}$$



(a) $t = 2$sec　(b) $t = 11$sec　(c) $t = 15$sec

Fig6.15: Contact Experiments under Impedance Control and Estimation of Contact based on Force Sensor. Blue Link Means Contact-Free State and Red Link Means Contacting State.

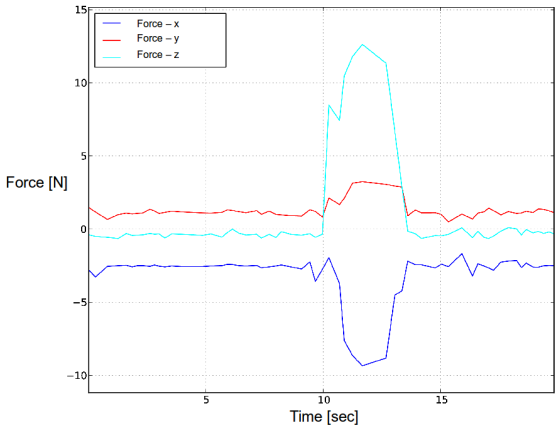


Fig6.16: Plot of Force Sensor Values in Contact Experiments (**Fig.** 6.14)

$$\boldsymbol{\theta}_{\text{sample}} \in \{\boldsymbol{\theta}_{\text{success}}\} \cup \{\boldsymbol{\theta}_{\text{fail}}\} \tag{6.9}$$

$$\text{graspSuccess}(\boldsymbol{\theta}) = \begin{cases} true & (if\ \boldsymbol{\theta}_{\text{min}} \in \{\boldsymbol{\theta}_{\text{success}}\}) \\ false & (if\ \boldsymbol{\theta}_{\text{min}} \in \{\boldsymbol{\theta}_{\text{fail}}\}) \end{cases} \tag{6.10}$$

$\{\boldsymbol{\theta}_{\text{success}}\}, \{\boldsymbol{\theta}_{\text{fail}}\}$ は実際に実機ロボット及び把持対象物を利用してサンプルを収集する.

Fig. 6.18, **Fig.** 6.19 に実機ロボットを用いたバルブの把持実験結果を示す. **Fig.** 6.18 は把持に失敗した場合, **Fig.** 6.19 は把持に成功した場合である. 右側の図は把持の成功・失敗の推定結果を示しており, 手先リンクの青色は非把持状態, 赤色は把持状態を示している.

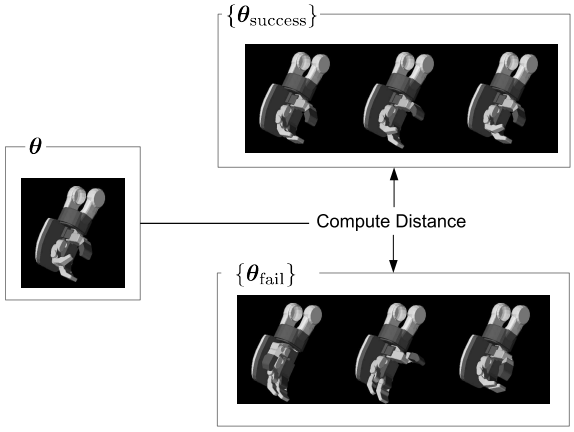


Fig6.17: Grasp State Observer

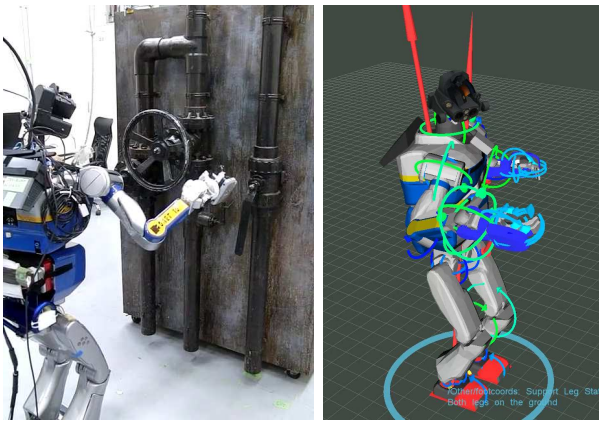


Fig6.18: Experimental Result of Grasping Valve and Detection: Left) View of Real Robot with Grasping Failure. Right) Detcetion Result. Links Colored Blue Means Estimation as Failure.

6.6　状態監視による局所的・大域的フィードバック

本節では, 局所的フィードバック, 大域的フィードバックによってシステムの環境の変化への即応性を向上させるシステム構成について述べる.

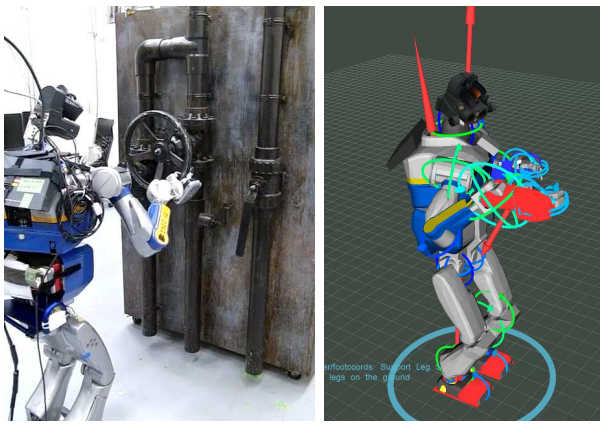


Fig6.19: Experimental Result of Grasping Valve and Detection: Left) View of Real Robot with Grasping Success. Right) Detcetion Result. Links Colored Red Means Estimation as Success.

6.6.1　局所的フィードバックのためのエラー検知とビジュアルフィードバックによるエラー復帰

局所的エラー検知

幾何的なレベルでのエラーを検出するのは難しい, というのも幾何的なレベルでのエラーはセンサのノイズの影響を受け, 環境に変化がなくてもエラーが常に発生する. そこで本システムでは, 幾何的なレベルでのエラーを検知というよりは, 幾何的なレベルの記述を常時更新する. これは, センサ入力が更新されるたびに, 幾何的なレベルの記述を更新するということである.

局所的エラー復帰

先に述べたように, 幾何的なレベルでの記述はは動作計画も含むシステムの他の部分とは非同期に更新される. これに対し, 動作計画器および Motion Controller は以下の要件を満たすことでビジュアルフィードバックを実現可能である.

- 常に最新の幾何的なレベルでの記述を利用した動作計画を行う.
- 常に最新の動作計画結果を用いてロボットの指令値とする.

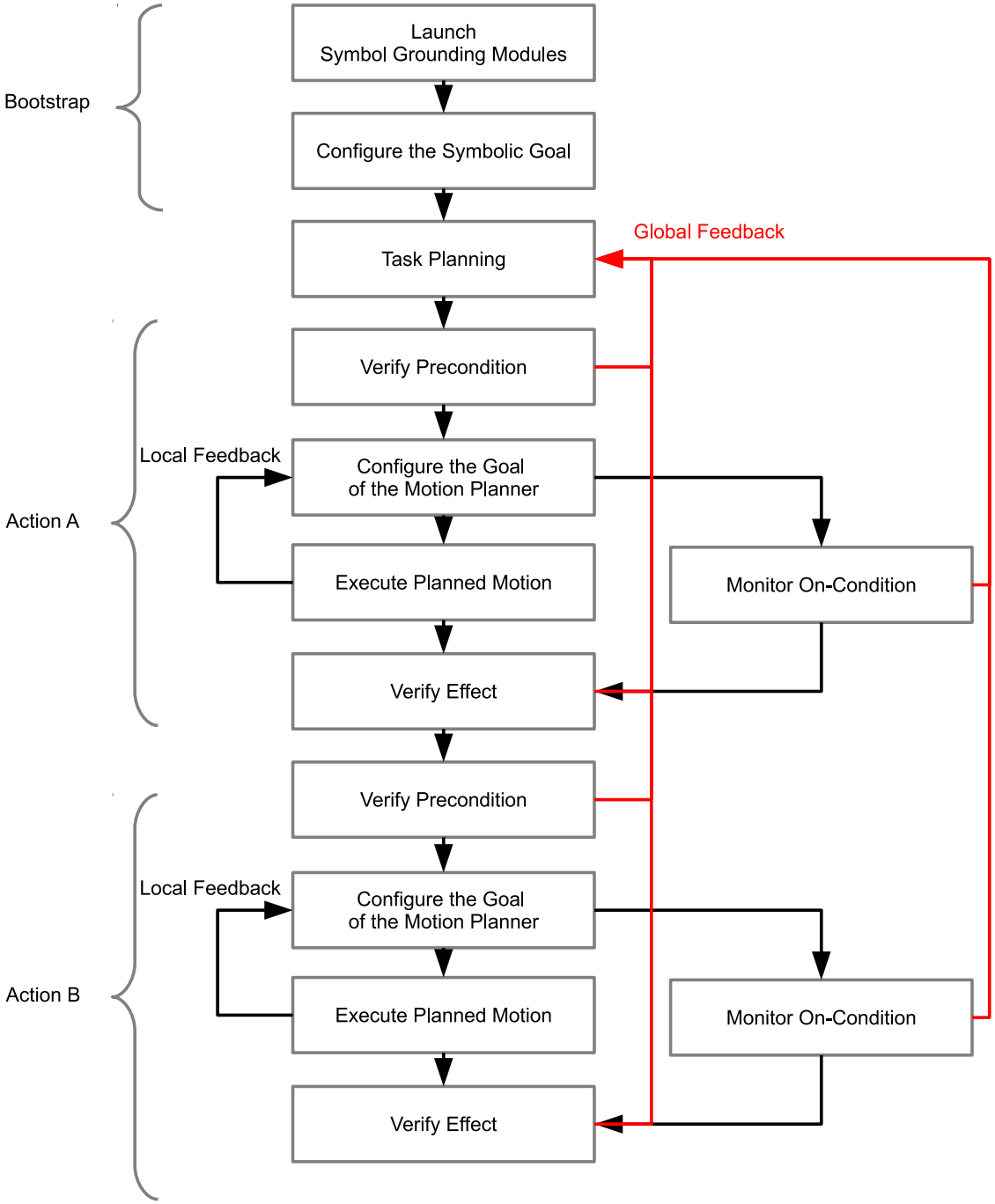


Fig6.20: Execution Timeline of Task Controller

6.6.2　大域的フィードバックのためのシンボリック記述におけるエラー検知と再計画によるエラー復帰

シンボリック記述におけるエラー検知による大域的エラー検知

先に述べたように大域的エラーはシンボリックなエラーと同一視される．シンボリックな表現は記号接地モジュールによって常に更新される．Task controller は行動の実行前・実行中・実行後の 3 つの区間でシンボリックな表現を監視する．シンボリックなタスク記述で想定されている記述と差分が検出されると, 大域的エラー復帰が実行される．

大域的エラー復帰

大域的エラー復帰は, 最もはじめにタスクプランニングを行うのと全く同様の手順によって行う．大域的エラーが検知されると即座に, Task Controller は現在実行している行動を停止する．その後に, 再度タスクプランニングを行うが, このときに必要とされるものは, 初期状態, 目標状態である．目標状態はすでにシステムに与えられているものを再び利用する．初期状態に関しては, 大域的エラー検出で述べたように, 常にシンボリックな記述は更新されている．これをそのままタスクプランニングに利用することが可能である．

6.6.3　実行時のタスクの遂行状態に従った監視機構スケジューリング

本小節では今までに述べた大域的フィードバックを含む実行手順について述べる．**Fig.** 6.20 に実行時の挙動について概要を示す:

1. 記号接地モジュールの立ち上げ

   Task Controller はシステムの初期化を行うが, この中で重要なものは記号接地モジュールの立ち上げである．PDDL によって記述された述語群から, 考慮しうる組み合わせに関する記号接地モジュールを列挙・立ち上げる．
2. 目標状態の設定

   ユーザが PDDL によって表現された目標状態を設定するとタスクプランニングが可能となり, 下位のレイヤへの展開が始まる．
3. タスクプランニングおよび行動列の生成

   タスクプランナは行動列をシンボリックなタスク記述に従って列挙する．記号接地モジュールはタスク記述における初期条件を常に更新している．

4. 前提条件の確認

   Motion Controller が行動を実行する前に, Task Controller は行動の前提条件が真となっているかを, 記号接地モジュールの出力に従って確認する.

5. 動作計画器の目標設定

   Task Controller は行動をロボットで実行するために, 動作計画器に目標値を設定する.

6. on 条件の監視を開始

   行動オペレータの on 条件は実行中に常に真となるべき条件である. Task Controller は常に記号接地モジュールの出力からこの on 条件が常に真であるかを実行中に監視する.

7. 行動の作用の確認

   行動は環境に対してシンボリックな変化を生じさせる. 行動動作の実行が終わった段階で, Task Controller は記号接地モジュールの出力から想定されているシンボリックな記述の変化が生じたかを確認する.

4), 6) および 7) の間において, Task Controller はシンボル記述の変化が想定していたものと異なるエラーを検知した場合, 3) のタスクプランニングまで戻る. シンボリックな記述におけるエラーを検出しなかった場合, 4), 5), 6) 7) を繰り返しタスクが遂行できるまで実行する.

6.6.4　行動の粒度設計

大域的なエラー復帰を効果的に行うためには, 行動の粒度を適切に設計することが重要である. というのも, 行動の粒度はエラー検知の細かさに直結するためである. このような行動の粒度設計は難しいが, 本論文では観測しうる一つのシンボリックな変化に対してひとつの行動を当てはめる, というのを基本的な方針としている. もちろん, シンボリックな記述の変化には依存関係があり, 同時に変化すべき値があるため, この方針に一致しないものが多くある. このような場合, 分割するのが難しい粒度まで行動を分割するという方針とした. このような方針に従うと, Task Controller はシンボルが変化すると即座にその影響を確認し, 大域的なエラー復帰が必要であるかを確認することができる.

6.7　大域的フィードバック行動・局所的フィードバック行動実験

Fig. 6.24, **Fig.** 6.25 に局所的エラー復帰および大域的エラー復帰の実験をそれぞれ示す. **Table** 6.2 にシンボリックな記述の中で利用される述語の一覧を示した. **Fig.** 6.21 はこの実

Table6.2: List of Predicates

predicate	description
On(Obj, Supporter)	True if *Obj* is on *Supporter*.
HandEmpty(Arm)	True if *Arm* does not have any objects.
Grasp(Arm, Obj)	True if *Arm* has *Obj* in its hand.
Poured(Obj)	True if *Obj* is filled with water.

験で使われる行動の定義一覧である. この実験ではロボットのタスクとしては机からペットボトルを掴み, お茶をくむというものであるが, タスク中に人によって介入を行う.

Fig. 6.24 および **Fig.** 6.25 の 2 つの実験ではどちらも同じ初期状態からタスクを始めている. **Fig.** 6.22 はタスクプランナによって初期状態から生成される行動列である. **Fig.** 6.24 に示す実験では, ロボットがペットボトルを把持しようと動き始めた時に, 人はペットボトルを机の上でスライドさせて位置を変化させた. その一方で, **Fig.** 6.25 に示す実験においては, 人はペットボトルを机から空中に一度移動させ, 後に机の上に戻した.

前者の実験においては, 大域的エラー復帰は必要でない. というのも人から加えられた外乱は **Table** 6.2 に示す述語で構成されたシンボリックな記述を変化させない. ロボットの動作とは非同期に実行されているビジュアルフィードバックによって変化した対象物の位置を常に更新され, その位置を反映させた動作計画が実行された.

一方で後者の実験においては人の外乱によって `REACH_GRASP` の on 条件である *On(PlasticBottle, Table)* に変化を生じる. それにしたがった大域的エラー復帰としてタスクプランニングの再計画が行われる. ペットボトルがテーブル上に再度置かれるまで, *On(PlasticBottle, Table)* は偽となっているため, タスクプランニングは失敗し, ロボットの行動列は制止されず, ロボットは静止したままとなる. ペットボトルがテーブルに再度置かれると *On(PlasticBottle, Table)* が真となり, タスクプランニングによって再度行動列が生成されロボットは再び動き始めた. 再度計画された行動列は **Fig.** 6.23 に示すようなものとなった.

6.8　おわりに

本章では, 高次のタスクプランニング機能で用いるシンボリックなタスク記述と連携して, 実行時にタスク状態を監視する監視機能について述べた. 環境や対象物が動的に変化する場合, 2 つのフィードバックによって環境の変化に対し速やかに対応可能なシステム構成であった.

```
reach_grasp(?arm ?from):
   precondition: (and (handempty ?arm)
                      (not (grasp ?arm ?obj))
                      (on ?obj ?from))
   effect: (and (not (handempty ?arm))
                (grasp ?arm ?obj))
   on_condition: (on ?obj ?from)

pick(?arm ?obj ?from):
   precondition: (and (grasp ?arm ?obj)
                      (on ?obj ?from))
   effect: (not (on ?obj ?from))

reach_put(?arm ?obj ?from):
   precondition: (and (not (handempty ?arm))
                      (grasp ?arm ?obj)
                      (not (on ?obj ?from)))

   effect: (on ?obj ?from)
   on_condition: (and (grasp ?arm ?obj)
                      (not (handempty ?arm)))

release(?arm ?obj):
   precondition: (and (not (handempty ?arm))
                      (grasp ?arm ?obj))
   effect: (and (handempty ?arm)
                (not (grasp ?arm ?obj)))

pour(?support_arm ?task_arm ?from_obj ?to_obj):
   precondition: (and
                   (not (poured ?to_obj))
                   (poured ?from_obj)
                   (grasp ?support_arm ?to_obj)
                   (grasp ?task_arm ?from_obj))
   effect: (and (poured ?to_obj)
                (not (poured ?from_obj)))
```

Fig6.21: List of Action Definition

```
1.  (REACH_GRASP LARM PLASTIC_BOTTLE TABLE)
2.  (PICK LARM PLASTIC_BOTTLE TABLE)
3.  (REACH_GRASP RARM CUP TABLE)
4.  (PICK RARM CUP TABLE)
5.  (POUR LARM RARM PLASTIC_BOTTLE CUP)
```

Fig6.22: Initial plans generated by task planner

```
1.  (REACH_GRASP LARM PLASTIC_BOTTLE TABLE)
2.  (REACH_GRASP LARM PLASTIC_BOTTLE TABLE)
3.  (PICK LARM PLASTIC_BOTTLE TABLE)
4.  (REACH_GRASP RARM CUP TABLE)
5.  (PICK RARM CUP TABLE)
6.  (POUR LARM RARM PLASTIC_BOTTLE CUP)
```

Fig6.23: Replanned actions (**Fig.** 6.25)

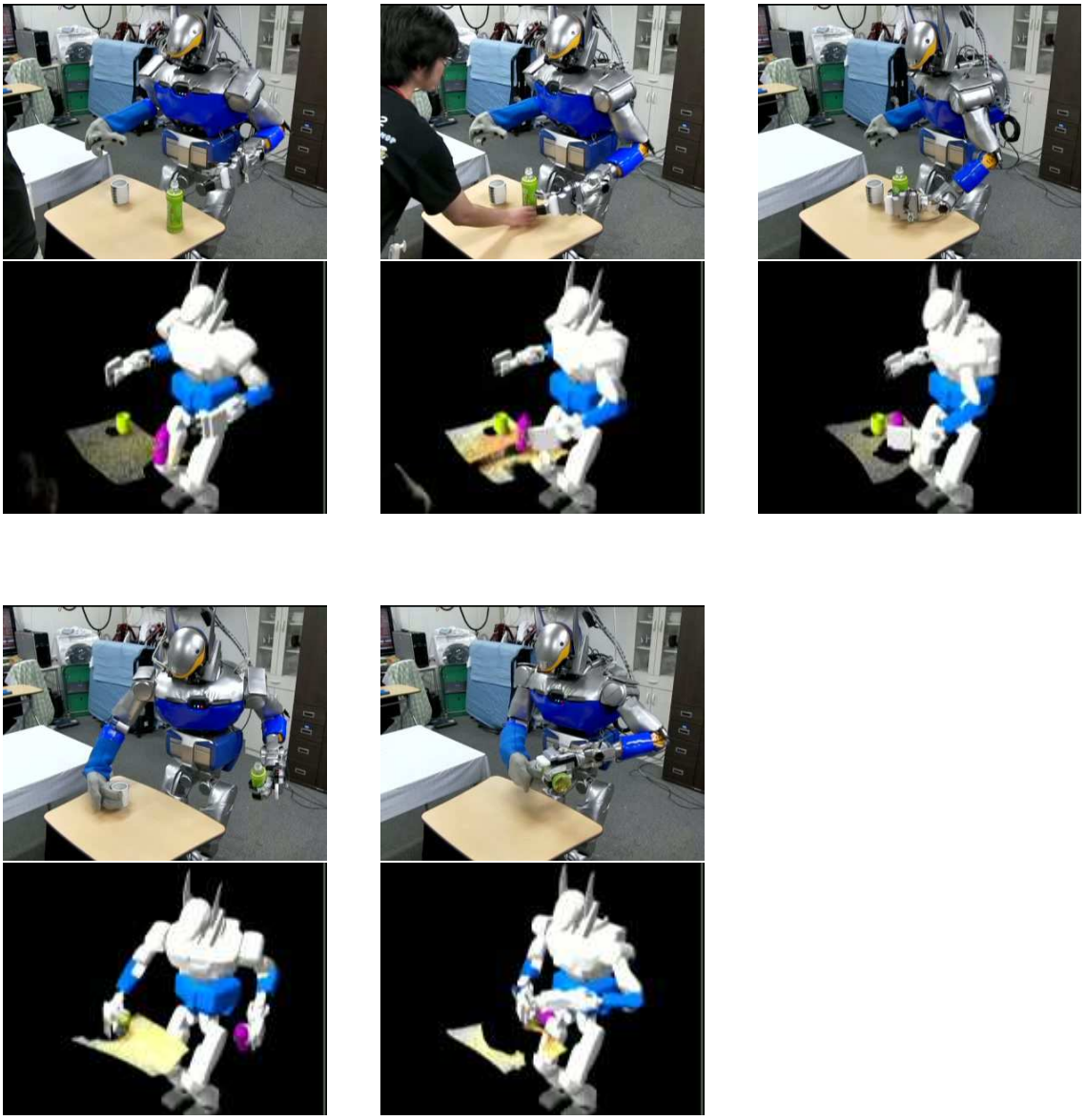


Fig6.24: Local error recovery by visual feedback. Lower images show the result of object detection and planned motion.

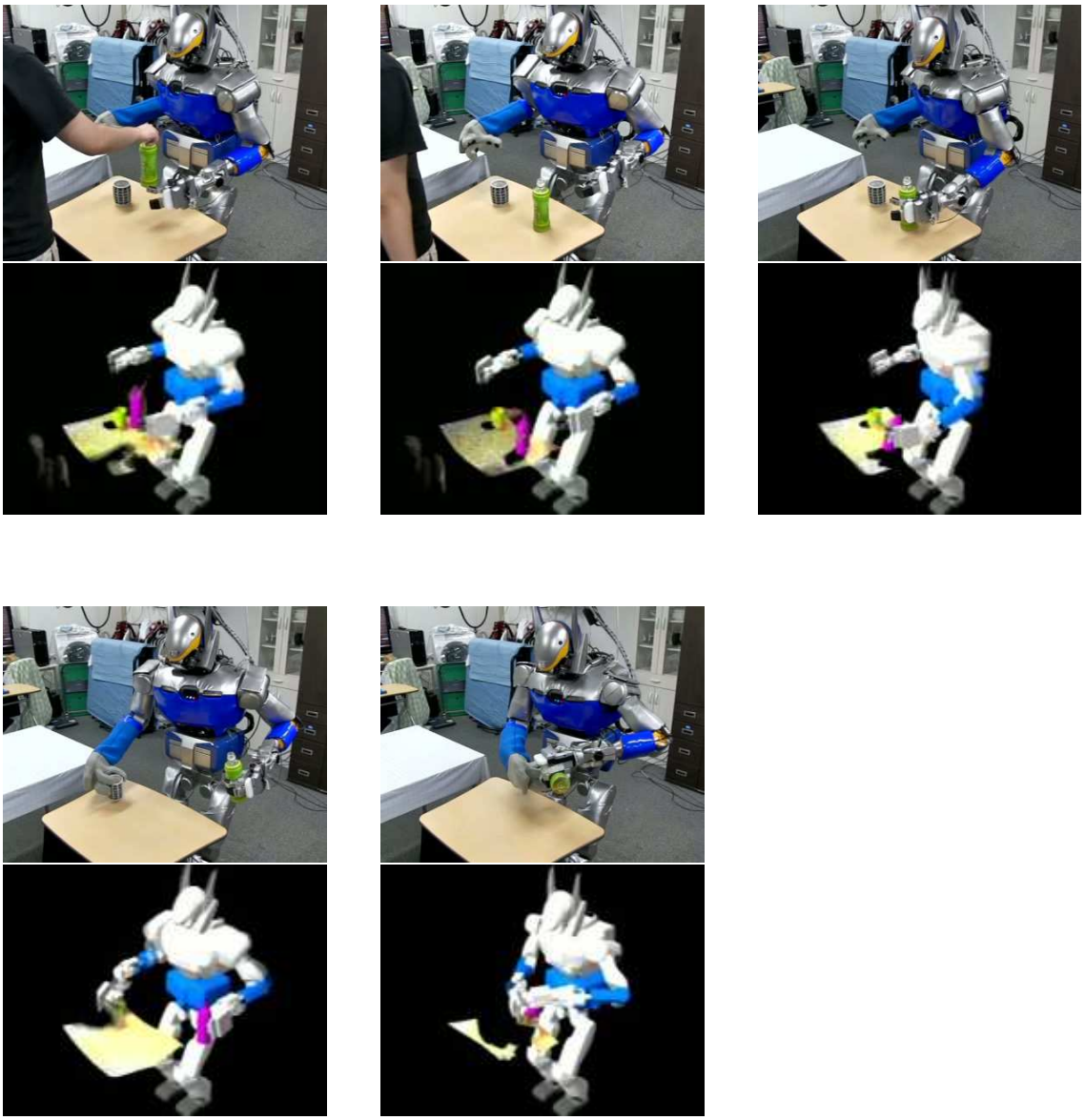


Fig6.25: Global error recovery by Monitoring of Symbolic Description Change. Lower images show the result of object detection and planned motion.

- シンボリックなレベルでのエラー解消のための大域的フィードバック
- 幾何的なレベルでのエラー解消のための局所的フィードバック

局所的フィードバックを可能とするためには, 第 5 章で述べた 3 次元点群追跡を常時実行し動作計画に最新の認識結果を反映することが重要であった. 常時更新される認識結果を利用して, 記号接地による環境のシンボリックな記述も常時更新することで想定と異なる場合に大域的なタスクプランニングを再度実行する大域的フィードバックを実現した.

また, シンボリックなタスク記述と連携するというよりは動作計画結果と連動して常時稼働する動作監視機能についても述べた. 動作監視機能については遠隔操縦インタフェイスを利用している場合でも実行されるものであった. 動作監視機能は動作計画器が想定していた状況とセンサ値から推定されるロボットと環境の関係が異なった場合に, タスクプランニングを再度実行もしくは遠隔操縦インタフェイスにオペレータの指示を仰ぐためのユーザインタフェイスを提示することでエラーをシステムにフィードバックする構成であった.

第 7 章

評価制御機構に基づくプロジェクトスケジューリングの実装と評価

7.1 はじめに

本章では, タスク遂行のための時間的制約を満たすように必要となる品質と時間を実行時に制御する評価制御機構に基づくプロジェクトスケジューリングについて実装と評価実験を行う (**Fig.** 7.1).

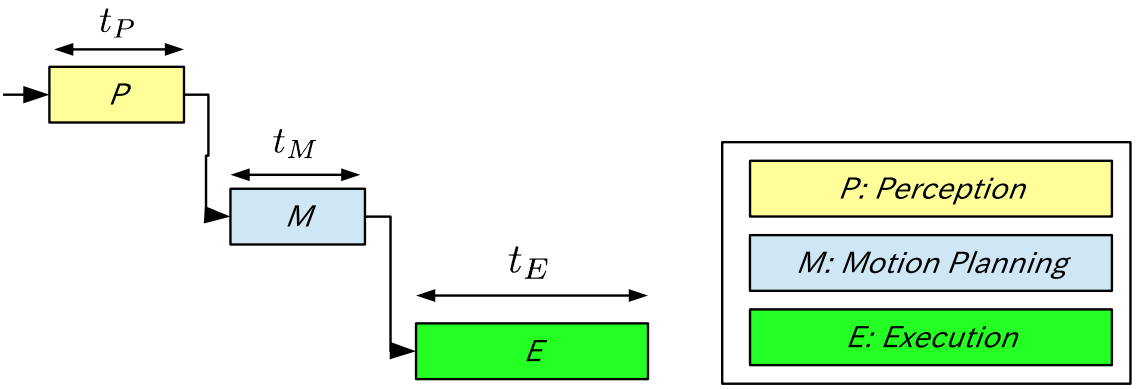


Fig7.1: Scheduling Duration of Perception, Motion Planning and Execution by Evaluation-Controlling Mechanism

本論文の目指す評価制御機構を備えたロボットシステムは必要時間に応じて認識, 動作計画, 動作実行の品質を変更可能なシステムである. 本章では第 2 章で述べた評価制御機構による認識, 動作計画, 動作実行の必要時間スケジューリング手法を実機ロボット上に実装し, 実験を通じて有効性を確認する. 品質として, タスク成功率に影響を与える評価値として品質を用いる. 本章で述べる内容は以下の 3 点に要約される.

- 評価制御のための品質-必要時間テーブルの構築方法
- 品質-必要時間テーブルに従った必要時間スケジューリング実験
- 品質に従ったスケジューリング時の実行モデルの選択実験

7.2　評価制御機構のための品質-時間テーブルの構築

本節では, 第 2.3.3 節で述べた評価制御機構のための品質-時間テーブルの構築方法について述べる.

7.2.1　逆運動学を用いた動作計画における品質-時間テーブル

本章では動作計画には逆運動学 [127, 128] による動作生成手法を用いる. また, [129, 130] において提案されている関節角度限界回避, および Sugiura et al. により提案されている自己衝突回避を考慮した逆運動学 [131, 132] を用いる.

本論文で逆運動学を用いた動作計画における品質において考慮するパラメータは以下の 2 つである.

$N_{\mathrm{collision}}$　自己衝突回避において考慮するロボットリンクのペア数. 衝突計算において考慮するリンク数が増えると, その結果得られる動作列の品質は向上する

N_{traj}　マニピュレーション起動を表現するための経由座標数. 軌道が連続に表現される場合, その軌道上から経由点をサンプリングする解像度が向上するほど得られる動作列の品質は向上する.

逆運動学の求解には Newton-Raphson 法による繰り返し計算を利用する. この時の最大繰り返し回数を $N_{\max}$ とする. 自己衝突回避を考慮することは, 安全な姿勢列を生成する上で重要であるが, 自己衝突回避のための衝突計算およびリンク間の距離計算は計算コストが高いため, 衝突計算が支配的な計算量を占める. マニピュレーションタスクを実現するための目標手先座標列を $\{R_1, R_2, \cdots, R_{N_{\mathrm{traj}}}\}$ とすると, 逆運動学計算は N_{traj} 回呼び出されることになる. したがってマニピュレーションタスク全体において逆運動学計算の計算量は $O(N_{\max} N_{\mathrm{collision}} N_{\mathrm{traj}})$ となる. 逆運動学の繰り返し計算最大回数 $N_{\max}$ は事前に一定の値を設定するため, 変更可能なパラメータに依存する計算量は $O(N_{\mathrm{collision}} N_{\mathrm{traj}})$ である.

Fig. 7.2 に考慮する衝突計算のリンク数を変化させた時の計算時間を図示する. また, **Fig.**

7.3 にマニピュレーション起動の粒度を変化させた時の計算時間を図示する. ただし, これら実験において逆運動学の繰り返し計算最大回数を $N_{\max} = 100$ とした. また, リンク間の距離計測には PQP[133] を用いた.

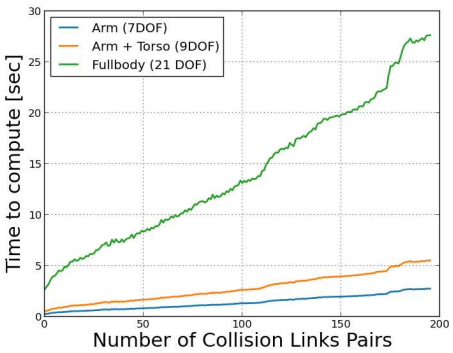


Fig7.2: Computational Time to Solve Inverse-Kinematics with Different Number of Collision Links

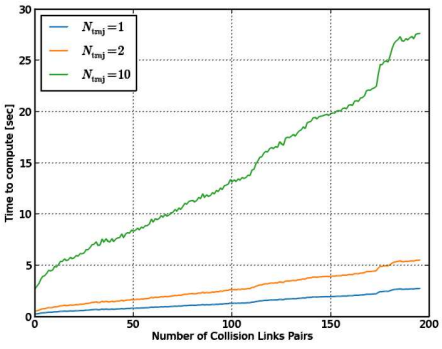


Fig7.3: Computational Time to Solve Fullbody Inverse-Kinematics with Different Number of Trajectory Resolution

このような関係に従って, 品質 q_M および計算時間 t_M の関係を求める. それぞれのパラメータに関する最大値, 最小値を考えることで, これらの値を値域 $[0, 1]$ に正規化する.

$$q_M(t_M(N_{\text{collision}}, N_{\text{traj}})) = \frac{N_{\text{collision}}}{N_{\text{collision,max}}} \frac{N_{\text{traj}} - N_{\text{traj,min}}}{N_{\text{traj,max}} - N_{\text{traj,min}}} \tag{7.1}$$

$N_{\text{collision}}$ に関する最小値は 0 であり, 最大値 $N_{\text{collision,max}}$ はロボットのリンク構成から明らかである. また, N_{traj} に関しては, タスク遂行に必要最低限な経由点数 $N_{\text{traj,min}}$ はタスク依存であるため, システム実装者が与える必要がある. 最大経由点数 $N_{\text{traj,max}}$ に関しては十分に大きな数を与える. **Fig.** 7.4 に $N_{\text{traj,min}} = 3, N_{\text{traj,max}} = 10$ とした時の品質 q_M-時間 t_M テーブルを図示する. このとき, q_M は 2 つの変数 $N_{\text{collision}}$ および N_{traj} の関数となるため, 短調増加な q_M を構成するような組み合わせ (**Fig.** 7.4 における橙の点列) のみを用いる.

7.2.2　脚型ロボットの動作実行速度に依存した品質-時間テーブル

第 2.3.3 節で述べたように, 動作実行の速度は脚型ロボットの安定性に大きな影響を与える. 本節では動作速度と脚型ロボットの安定性の関係を品質-時間テーブルとして整理する. 本論文では脚型ロボットの安定性は以下の二つの指標に基づいて評価する.

1. 動作中の姿勢および速度から推定される転倒への安定性に基づく品質 q_{zmp}
2. 環境との衝突が発生した際に期待される外乱に基づく品質 q_{impulse}

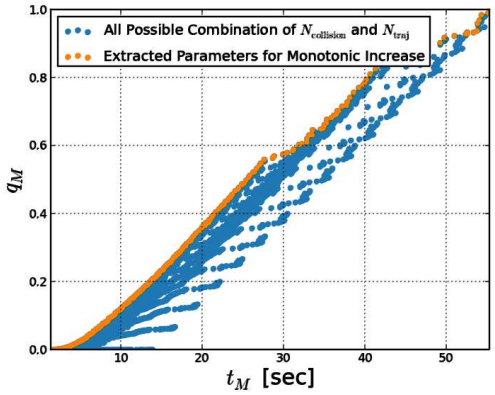


Fig7.4: Motion Quality q_M from **eq.** 7.1 and **Fig.** 7.3. Blue Points are All the Combination of N_{traj} and $N_{\text{collision}}$. Orange Points are Selected Data for Monotonic Increase.

脚型ロボット の動作実行速度に依存した品質 q_E は, これらの積として表現する.

$$q_E(t_E) = q_{\text{zmp}} q_{\text{impulse}} \tag{7.2}$$

Fig. 7.5 にヒューマノイドロボット 全身の関節を 利用することで手先をリーチングするという 簡単なタスクの動作を 示す. この動作列を 例として, 脚型ロボット の動作実行速度に依存した品質-時間テーブルの構築方法について 述べる. **Fig.** 7.5 に示す動作は静的に安定な動作とするため, 重心位置を x, y 方向に変化させないように動作生成する. 動作速度はロボット のハードウェア的な制約である 各アクチュエータの最大速度に 従って 決定し, その最大速度から比例させた速度で全身の関節角度列を 動かす.

Fig7.5: Simple Reaching Motion with Fullbody Inverse Kinematics

動作中の姿勢および速度から 推定される 転倒への安定性に基づく 品質には Zero Moment

Point(ZMP)[134] を指標として用いる．ZMP は歩行軌道計画やバランス制御において広く利用されている指標であり，ZMP が接地している脚が作る支持多角形の外縁上に位置するとき，2 足型ロボットは静止することが出来ず転倒に繋がる．ZMP の位置 (x_P, y_P) は [135] より以下の **eq.** 7.3, **eq.** 7.4 のようになる．

$$\begin{aligned} x_P = & \frac{-\dot{L}_{G_y} + Mx_g(\ddot{z}_G + g) - M(z_G - z_P)\ddot{x}_G}{M(\ddot{z}_G + g) - \sum f_{H_{j,z}}} \\ & - \frac{\sum(x_{H_j} f_{H_{j,z}} - (z_{H_j} - z_P) f_{H_{j,x}})}{M(\ddot{z}_G + g) - \sum f_{H_{j,z}}} \end{aligned} \tag{7.3}$$

$$\begin{aligned} y_P = & \frac{-\dot{L}_{G_x} + My_g(\ddot{z}_G + g) - M(z_G - z_P)\ddot{y}_G}{M(\ddot{z}_G + g) - \sum f_{H_{j,z}}} \\ & - \frac{\sum(y_{H_j} f_{H_{j,z}} - (z_{H_j} - z_P) f_{H_{j,y}})}{M(\ddot{z}_G + g) - \sum f_{H_{j,z}}} \end{aligned} \tag{7.4}$$

ここで，L_{G_x}, L_{G_y} は重心周りの角運動量である．ロボットの動作速度が速くなれにつれて，ZMP の位置は速度および加速度に依存した項によって変化量が大きくなる．動作中の重心位置は一定のため，$\ddot{x}_G = \ddot{y}_G = 0$ となる．ZMP の変化量 d_{zmp} は ZMP の位置 (x_P, y_P) によって表現される (**eq.** 7.5).

$$d_{\mathrm{zmp}} = \sqrt{x_P^2 + y_P^2} \tag{7.5}$$

Fig. 7.6 に示すように，動作速度が速くなるに従って ZMP の変化量 d_{zmp} は大きくなる．ZMP はその定義から支持多角形内部に存在するため，d_{zmp} の取りうる値は以下のようになる．

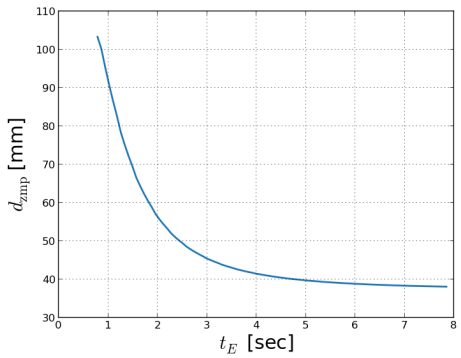


Fig7.6: Error of ZMP d_{zmp}

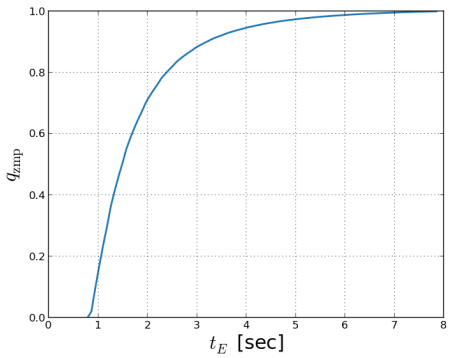


Fig7.7: Quality q_{zmp} based on Error of ZMP

$$0 \leq d_{\mathrm{zmp}} \leq d_{\mathrm{max}} \tag{7.6}$$

ここで d_{max} は重心座標から最も遠い支持多角形の頂点までの距離である．これらを利用して

品質 q_{zmp} を定義する (**Fig.** 7.7).

$$e_{\mathrm{zmp}} = \frac{d_{\max} - d_{\mathrm{zmp}}}{d_{\max}} \tag{7.7}$$

$$q_{\mathrm{zmp}}(t_E) = \frac{e_{\mathrm{zmp}} - e_{\mathrm{zmp,min}}}{e_{\mathrm{zmp,max}} - e_{\mathrm{zmp,min}}} \tag{7.8}$$

q_{zmp} は最大値, 最小値を 利用して $[0, 1]$ の値域に正規化している.

一方, 環境との衝突が発生した際に期待される外乱に基づく 品質 q_{impulse} に関しても同様にロボット の動作速度が速くなるに従って減少する. 本論文では環境との衝突時の影響をエンドエフェクタに発生する力積の期待値として評価する. エンド エフェクタの質量を m, 時刻 t におけるエンド エフェクタ速度を $v(t)$ とすると, 時刻 t において環境と 衝突した時に発生する力積 $N\Delta t$ は以下のようになる.

$$N\Delta t = mv(t) \tag{7.9}$$

ロボット の動作中に一定確率 p で環境と 衝突すると 考えると, 発生する 力積の期待値 E_{impulse} は **eq.** 7.10 のようになる.

$$E_{\mathrm{impulse}} = \int_{T_{\mathrm{traj}}} pmv(t)dt \tag{7.10}$$

ここで, T_{traj} はマニピュレーション 軌道を 意味する. エンド エフェクタ速度 $v(t)$ は各アクチュエータの最大速度の比例値として決定する. また, 最低速度は十分に小さい値として, アクチュエータ 最大速度の 0.1 とした. 環境との衝突による 外乱に基づく 安定性による品質を q_{impulse} とすると, q_{impulse} は動作の最大速度および最低速度の間で力積の期待値 E_{impulse} によって $[0, 1]$ に正規化した値を用いる (**eq.** 7.11, **Fig.** 7.8).

$$q_{\mathrm{impulse}}(t_E) = \frac{E_{\mathrm{impulse}} - E_{\mathrm{impulse,min}}}{E_{\mathrm{impulse,max}} - E_{\mathrm{impulse,min}}} \tag{7.11}$$

Fig. 7.9 に q_{zmp} および q_{impulse} から **eq.** 7.2 に従って計算した動作実行速度に依存した品質 q_E を図示する.

7.2.3　点群認識処理における 品質-時間テーブル

認識処理における 品質と 時間の関係性は, 各種認識器のアルゴリ ズム, 実装および環境に強く依存する. アルゴリ ズムに依存するパラメータはそれぞれの認識器においてモデル化および計測によって品質と 時間の関係を 評価する. 本小節では 3 次元点群を 利用するアルゴリ ズムにおいては, その実装に依存せず高速化が可能である 立方体格子による 低解像度化の効果を 見積る.

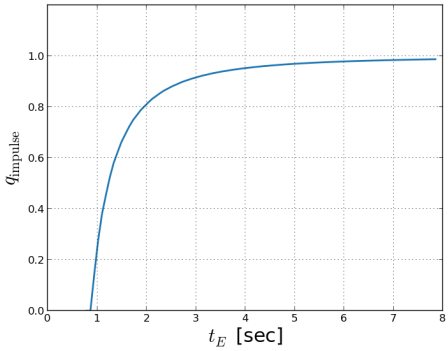


Fig7.8: Quality q_{impulse} based on Expected Impulse by Collision with Environment

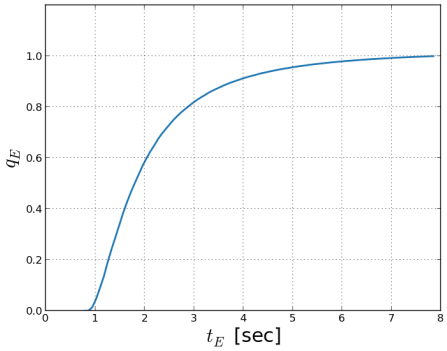


Fig7.9: Merged Execution Quality q_E

立方体格子の格子のサイズを d_{voxel} とすると, 入力 3 次元点群は立方体格子による低解像度化によって最大 d_{voxel} だけ座標が変化する. 入力点群全体が同一方向に d_{voxel} 移動したと考えると, 対象物の認識位置は同様に最大 $e_{\mathrm{translation}}$ だけ移動する.

$$e_{\mathrm{translation}} = \sqrt{3} d_{\mathrm{voxel}} \tag{7.12}$$

eq. 7.12 に従い, 本章では有意な結果を返す d_{voxel} の値域 $[0, d_{\mathrm{voxel,max}}]$ において, 品質は線形に変化するという簡易化されたモデルを利用する. $d_{\mathrm{voxel,max}}$ は格子サイズ d_{voxel} が認識対象物の大きさや形状に従って認識器が推定に失敗する値として設定する.

$$q_P(t_P(d_{\mathrm{voxel}})) = \frac{d_{\mathrm{voxel,max}} - d_{\mathrm{voxel}}}{d_{\mathrm{voxel,max}}} \tag{7.13}$$

eq. 7.13 は精度の見積が難しい 3 次元点群の認識アルゴリズムにおいて, 点群の低解像度化を制御パラメータとした場合に共通の簡易モデルである. **Fig.** 7.10 に d_{voxel} 変化させた時に必要となる計算時間を実測したものを示す. **Fig.** 7.10 において実行している認識処理は 3 次元点群から RegionGrowing によって複数の平面領域を抽出する認識処理である. ただし, $d_{\mathrm{voxel}} = 0$ のときは立方体格子による低解像度化処理を行なわずに, センサからの入力点群をそのまま利用している. **Fig.** 7.10 および **eq.** 7.13 に基づいた, 品質 q_P-計算時間 t_P の関係を **Fig.** 7.11 に示す. ただし ,t_P の算出においては, センサデータのサンプリングのための時間 3.14sec を含んでいる. これは 2 次元レーザが 1 rad/sec でロール軸周りに半回転することで 3 次元環境の点群データを観測するのに必要な時間である.

一方, 継続的認識機能に関する品質-時間テーブルは, 第 5.7.3 節における実験結果を利用して作成する. 時間や精度に影響を与えるパラメータは複数あるが, 第 5.7.3 節での議論から, 静

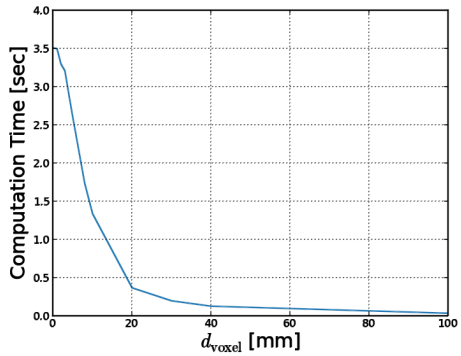


Fig7.10: Computational Time of Segmentation of Multiple Planar Regions using RegionGrowing with Different d_{voxel}

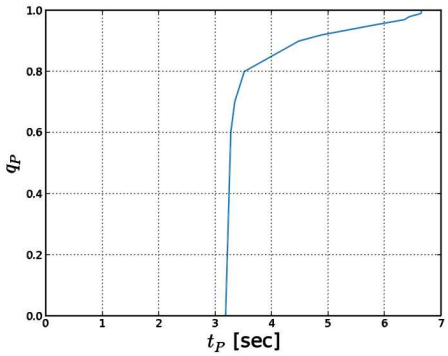


Fig7.11: Quality q_P and Duration t_P from **eq.** 7.13 and **Fig.** 7.10.

的な環境においては格子サイズ d_{voxel} が支配的なパラメータであった. ここで, 品質 q_{tracking} を以下のようにして決める.

$$err = err_{\mathrm{translation}}(d_{\mathrm{voxel}})err_{\mathrm{rotation}}(d_{\mathrm{voxel}}) \tag{7.14}$$

$$q_{\mathrm{tracking}}(t_{\mathrm{tracking}}(d_{\mathrm{voxel}}))) = \frac{1}{1+\alpha err^2} \tag{7.15}$$

eq. 7.13 では誤差のモデルを 立式することで品質の見積もりをおこなったが, 継続的認識機能に関する品質では, 実際の測定結果を 用いて品質を 評価している. これは, トラッキングという認識器の性質上, 静止環境を 用いることで真値の設定が可能だからである. **eq.** 7.15 では, 測定した中で最小の err のとき q_{tracking} は 1 となるように係数 α を 30 として設定している. また, 第 5.7.3 節の計測結果を 直接用いると計測のノイズが大きいため, **Fig.** 5.25 に灰色で示した指数関数および 2 次関数で近似したモデルを 利用している. これらから 得られる 3 次元点群追跡を 利用した品質 q_{tracking}-時間 t_{tracking} テーブルを **Fig.** 7.12 に示す.

7.3　評価制御機構を利用した必要時間制御によるプロジェクトスケジューリング実験

本節では, 第 2 章おいて述べた評価制御機構によるスケジューリングの有効性を確認するための実機ロボットによる実験を行う. タスクとしては, ロボットがドアの近くまで移動する移動タスクとその後にドアハンドルを把持してドアを開くマニピュレーションタスクの 2 つを組み合わせたものである. **Fig.** 7.13 にロボットがこれらのタスクを実行している様子を示す.

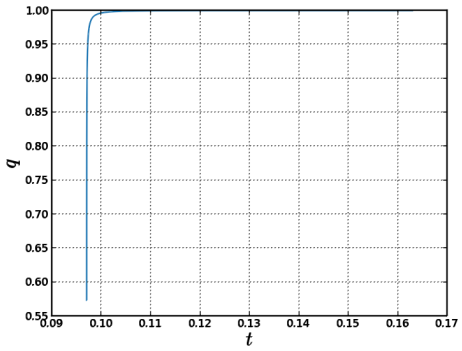


Fig7.12: Quality q_{tracking} and Duration t_{tracking} from **eq.** 7.15 and **Fig.** 5.25.

Fig7.13: Move to Door and Open a Door by JAXON

7.3.1　実験設定

実験結果の詳細に先立ち, 本節での実験に用いる動作計画や認識処理について述べる.

認識処理

動作計画に必要な環境の情報はドアハンドルの位置と姿勢であるため, 認識処理ではこれらの値を推定する必要がある. 本実験ではドアハンドルの取り付けられているドアの平面を推定し, その平面より手前側に取り付けられているドアハンドルを直方体形状として 3 次元点群から推定する. ドア平面の推定には第 7.2.3 節でも述べた Region Growing による複数平面推定を用いる. ドアの大きさは既知であるとして, 推定された複数の平面からドアを発見する. また, ドアハンドルに関しては検出されたドア平面上に観測される点群をユークリッド距離でク

ラスタリングする. その結果事前に定義したドアハンドルの大きさと近いクラスタをドアハンドルとして検出する. この認識器の品質-時間テーブルは **Fig.** 7.11 に示したものと同一のものを利用した.

また, 継続的認識機能の実現には第 5 章で述べたパーティクルフィルタによる 3 次元点群追跡手法を利用した.

動作計画処理

タスクに依存した動作計画に関するパラメータとしては, エンドエフェクタ軌道およびその解像度であった. ドア開けマニピュレーションタスク実現のための最小のエンドエフェクタ軌道は **Fig.** 7.14 に赤線で示したものであり, タスク遂行のための最低限の経由点を $\{R_1, \cdots, R_5\}$ として示した. 最大, 最小の経由点数は $N_{\mathrm{traj,min}} = 5, N_{\mathrm{traj,max}} = 20$ とした. この時の動作計画処理の品質-時間テーブルは第 7.2.1 節での議論に従って **Fig.** 7.15 のよう

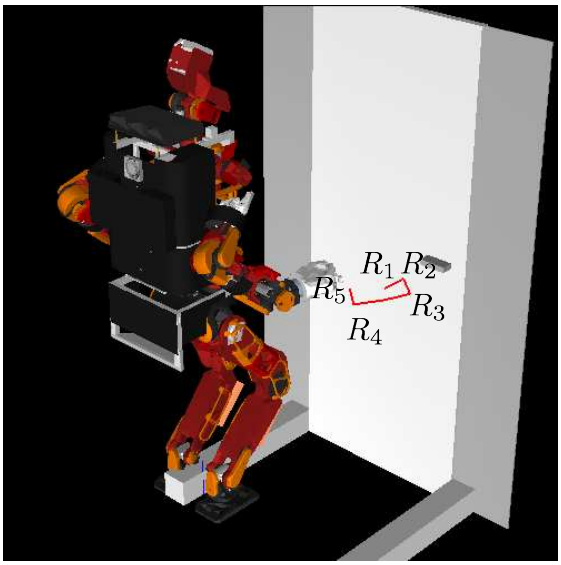


Fig7.14: Manipulation Trajectory of Door-Open Task. $\{R_1, \cdots, R_5\}$ are Minimum Waypoints.

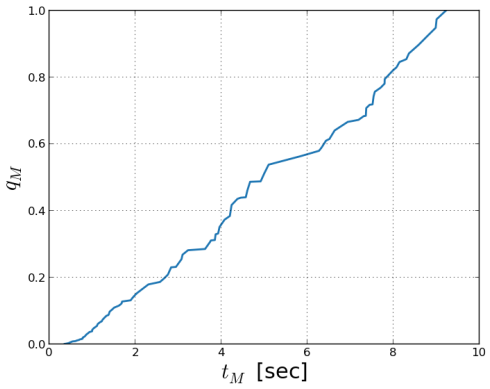


Fig7.15: Motion Quality q_M from **eq.** 7.1 and Door Trajecotry (**Fig.** 7.14)

になる.

動作速度の品質評価

エンドエフェクタ軌道がタスク依存であることから, 動作速度に依存する実行部の品質-時間テーブルはタスクごとに作成する必要がある. **Fig.** 7.14 に示した動作軌道から, **eq.** 7.8 に

従って求めた関節速度に依存する実行部の品質-時間テーブルを **Fig.** 7.16 に示す. また, 実験

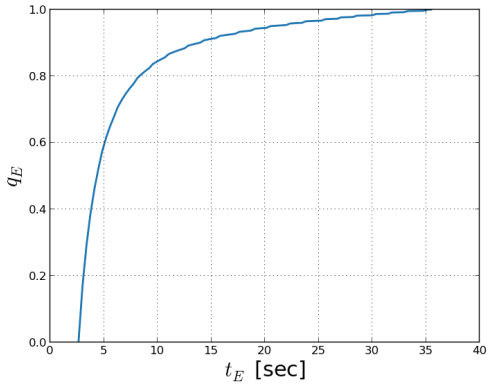


Fig7.16: Execution Quality-Time Table q_E for Door Manipulation Task

の都合から歩行に関する動作速度は一定とする. そのため, 移動に関する時間 t およびその品質 q を無視して実験を行った.

7.3.2　移動を伴わない実行モデルによるスケジューリング実験

まず, 基本的な実行モデルである移動を伴わない認識計画実行モデル (第 2.5 節) を用いたスケジューリング実験を行う. **eq.** 7.16, **eq.** 7.17 に時間と品質の関係を示す.

$$t = t_{P_0} + t_{M_0} + t_{E_1} \tag{7.16}$$

$$q = w_{P_0} q_{P_0} + w_{M_0} q_{M_0} + w_{E_0} q_{E_0} \tag{7.17}$$

Fig. 7.17 に $q_{P_0} - t_{P_0}, q_{M_0} - t_{M_0}, q_{E_0} - t_{E_0}$ およびそれらの初期値を図示する. ただし, このとき $w_{P_0} = w_{M_0} = w_{E_0} = 1$ としている. また, 初期値はシステム実装者が与えている. この品質-時間テーブルを利用し, 目標となるデッドライン時間 t_{d} を 0 としたときのパラメータの変化の挙動を **Fig.** 7.18 に矢印で示す. 逆に, 目標となるデッドライン時間 t_{d} を十分に大きくしたときのパラメータの変化を **Fig.** 7.19 に矢印で示す. また, $w_{E_0} = 10$ とし, デッドライン時間 t_{d} を 0 としたときのパラメータの挙動を **Fig.** 7.20 に示す. **Table** 7.1 に様々なデッドライン時間を与えた時の実際に決定される各種必要時間と品質の実験結果を示す. **Fig.** 7.18 および **Fig.** 7.19 に示したパラメータの挙動から, 要求されたデッドライン時間 t_{d} にしたがって効果の大きい要素の品質が制御されていることがわかる. また, $w_{E_0} = 10$ のように重み係数を大きくするということは, その要素の品質を優先して考慮したスケジューリングを行

うという 事である . $w_{E_0} = 1$ である **Fig.** 7.18 でははじ めに動作速度を 大きく 速めるような挙動がみられる 一方で, $w_{E_0} = 10$ では動作速度に関する 品質が優先されているため, **Fig.** 7.20 においては動作計画器の品質を 下降させる 挙動が早い段階で出ていることを 示していることが確認できる .

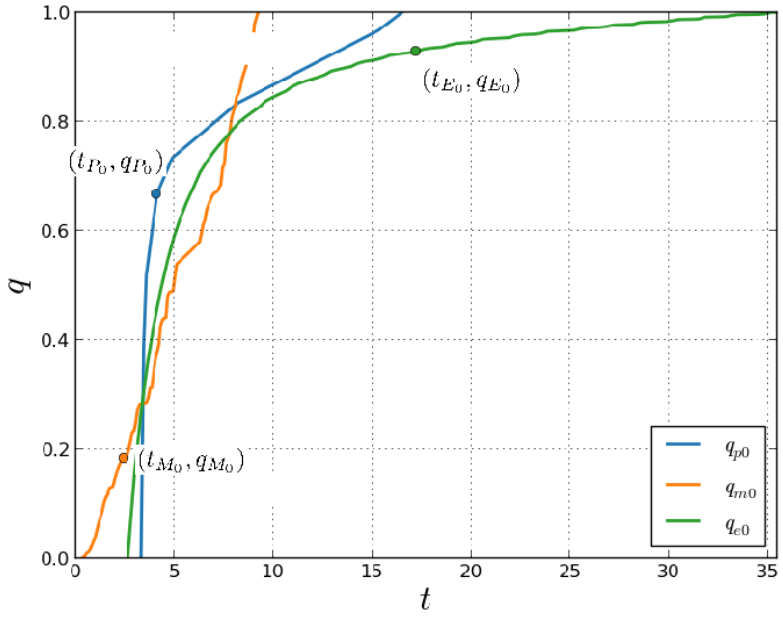


Fig7.17: Quality-Time Table $q_{P_0}, q_{M_0}, q_{E_0}$ and Initial Values

Table7.1: Experimental Result of Project Scheduling

t_d	t	q	t_{P_0}	t_{M_0}	t_{E_0}	d_{voxel}	N_{traj}	$N_{\text{collision}}$
50	50.0	2.93	16.4	9,02	24.7	8.3	38	29
40	40.1	2.51	16.4	6.32	17.5	8,3	23	28
23.7	23.7	1.78	4.10	2.42	17.2	10	7	27
20	19.9	1.74	4.10	2.42	13.3	10	7	27
10	9.95	1.27	4.10	1.22	4.63	10	4	21

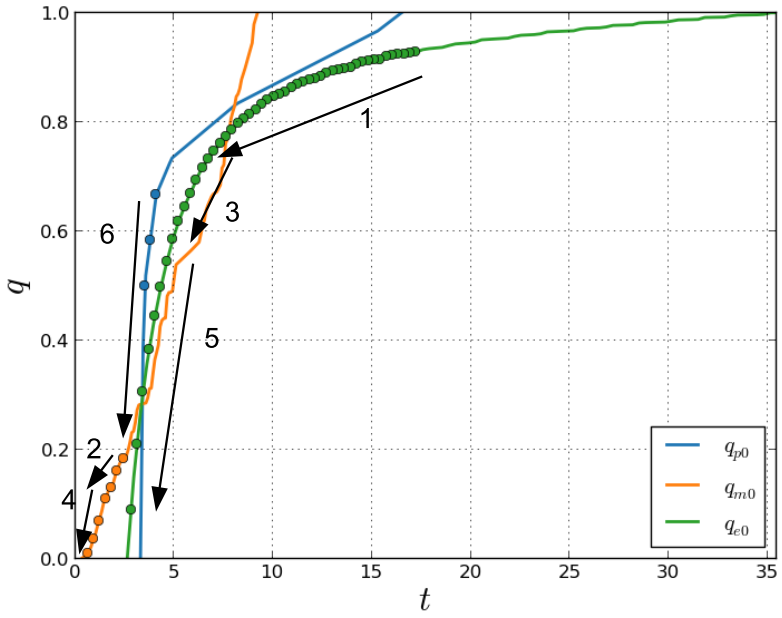


Fig7.18: Variation of Quality according to Project Scheduling for Shorter Deadline $t_d \to 0$

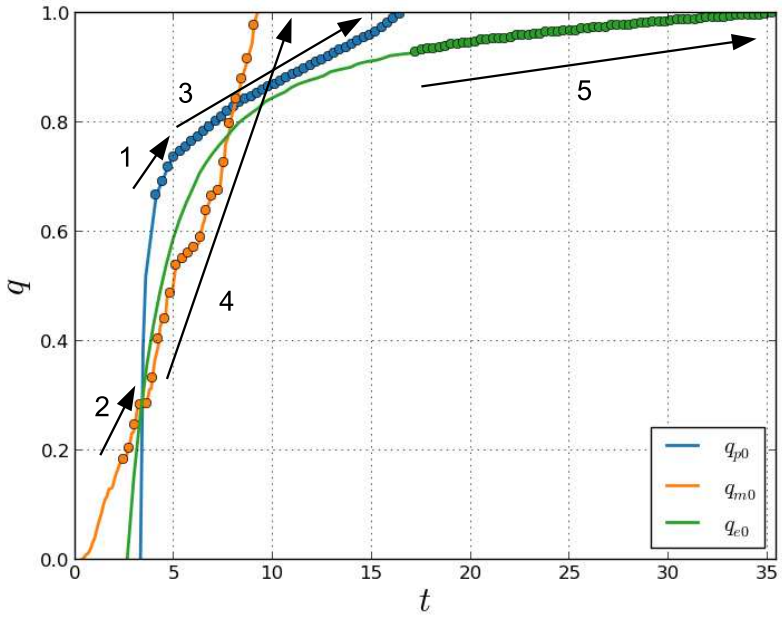


Fig7.19: Variation of Quality according to Project Scheduling for Longer Deadline $t_d \to \infty$

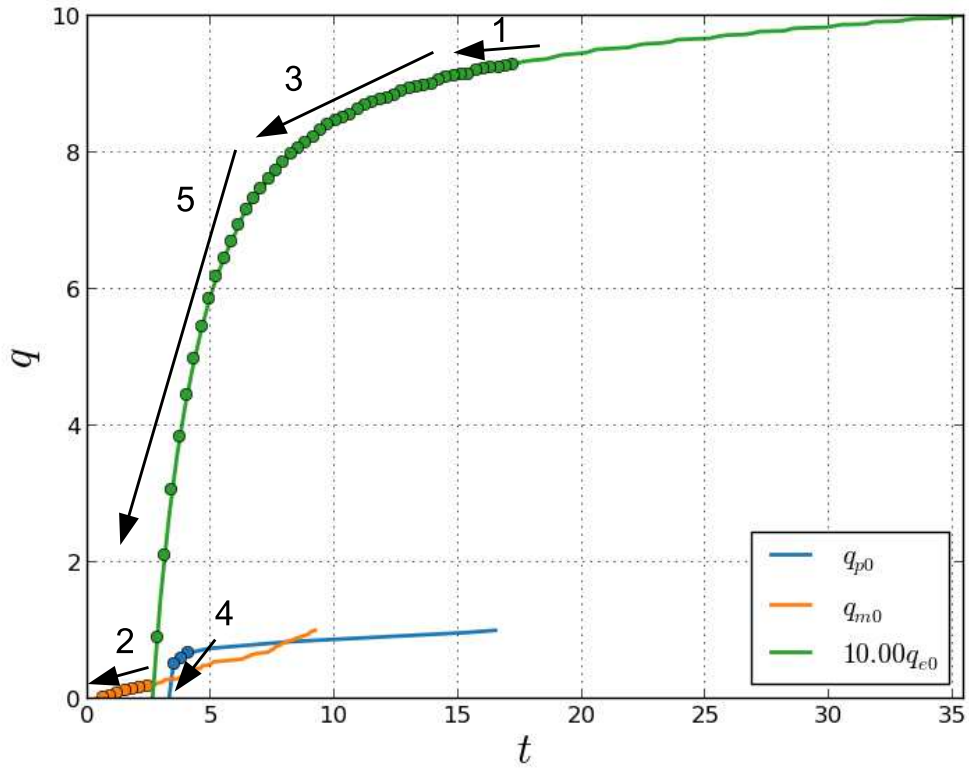


Fig7.20: Variation of Quality according to Project Scheduling for Shorter Deadline $t_d \to 0$ with $w_{E_1} = 10$

7.3.3　移動前後認識実行モデルと 移動前認識実行モデルによるスケジューリングモデル選択実験

本小節では, 移動前後認識実行モデルと 移動前認識実行モデルの 2 つを 利用することで, 適したスケジューリングモデルが選択できることを 実際の品質-時間テーブルを 利用した実験を通じて示す.

第 2.5.1 節, 第 2.5.2 節における 議論から , 以下のように移動前認識実行モデルと 移動前後認識実行モデルの品質 $q_{\text{once}}, q_{\text{twice}}$ および必要時間 $t_{\text{once}}, t_{\text{twice}}$ を 定式化する .

$$t_{\text{once}} = t_{P_0} + t_{M_0} + t_{E_1} \tag{7.18}$$
$$q_{\text{once}} = L(l)\left(w_{P_0} q_{P_0} + w_{M_0} q_{M_0}\right) + w_{E_1} q_{E_1} \tag{7.19}$$
$$t_{\text{twice}} = t_{P_0} + t_{M_0} + t_{P_1} + t_{M_1} + t_{E_1} \tag{7.20}$$
$$q_{\text{twice}} = L(l)\left(w_{P_0} q_{P_0} + w_{M_0} q_{M_0}\right) + w_{P_1} q_{P_1} + w_{M_1} q_{M_1} + w_{E_1} q_{E_1} \tag{7.21}$$

$L(l)$ は距離 l に応じて移動前の品質を減衰させるための項である．$N(l)$ を距離 l が与えられた時に必要となる歩数,k をロボットのハードウェアおよび状態推定を含む歩行中の制御によって決定されるパラメータ, ロボットの一歩あたりの移動量を 200mm とした時, $L(l)$ は以下のように設定した．

$$L(l) = \frac{1}{1 + kN(l)^2} \tag{7.22}$$

$$= \frac{1}{1 + k\left(\frac{L}{0.2}\right)^2} \tag{7.23}$$

$$= \frac{1}{1 + k'l^2} \tag{7.24}$$

ただし, $k' = 0.04k$ として整理した．

これらの式では, 実験の都合上歩行に関するパラメータが変更出来なかったため, q_{E_0} および t_{E_0} を除去したものとして実験を行った．それぞれの実行モデルおよび各種時間のパラメータの関係をタイムライン状に図示したものを **Fig.** 7.21, **Fig.** 7.22 に示す． 移動に伴う品質

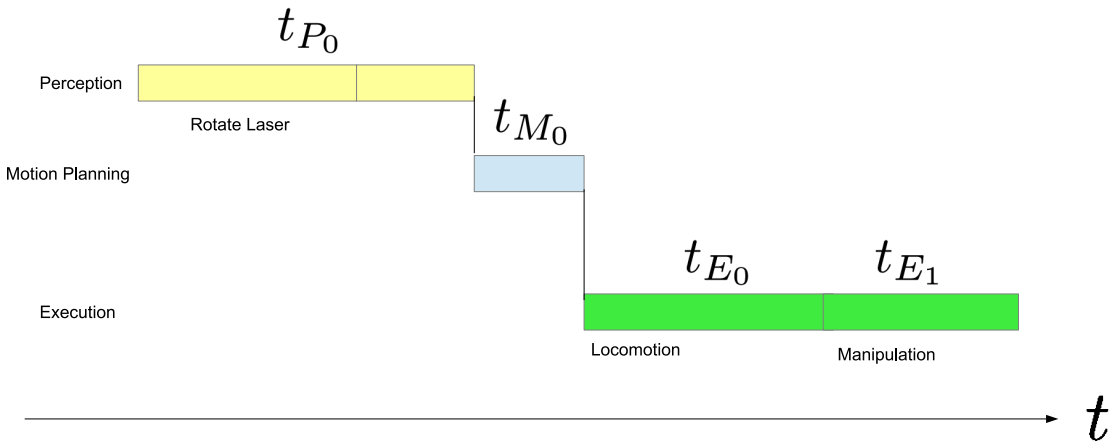


Fig7.21: Task Execution Timeline of Pereption-Before-Traversing Model

の低下を表すパラメータ k' は実験的に 1.0 とした．これは移動量 1.5m によって認識の品質が 0.1 になるようなパラメータである．$l = 0.3$m の時の移動前後認識実行モデルにおいて, 必要時間 t を減少させた時の品質の変化を **Fig.** 7.23 に示す．**Fig.** 7.23 から, プロジェクトスケジューリングの早い段階で q_{P_0} および q_{M_0} の品質が下げられていることがわかる．これは, 移動後の認識および動作計画が最終的なマニピュレーション成功率に大きな役割を占めるため, 相対的に移動前の認識および動作計画の重要性が下がり , それがプロジェクトスケジューリングに反映されていると解釈することができる．

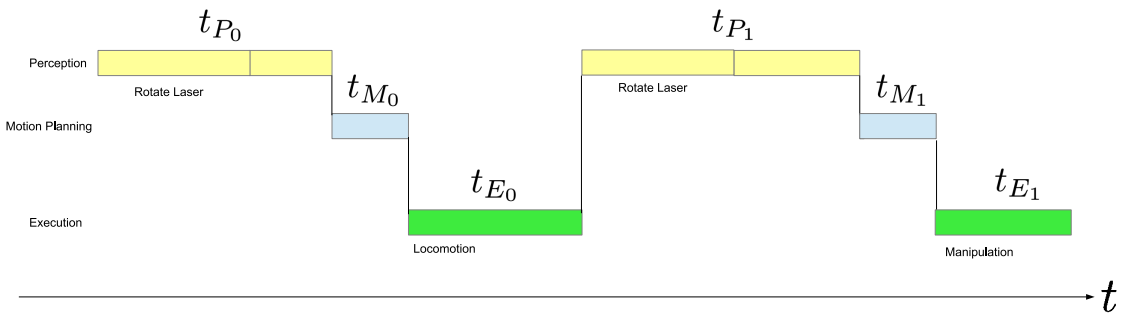


Fig7.22: Task Execution Timeline of Pereption-Before-and-After Traversing Model

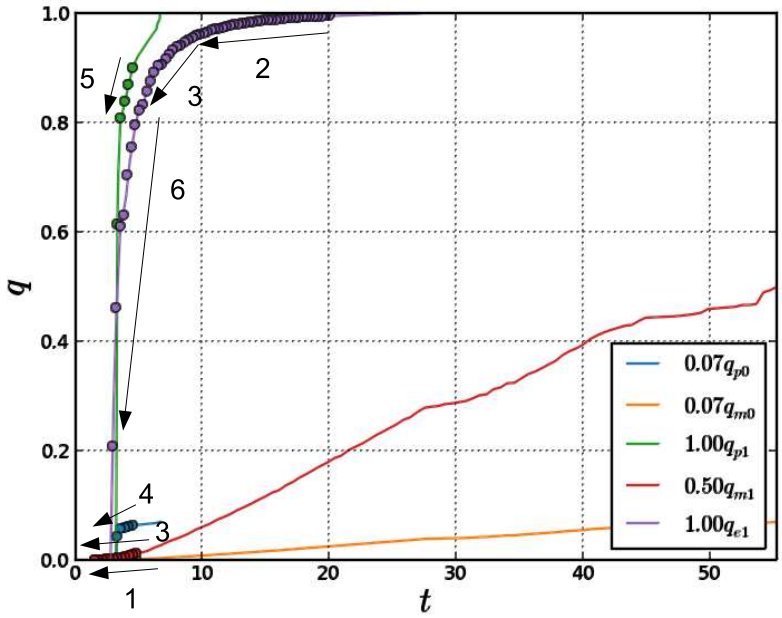


Fig7.23: Variation of Quality q_{twice} according to Project Scheduling for Shorter Deadline of Perception-Before-and-After-Traversing Model with $l = 0.3m$.

移動前後認識実行モデルと移動前認識実行モデルのどちらの実行モデルを利用するかに関しては, 与えられた t_d を満たすようにそれぞれの実行モデルを用いて計算した $q_{\text{once}}, q_{\text{twice}}$ が大きい方を用いる (第 2.5 節). l を変化させた時の $q_{\text{once}}, q_{\text{twice}}$ の挙動について **Fig.** 7.24 に示す. **Fig.** 7.24 における赤色の点は同じ l における $q_{\text{once}}, q_{\text{twice}}$ の交点である. また, この時に利用した各種重み係数 w の値を **Table** 7.2 に示す. 移動前後認識実行モデルを利用するとき, 移動前の認識および動作計画の品質の重み係数は 0.2 と小さな値を利用した. これは, 移動前の認識および動作計画の品質よりも移動後の認識および動作計画の品質を重要視するためで

ある.

Fig. 7.24 より, 距離に従って減少する関数 $L(l)$(**eq.** 7.24) が減少するに従って, つまり移動距離が大きくなるに従って, 移動前認識実行モデルを利用するデッドライン時間 t_{d} が減少していることがわかる (**Fig.** 7.24 矢印). これは, 移動距離が少ない場合においては 2 回認識および計画を行う移動前認識実行モデルよりも 1 回の認識および計画を行う移動前認識実行モデルが優先されて利用されるという事である. 短いデッドライン時間 t_{d} が与えられた時は認識・動作計画が一度である移動前認識実行モデルの方が有利である. 特に見積もり移動距離 l が小さい時は移動前認識実行モデルを利用することによって得られる利点が大きい. **Fig.** 7.24 における $q_{\mathrm{once}}, q_{\mathrm{twice}}$ の交点の挙動はこのような性質を表現しており, 移動距離に従った実行モデル選択が意図したとおりに機能していることが確認できる.

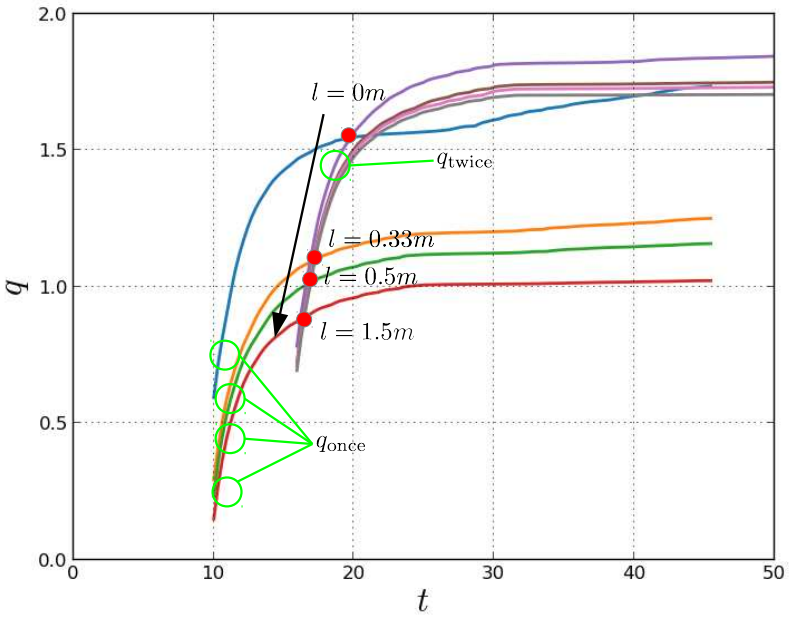


Fig7.24: Comparison of q_{once} and q_{twice} with Different Distances l. Red Points Mean Intersection of q_{once} and q_{twice} with same l.

7.3.4　継続的認識機能による移動中認識実行モデルを利用したスケジューリング実験

第 7.3.3 節において用いた移動前後認識実行モデルと移動前認識実行モデルの 2 つの実行モデルでは, 要求時間 t_{d} が小さい場合, 認識や動作計画に割り当てる時間が少なくなり, タスク

Table7.2: Weight Parameters for Experiment of Switching Execution Model (**Fig.** 7.24)

Execution Model	w_{P_0}	w_{M_0}	w_{P_1}	w_{M_1}	w_{E_1}
Once Perception	1.0	1.0	N/A	N/A	1.0
Twice Perception	0.2	0.2	1.0	0.5	1.0

遂行が困難になるほど品質が下がってしまう. このような条件下では, 移動中認識実行モデル (第 2.5.3 節) を利用して継続的にマニピュレーション対象を認識し, 移動中に動作計画を行うことで全体の品質を高めるアプローチが有効である. しかしながら, 継続的認識機能を実現するトラッキングによる対象物の位置推定は, オクルージョンやトラッキング中の照明条件の変化などによってトラッキングに失敗する可能性が高く, 信頼性の低い認識機能であるとみなすことができる.

移動中認識実行モデルおよび品質を以下のようにモデル化する (第 2.5.3 節).

$$t_{\text{continuous}} = t_P + t_{M_0} + t_{E_1} \tag{7.25}$$

$$q_{\text{continuous}} = w_P q_P + w_{M_1} q_{M_1} + w_{E_1} q_{E_1} \tag{7.26}$$

このとき, 必要時間 $t_{\text{continuous}}$ には M_0 に必要時間が含まれているが, 品質 q にはその品質が含まれていない. というのも, M_0 は移動の目標値を決定するような動作計画であるが, これは歩行中に更新されるからである. このため, t_{M_0} は可能な限り最小なものが常に選択される. 一方 M_1 は品質 q には含まれているが, 必要時間 t には含まれていない. そのため,M_1 は常にもっとも品質の高くなるパラメータが選ばれてしまう. これは歩行時間を超過してしまうパラメータが選ばれる可能性があることを意味している. そのため, 本実験ではシステム実装者が与えた初期値を常に利用するとする. このパラメータは $t_{M_1} = 2.42\text{sec}$ である. **Fig.** 7.25 に短いデッドライン時間 t_{d} を与えた時のパラメータの変化を示す.

本実験では重み係数 w_P および w_{E_1} は $w_P = 0.5, w_{E_1} = 0.5$ とした. この値は,$l = 1.5\text{m}$ の時に移動前後認識実行モデル, 移動前認識実行モデル, 移動中認識実行モデルの 3 つのスケジューリングモデルが一点で交わるように設定した (**Fig.** 7.26). これは,$l > 1.5\text{m}$ においては, 移動前認識実行モデルよりも移動中認識実行モデルが優先されることを意図したものである. **Fig.** 7.26 より, $l > 1.5\text{m}$ において $q_{\text{continuous}} > q_{\text{once}}$ となるため, 意図したようにスケジューリングができていることがわかる. **Fig.** 7.28 に $t_{\text{d}} = 10.0$ とした時の実機の挙動についてまとめた. また, このときの点群追跡結果を **Fig.** 7.27 に示す. このとき, $t_P = 0.14\text{sec}, t_{E_1} = 9.7\text{sec}$ とスケジューリングされた. $t_{M_1} = 2.42\text{sec}$ であったが, 第 7.2.1

節では最悪計算時間を用いて見積もっていた．実機実験では 1 回目の計画では 1.2sec, 2 回目以降は 0.2sec となった (**Fig.** 7.28)．これは 2 回目以降の計画では立ち位置を変更することなく逆運動学解を導出することが可能であったため, 計算コストが少なく実行できたためである．

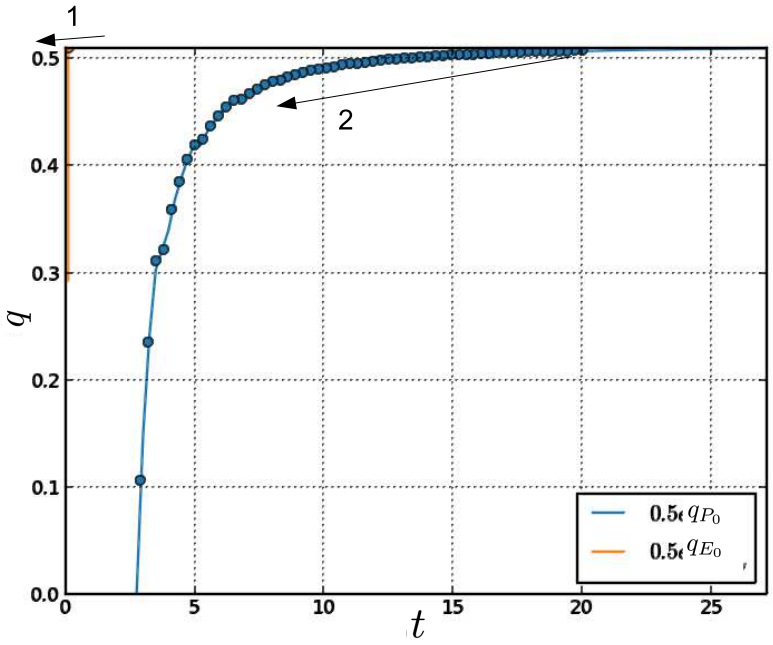


Fig7.25: Variation of Quality $q_{\text{continuous}}$ according to Project Scheduling for Shorter Deadline of Perception-During-Traversing Model

7.4 おわりに

本章では第 2 章で述べた評価制御機構による認識, 動作計画, 動作実行の必要時間スケジューリングを実機ロボット上に実装し評価実験を行った．タスク成功率に影響を与えるものを品質として用いた．認識では 3 次元点群の低解像度化, 動作計画では衝突計算およびマニピュレーション軌道の詳細度, 実行部においては動作速度が与える安定性への影響を考慮して品質-時間テーブルを構築した．提案手法により構築した品質-時間テーブルを利用することで, 必要時間に応じたパラメータ調整が可能であることを実験を通じて示し, その有効性を確認した．また, 移動を伴うマニピュレーションプロジェクトにおいて, 移動による品質低下を考慮することで第 2 章で述べた 3 種類の実行モデルから適した実行モデルが選択可能であることを実験によって示した．

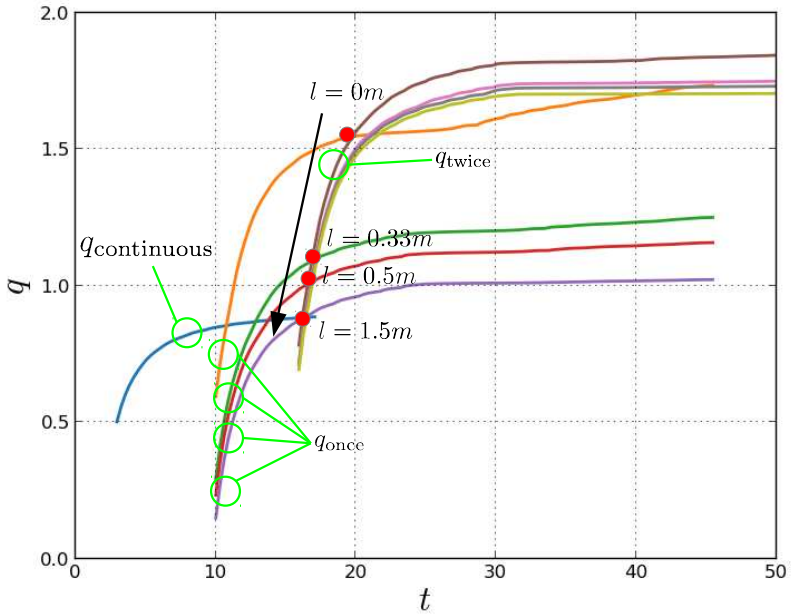
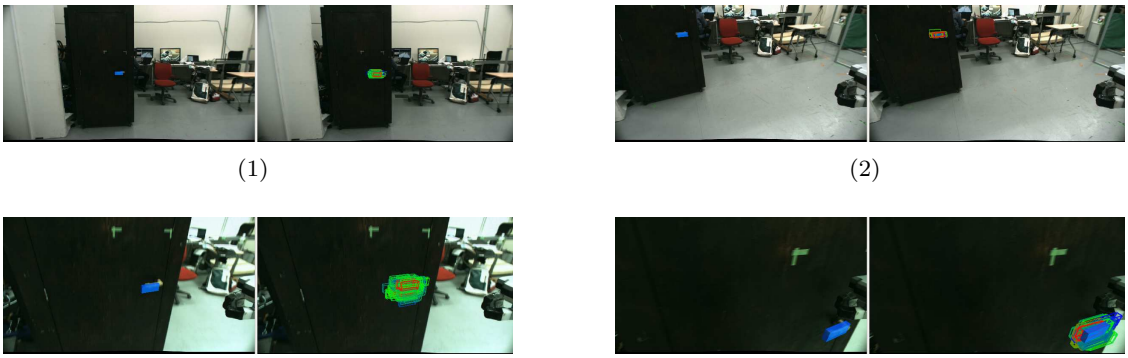


Fig7.26: Comparison of q_{once}, q_{twice} and $q_{\text{continuous}}$ with Different Distances l.



(1)　(2)

(3)　(4)

Fig7.27: Tracking Door Handle using PointCloud. Left) Blue Cuboid Shows Result of Tracking. Right) Lines Visualizes Top 10% Particles. Red Cuboids Means Larger Weights.

7.4.1　災害対応競技会における遠隔操縦システムと評価制御機構によるシステム構成の比較と考察

第 3 章で述べた災害対応競技会における遠隔操縦システムを本章で述べた評価制御機構によるスケジューリング機能と比較することで考察を行う .

Fig. 7.29 に第 3 章で述べた遠隔操縦システムを利用した場合の詳細なオペレータ及び認

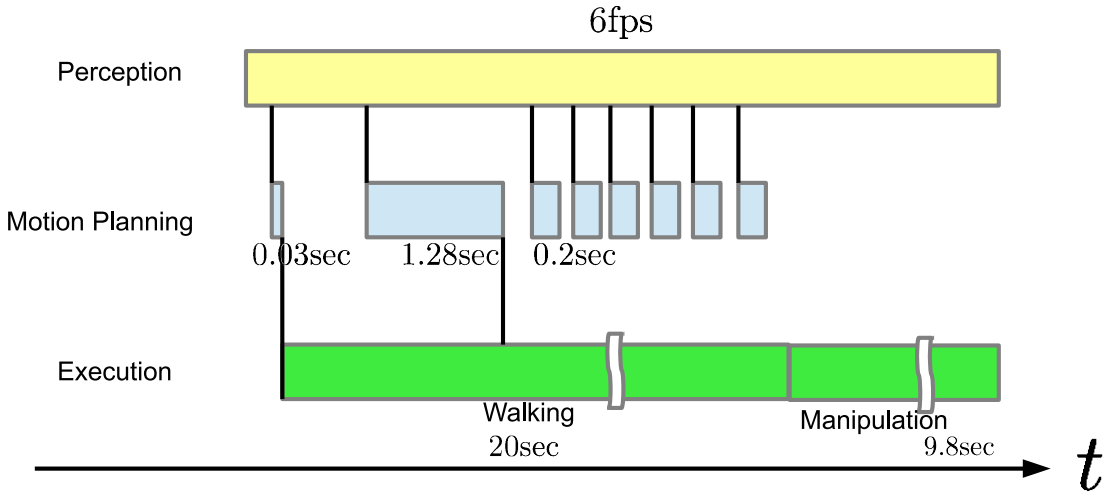


Fig7.28: Task Execution Timeline under $t_{\mathrm{d}} = 10.0$

識・動作計画・実行に必要となる時間を図示したものを示す．遠隔操縦システムを利用する際

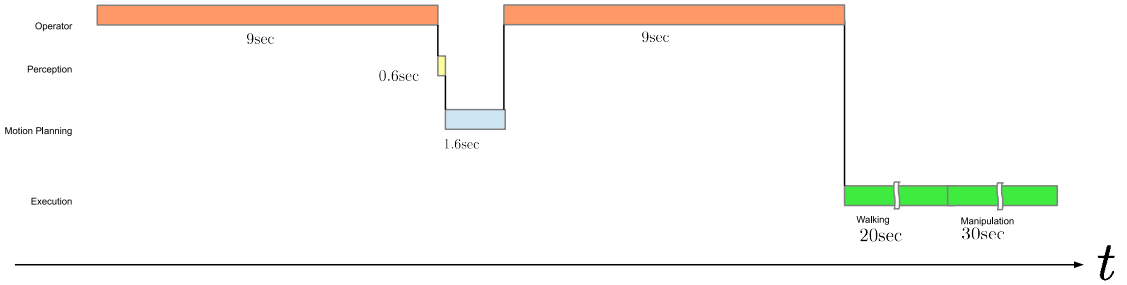


Fig7.29: Task Execution Timeline of Teleoperation Implementation

の特徴は, オペレータによる操作時間がシステムの実行中に介入することである．その結果, オペレータの操作時間のためにシステム全体の実行時間は伸びてしまう．その一方で, オペレータの操作によって認識器に手がかりを与えたり , 動作計画に対して確認を行うことができるためそれらの計算時間を短縮することができる．しかしながら , **Fig.** 7.29 に示した実験結果では人による操作が合計 18 秒発生してしまっている．与えられたデッドライン時間が長い場合, オペレータは十分に時間をかけて慎重に操作することができるが, デッドライン時間が短い場合では対応が難しい．一方評価制御機構によるデッドライン時間に従ったパラメータ制御では **Table** 7.1 に示したようにデッドライン時間が短い場合においてシステム全体の消費時間を調整可能である．このような特性を考えると , 十分にデッドライン時間が長い場合はオペレータの操作を利用することで遠隔操縦システムの利点を取り込むことが可能である．

7.4.2　ハードリアルタイムなスケジューリングシステムへの拡張に関する考察

ロボットのタスクがデッドライン時間よりも長い時間がかかると価値がなくなるようなハードリアルタイムな評価を受ける場合, 本論文で示したプロジェクト スケジューリング手法は機能しない場合がある. というのも, 本論文で示したプロジェクト スケジューリング手法は, 見積もり時間を利用してデッドライン時間全てを利用するような手法であったため, 実行時の環境の状態によっては見積もり時間よりも長い時間が必要となることがある. このような問題に対応するためには, 本論文のプロジェクト スケジューリング手法を拡張し, 認識や計画を複数回より短いデッドライン時間を用いて実行することでハードリアルタイムなデッドラインを満たすシステムを構築することが可能である.

また, 精度が制御目標となるようなフィードバックループである Active Vision をタスク要素に用いることで, 認識や計画に関してはより見積もり時間を効果的に利用することが可能である.

7.4.3　実行時のタスク経験に応じたパラメータ獲得可能なシステム構成に関する考察

本章で述べた評価制御機構において, 実行時のパラメータ制御の挙動に影響を与える要素は以下の 2 つである.

1. 品質-時間の関係を表すテーブル
2. 品質 q を求めるため, 各要素の品質を結合する際の重みパラメータ w

前者に関しては本章では事前に品質のモデル化を行い, 必要時間を計測することで品質-時間テーブルをシステムに与えた. 後者に関してはシステム実装者が事前にシステムに与えていた. これらは実行時のタスク経験に基づいて修正を行うことが可能である. 特に前者に関しては, 計算に必要となる時間は環境の状態に強い影響を受けるため, 実行時の必要時間を品質-時間テーブルに反映させることで, より精度の良い予測が可能となる.

第 8 章

結論

本論文は等身大ヒューマノイドロボットのための統合ロボットシステムを構築する際に, 要求される時間に従って, システムが自動的に品質を保持しながら認識・動作計画・実行に費やす時間を制御可能な機能をロボットシステムに持たせるためのシステム構成法を主題に取り組んだものであった.

第 1 章「序論」において研究背景としてロボットシステムの研究について述べ, 要求される時間に従って, システムが自動的に品質を保持しながら認識・動作計画・実行に費やす時間を制御可能な機能をロボットシステムに持たせることの重要性について述べた. また, 研究背景として関連研究との対比を行い本論文の位置づけを行った.

第 2 章「認識計画実行機能の評価制御機構に基づくプロジェクトスケジューリング」ではロボットシステムにおいて認識, 動作計画, 動作実行のパラメータを変更することで, 品質と必要時間を制御可能な仕組みとして評価制御機構を導入した. 評価制御機構の特徴はタスク実行中の認識や動作計画, 動作実行において精度や信頼性に影響を与えるパラメータを制御することでタスク遂行に必要となる時間を, 与えられた遂行目標時間に対してスケジューリングする機構であった. これは認識, 動作計画および動作実行において計算量や精度を制御可能なパラメータを持つものを利用し, パラメータに関して必要となる計算時間や実行時間および品質の関係を品質-時間テーブルとして予め保持することで可能となるシステムであった. また、移動を伴うマニピュレーションプロジェクトにおいては, 実行時のスケジューリングモデルとして移動前にのみ認識を行う実行モデル, 移動前後で認識を行う実行モデル, そして移動中に認識を継続して行う 3 種類の実行モデル, および特殊な実行モデルとして認識器と動作計画器の協調的サブスケジューリングモデルを定義した.

第 3 章「遠隔操縦ロボットシステムのためのソフトウェア環境」では遠隔操縦型のロボット

システムにおいてユーザインタフェイスを中心としてシステムをどのように設計するべきであるかについて災害対応ロボット競技会を題材として述べた．遠隔操縦システムでは予期しない問題が発生した場合においても対応可能なシステム構成が重要であった．人がフィードバックループに常に介在するような Human-in-the-loop 型の遠隔操縦ではなく, 人が時々ロボットにヒントをあたえ, 問題が発生しない限りにおいてはロボットが認識・計画・実行を自律的に行うことが可能な Supervised Autonomy 型のシステム構成を用いた．自律的に動くシステムを基本とし, システムの自律性の抽象度にしたがってオペレータが介入可能なユーザインタフェイスを用いることで, 遠隔操縦において予期しない状況でも復帰可能なシステム構成となっていることを実際の競技会を通じて示した．また, 競技会での時間経過について分析することで, 競技会ではオペレータが必要以上に慎重になったために全てのタスクを完遂出来なかったことから, システムがタスク遂行に必要となる時間を制御するシステムが重要であることを指摘した．

第 4 章「**認識器と動作計画器の協調的サブスケジューリングによる足配置計画法**」では足配置計画法を題材にし, 動作計画と認識器を協調的に動作させることで高速に動作計画を行う手法について提案した．動作計画器と認識器を協調的に動作させることで, 認識器は動作計画が必要としている領域を知ることができ, それを注視領域として必要となる認識処理を削減することが可能であった．さらに環境の多くが連続した平面で構成されるという環境平面仮説を導入することで, 認識処理を削減し, 探索を高速化することが可能であった．協調的スケジューリングおよび環境平面仮説によって足配置計画法の高速化手法を実験を通じて有効性を確認した．

第 5 章「**継続的認識機能のための 3 次元点群追跡による物体認識器**」では 3 次元点群を利用したパーティクルフィルタによる実時間物体追跡手法について述べた．継続的認識機能は移動中に認識を行いながら認識処理や動作計画に必要となる時間を短縮する実行モデルのために重要な機能であった．3 次元点群処理は計算量が大きく実時間処理は難しいため, 並列計算が容易なパーティクルフィルタを用いた実時間での物体追跡を可能とするための最適化を行い, 物体追跡実験を通じてその有効性を確認した．また, 精度と計算時間および高速化のためのパラメータの関係について静的な環境での実験を通じて示した．

第 6 章「**環境およびロボットの状態監視のための常時稼働型ソフトウェア**」では高次の意思決定機能となるタスク計画器を利用した実行時の監視機能について提案し, マニピュレーション実験を通して人の割り込みによる環境の変化に対応可能であることを示した．タスク計画器

は古典的な AI として知られるシンボリックな記述に従ってロボットの行動手順を計画するものである．ロボットはこの計画された行動手順にしたがってタスクを遂行していくが, 実行時に発生するエラーに対して素早く対応することは難しくなる．本論文ではこの環境への即応性を解決するため, 大域的・局所的なフィードバックという 2 つのフィードバックを利用した．局所的なフィードバックは第 5 章で述べた実時間 3 次元点群追跡器による継続的認識機能を利用したビジュアルフィードバックであった．大域的なフィードバックはシンボリックな記述によって得られる行動の前後および実行中に成立すべき条件を継続的認識機能利用することで監視し実現した．

第 7 章「評価制御機構に基づくプロジェクトスケジューリングの実装と評価」では評価制御機構における品質-時間テーブルの構築法について述べ, 品質-時間テーブルを利用したスケジューリング手法について述べた．第 7 章では具体的な品質として, タスク成功率を規範とした指標を用いた．認識では 3 次元点群の低解像度化, 動作計画では衝突計算およびマニピュレーション軌道の詳細度, 実行部においては動作速度が与えるバランス制御の安定性への影響を考慮することで、タスク成功率を表現する品質-時間テーブルを構築した．提案手法により構築した品質-時間テーブルを利用することで, 必要時間に応じたパラメータ調整が可能であることを実験を通じて示し, その有効性を確認した．また, 移動を伴うマニピュレーションプロジェクトにおいて, 移動による品質低下を考慮することで第 2 章で述べた 3 種類の実行モデルから適した実行モデルが選択可能であることを実験によって示した．

本論文の成果は以下のようにまとめられる．活動環境が固定されないため計算時間を必要とする高度な認識および動作計画が必要とされるヒューマノイドの統合システムの認識, 動作計画および実行に関して, 必要時間とトレードオフの関係となる精度や詳細度といった品質パラメータを導入し, システムが品質と時間の関係性を保持することに基づいたプロジェクトスケジューリング機構を用いることで, 従来はシステム実装者が注意深く調節していたパラメータを, 人間は必要となる時間を指示するのみでロボットのタスク遂行時間や各構成要素のパラメータを自動的に構成し, 与えられた時間内で品質を保証する制御機構を示し, 災害対応用タスクでの既存システムとの比較を通じて実際に認識計画実行機能の評価制御機構の効果を示すことで本提案の有効性を確認した．本論文は求められる要求度緊急度が異なるタスクに対応する基盤となるシステムを人間がその機構を利用して要求に応じたシステムをカスタマイズ可能であることを示したのみでなく, システム自体がシステムの運用履歴を通してそのカスタマイズの方法を導入することで適応的・学習的に要求タスクへ対応するロボットシステムへの基盤

となるものとしての可能性を示したものとなっている.

謝辞

本論文は筆者が東京大学情報理工学系研究科創造情報学専攻に在学中, 稲葉雅幸教授の御指導のもとで行なった博士論文である.

稲葉雅幸教授には学部 4 年生の頃から 6 年間にわたりご指導頂きました. 小さくまとまりそうになりがちな筆者に対して広い視点から研究の方向性を示してくださいましたし, 研究の厳しさも教えてくださいました. 先生に教えて頂いたことを今後の活動にもいかしていきたいと思います. ありがとうございました.

石川正俊教授, 國吉康夫教授, 山西健司教授, 千葉滋教授には本論文をまとめるにあたり非常に有意義なご意見を頂きました. 各研究分野で, 第一線に立たれる先生方に論文を見て頂くことになり, おそれ多くも身の引き締まる思いでした.

岡田慧准教授には直接指導頂く機会がおおく, 気にかけて頂き時折ハイセンスなアドバイスを下さいました. 先生の技術に対する意識の高さからは学ぶところが多くありました. 筆者のソフトウェアに関する技術は全て岡田慧から学んだと思っております.

山口技官には事務処理にあたって研究活動を助けて頂きました.

矢口講師, 垣内講師, 野沢講師, 菅井助教, 長濱助教には相談する機会も多く, 研究全般にわたって様々なアドバイスを下さいました.

その他先に卒業された方も含めて研究室で共に過ごしたみなさんや, 研究室外で学会や創造輪講等で研究に対する意見をくださった方々, 精神的に支えてくれた人たち, 近くで応援しれくれた人, 多くの人に助けてもらうことでこの論文を書くことができました. みなさまにお礼申し上げます.

最後に筆者の希望や自由な行動を理解し, 長い学生生活を最後まで支えてくださった両親に感謝いたします.

2015 年 12 月 4 日　植田　亮平

発表文献

(1) 植田亮平, 小倉崇, 石坂唯, 林摩梨花, 吉海智晃, 稲葉雅幸. Eusdyna における PhysX を用いたヒューマノイドの柔軟外装のモデリングと柔軟物の操作行動生成. 第 25 回日本ロボット学会学術講演会講演論文集, pp.1L39, 2007

(2) 林摩梨花, 石坂唯, 植田亮平, 吉海智晃, 稲葉雅幸. 全身分布三軸力覚センサを有する柔軟外装ロボットの密着インタラクション. 第 25 回日本ロボット学会学術講演会講演論文集, pp.2L13, 2007

(3) Ryohei Ueda, Takashi Ogura, Kei Okada, Masayuki Inaba: Design and Implementation of Humanoid Programming System Powered by Deformable Objects Simulation, in Proceedings of the 10th International Conference on Intelligent Autonomous Systems, pp.374–381, 2008.

(4) Marika Hayashi, Ryohei Ueda, Tomoaki Yoshikai, Masayuki Inaba: A Fall Dwon Resistant Humanoid Robot with Soft Cover and Automatically Recoverable Mechanical Overload Protection, in Proceedings of 11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, 2008.

(5) 植田亮平, Zaoputra Nikolaus, 西野環, 矢口裕明, 山崎公俊, 花井亮, 岡田慧, 稲葉雅幸: 車載視覚計測システムを利用した Gloogle Earth の 3D 環境モデリング, in 日本機械学会ロボティクス・メカトロニクス講演会'08 講演論文集, 2P1-G11, 2008.

(6) 植田亮平, 小倉崇, 神崎秀, 山崎公俊, 岡田慧, 稲葉雅幸: 柔軟物シミュレーションを利用した布団保持姿勢の獲得, in 日本機械学会ロボティクス・メカトロニクス講演会'08 講演論文集, 2A1-D17, 2008.

(7) 小堀浩子, 植田亮平, 得津覚, 岡田慧, 稲葉雅幸: ヒューマノイドによる洗濯物展開操作における柔軟物認識行動制御システムの実現, in 第9 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, pp.743–744, 2008.

(8) 山崎 公俊, 植田 亮平, 野沢 峻一, 森 優人, 槙 俊明, 畑尾 直孝, 岡田 慧, 稲葉 雅幸: 掃除片付け作業をこなす生活支援ロボットのための認識行動システム実証研究, in 第 14 回ロボティクスシンポジア予稿集, pp.522–527, 2009.

(9) Kimitoshi Yamazaki, Ryohei Ueda, Shunichi Nozawa, Yuto Mori, Toshiaki Maki, Naotaka Hatao, Kei Okada, Masayuki Inaba: A Demonstrative Research for Daily Assistive Robots on Tasks of Cleaning and Tidying up Rooms, in First International

Symposium on Quality of Life Technology, J, 2009.

(10) 吉海 智晃, 林 摩梨花, 門脇 明日香, 植田 亮平, 稲葉 雅幸: 柔軟センサ肉質外装と自動復帰可能な関節過負荷保護機構を備えたヒューマノイドの設計と開発, in 日本機械学会ロボティクス・メカトロニクス講演会'09 講演論文集, 2A1-E04, 2009.

(11) 伊東 信之, 植田 亮平, 浦田 順一, 中西 雄飛, 岡田 慧, 水内 郁夫, 稲葉 雅幸: 滑り状態制御のためのヒューマノイドハンドの設計と実現, in 日本機械学会ロボティクス・メカトロニクス講演会'09 講演論文集, 2A2-C06, 2009.

(12) 植田 亮平, 野沢 俊一, 森 優人, 山崎 公俊, 岡田 慧, 稲葉 雅幸: ロボットによる家事支援タスクシーケンスにおける失敗検知回復の構造記述, in 日本機械学会ロボティクス・メカトロニクス講演会'09 講演論文集, 1P1-D04, 2009.

(13) 野沢峻一, 植田亮平, 槙俊明, 岡田慧, 稲葉雅幸: ヒューマノイドによる未知重量対象物の持ち上げ動作のための力・運動量制御法, in 第 27 回日本ロボット学会学術講演会講演論文集, IS1-04, 2009.

(14) 岡田 慧, 野沢 俊一, 植田 亮平, 渡辺 義明, 得津 覚, 稲葉 雅幸: 失敗からの復帰が可能な認識行動タスクの高次プランニングシステムにおける感覚に基づく動作の修正, in 第 27 回日本ロボット学会学術講演会講演論文集, 1L3-04, 2009.

(15) 山崎公俊, 野沢峻一, 植田亮平, 槙俊明, 森優人, 岡田慧, 松本潔, 稲葉雅幸: 日用品データベースを利用する家事支援ロボットによる思い出し・片付け支援, in 第 27 回日本ロボット学会学術講演会講演論文集, 2E2-05, 2009.

(16) 長濱虎太郎, 野沢峻一, 植田亮平, 岡田慧, 稲葉雅幸: ヒューマノイドの生活支援行動タスク模倣における作用点認識と全身行動生成, in 第 27 回日本ロボット学会学術講演会講演論文集, 1S1-05, 2009.

(17) 植田亮平, 野沢峻一, 岡田慧, 稲葉雅幸: 等身大ヒューマノイドのスライド動作による物体移動戦略を特徴とする全身動作と視点位置の計画法, in 第 27 回日本ロボット学会学術講演会講演論文集, C1S2-07, 2009.

(18) Shunichi Nozawa, Ryohei Ueda, Youhei Kakiuchi, Kei Okada, Masayuki Inaba: A Full-Body Motion Control Method for a Humanoid Robot based on On-Line Estimation of the Operational Force of an Object with an Unknown Weight, in Proceedings of The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.2684–2691, 2010.

(19) Yohei Kakiuchi, Ryohei Ueda, Kazuya Kobayashi, Kei Okada, Masayuki Inaba: Working with Movable Obstacles Using On-line Environment Perception Reconstruction Using Active Sensing and Color Range Sensor, in Proceedings of The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1696–1701, 2010.

(20) Kimitoshi Yamazaki, Ryohei Ueda, Shunichi Nozawa, Yuto Mori, Toshiaki Maki, Naotaka Hatao, Kei Okada, Masayuki Inaba: System Integration of a Daily Assistive Robot and Its Application to Tidying and Cleaning Rooms, in Proceedings of The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1365–1371, 2010.

(21) 野沢 峻一, 植田 亮平, 垣内 洋平, 岡田 慧 and 稲葉 雅幸: ヒューマノイドにおける未知重量対象物への操作戦略決定法に基づく 全身動作制御法, in 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, 2P1-C21, 2010.

(22) 垣内 洋平, 植田 亮平, 岡田 慧, 稲葉 雅幸: カラー距離画像センサと能動的センシングを用いた可動障害物の認識と環境操作を伴う行動の実現, in 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, 2A2-E06, 2010.

(23) 植田 亮平, 野沢 峻一, 伊藤 司, 山崎 公俊, 岡田 慧, 稲葉 雅幸: 等身大ヒューマノイドの状態表現に基づく失敗検知復帰ルール記述可能な行動動作システム, in 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, 2A2-E29, 2010.

(24) 植田 亮平, 野沢 峻一, 岡田 慧, 稲葉 雅幸: 計算機クラスタを利用したオンライン動力学シミュレーションによる失敗時の復帰困難性に着目した動作評価を含んだ動作計画法, in 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, 1P1-E12, 2010.

(25) 伊藤 司, 植田 亮平, 垣内 洋平, 岡田 慧, 稲葉 雅幸: 等身大ヒューマノイドの片付け行動におけるカラー距離センサによる物体の把持形状認識と対象形状を考慮した把持動作生成, in 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, 2A2-C01, 2010.

(26) 植田亮平, 垣内洋平, 斉藤学, 伊藤司, 岡田慧, 稲葉雅幸: 非同期タスク監視機構によるタスク実行と再計画システム, in 第 28 回日本ロボット 学会学術講演会講演論文集, 1A1-1, 2010.

(27) 石田正穂, 野沢峻一, 植田亮平, 岡田慧, 稲葉雅幸: 可達域マップを用いた足配置決定法

によるヒューマノイドの全身拾い上げ動作の実現, in 第 28 回日本ロボット学会学術講演会講演論文集, 1A1-2, 2010.

(28) 伊藤司, 植田亮平, 垣内洋平, 岡田慧, 稲葉雅幸: ヒューマノイドにおける視覚と指先触覚を利用した未知対象物体把持のための行動選択, in 第 28 回日本ロボット学会学術講演会講演論文集, 1O3-5, 2010.

(29) 津田敦史, 垣内洋平, 野沢峻一, 植田 亮平, 伊藤 司, 岡田慧, 稲葉雅幸: ヒューマノイドによる双腕物体操作を用いた 3 次元形状テクスチャモデルの自動取得, in 第 11 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, 1G2-5, 2010.

(30) Yohei Kakiuchi, Ryohei Ueda, Kei Okada, Masayuki Inaba: Creating Household Environment Map for Environment Manipulation Using Color Range Sensors on Environment and Robot, in Proceedings of The 2011 IEEE International Conference on Robotics and Automation, pp.305–310, 2011.

(31) Shunichi Nozawa, Masaho Ishida, Ryohei Ueda, Youhei Kakiuchi, Kei Okada, Masayuki Inaba: Full-Body Motion Control Integrated with Force Error Detection for Wheelchair Support, in Proceedings of the 2011 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2011), pp.193–198, 2011.

(32) Wilma Bainbridge, Shunichi Nozawa, Ryohei Ueda, Kei Okada, Masayuki Inaba: Robot Sensor Data as a Means to Measure Human Reactions to an Interaction, in Proceedings of the 2011 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2011), pp.452–457, 2011.

(33) Atsushi Tsuda, Yohei Kakiuchi, Ryohei Ueda, Shunichi Nozawa, Kei Okada, Masayuki Inaba: Grasp, Motion, View Planning on Dual-arm Humanoid for Manipulating In-Hand Object, in IEEE Workshop on Advanced Robotics and Its Social Impacts, 2011.

(34) Ryohei Ueda, Yohei Kakiuchi, Shunichi Nozawa, Kei Okada, Masayuki Inaba: Anytime Error Recovery by Integrating Local and Global Feedback with Monitoring Task States, in Proc. of the 15th International Conference on Advanced Robotics, TuII2.3, 2011.

(35) Wilma BAINBRIDGE, Shunichi NOZAWA, Ryohei UEDA, Yohei KAKIUCHI, Kotaro NAGAHAMA, Kei OKADA, Masayuki INABA: Understanding expectations

of a robot's identity through multi-user interactions, in Proceedings of the HRI 2011 Workshop on Exepctations in intuitive human-robot interaction, , 2011.

(36) 野沢 峻一, 石田 正穂, 植田 亮平, 岡田 慧, 稲葉 雅幸: 関節負荷・可操作範囲に基づくヒューマノイドの物体操作戦略選択法, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2P2-J06, 2011.

(37) 津田 敦史, 野沢 峻一, 植田 亮平, 垣内 洋平, 岡田 慧, 稲葉 雅幸: ヒューマノイドにおける複雑形状物品の双腕物体操作による見かけ認識モデル獲得, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2P2-J08, 2011.

(38) 植田 亮平, 垣内 洋平, 野沢 峻一, 岡田 慧, 稲葉 雅幸: 3 次元点群形状位置合わせによる視覚フィードバックのタスクレベル障害復帰システムへの統合, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2P2-J04, 2011.

(39) 垣内 洋平, 植田 亮平, 伊藤 司, 岡田 慧, 稲葉 雅幸: 三次元視覚を用いた生活環境認識による生活支援行動フローの作成, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2P2-J09, 2011.

(40) 石田 正穂, 野沢 峻一, 植田 亮平, 岡田 慧, 稲葉 雅幸: ヒューマノイドによる全身動作生成のための足配置計画および床上物体認識に関する研究, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2P2-J07, 2011.

(41) ベインブリッジ・ウィルマ, 野沢峻一, 植田 亮平, 垣内 洋平, 岡田 慧, 稲葉 雅幸: バイオフィードバックに基づく人間・ロボット交流実験の分析, in 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, 2A1-C04, 2011.

(42) 津田 敦史, 野沢 峻一, 植田 亮平, 垣内 洋平, 岡田 慧, 稲葉 雅幸: ヒューマノイドによる把持物体の双腕見回し操作による 3 次元見かけ認識モデル構築のための最良把持選択, in 第 12 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, 1C1-8, 2011.

(43) M. Inaba, K. Okada, T. Yoshikai, R. Hanai, K. Yamazaki, Y. Nakanish, H. Yaguchi, N. Hatao, J. Fujimoto, M. Kojima, S. Tokutsu, K. Yamamoto, Y. Kakiuchi, T. Maki, S. Nozawa, R. Ueda, I. Mizuuchi: Enhanced Mother Environment with Humanoid Specialization in IRT Robot Systems, in Cedric Pradalier and Roland Siegwart and Gerhard Hirzinger (Eds.): Robotics Research: The 14th International Symposium ISRR, pp.379–396, Springer, 2011.

(44) Shunichi Nozawa, Ryohei Ueda, Yohei Kakiuchi, Kei Okada, Masayuki Inaba:

Sensor-Based Integration of Full-Body Object Manipulation Based On Strategy Selection in a Life-Sized Humanoid Robot, Journal of Robotics and Mechatronics, Vol.23, No.2, pp.239–248, 2011.

(45) Kimitoshi Yamazaki, Ryohei Ueda, Shunichi Nozawa, Mitsuharu Kojima, Kei Okada, Kiyoshi Matsumoto, Masaru Ishikawa, Isao Shimoyama, Masayuki Inaba: Home-Assistant Robot for an Aging Society, Proceedigns of The IEEE, Centennial Year, Special Issue, Quality of Life Technology, Vol.100, No.8, pp.2429–2441, 2012.

(46) Wilma BAINBRIDGE, Shunichi NOZAWA, Ryohei UEDA, Kei OKADA, Masayuki INABA: A Methodological Outline and Utility Assessment of Sensor-based Biosignal Measurement in Human-Robot Interaction, International Journal for Social Robotics, Vol.4, No.3, pp.303–316, 2012.

(47) Ryohei Ueda, Shunichi Nozawa, Kei Okada, Masayuki Inaba: Biped Humanoid Navigation System Supervised through Interruptible User-Interface with Asynchronous Vision and Foot Sensor Monitoring, in Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014), pp.022, 2014.

(48) 古田 悠貴, 垣内 洋平, 三喜田浩行, 植田 亮平, 岡田 慧, 稲葉 雅幸: 日常生活支援ロボットによるオンサイト教示可能なロボットシステム, in 日本機械学会ロボティクス・メカトロニクス講演会'14 講演論文集, 1P2-Q06, 2014.

(49) 垣内 洋平, 植田 亮平, 室岡 雅樹, 野田 晋太朗, 野沢 峻一, 岡田 慧, 稲葉 雅幸: Gazebo/ROS と OpenRTM によるロボットシミュレーション環境を用いた透過的なヒューマノイド評価環境の構築, in 日本機械学会ロボティクス・メカトロニクス講演会'14 講演論文集, 3P1-G07, 2014.

(50) 東風上奏絵, 稲垣祐人, 古田悠貴, 植田亮平, 垣内洋平, 岡田慧, 稲葉雅幸: 具体的操作期の子供との対話を目指した視聴覚対話協調システムに関する研究, in 第 32 回日本ロボット学会学術講演会講演論文集, 1C3-08, 2014.

(51) 古田 裕介, 植田 亮平, 垣内 洋平, 岡田 慧, 稲葉 雅幸: 通信帯域制限の変化に応じた環境情報取得とロボット操作レベルの動的変更を行う遠隔操縦ロボットシステム, in 第 15 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, pp.2087–2090, 2014.

(52) 岩石 智志, 室岡 雅樹, 植田 亮平, 佐藤 顕治, 岡田 慧, 稲葉 雅幸: 物体認識に基づく操縦補助ソフトウェアとヒューマノイド型入力デバイスを統合したタスク実現システム,

in 第 15 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, pp.2110–2113, 2014.

(53) 大坪 諭史, 室岡 雅樹, 植田 亮平, 黒岩 英則, 野沢 峻一, 垣内 洋平, 岡田 慧, 稲葉 雅幸: 三次元視覚に基づく等身大ヒューマノイドロボットによるバルブ開閉操作, in 第 15 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, pp.2294–2297, 2014.

(54) Masaki Murooka, Yuto Inagaki, Ryohei Ueda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Whole-body Holding Manipulation by Humanoid Robot based on Transition Graph of Object Motion and Contact, in Proceedings of The 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3950-3955, 2015.

(55) 小原 由羽, 室岡 雅樹, 植田 亮平, 大坪 諭史, 岡田 慧, 稲葉 雅幸 (東大): 未知環境における人の遠隔指示と自律認識に基づく等身大ヒューマノイドの操作計画と行動実現, in 日本機械学会ロボティクス・メカトロニクス講演会'15 講演論文集, 2P1-D10, 2015.

(56) 室岡 雅樹, 稲垣 祐人, 植田 亮平, 野沢 峻一, 垣内洋平, 岡田 慧, 稲葉 雅幸: 物体姿勢・把持接触グラフに基づく双腕ロボットによる全身接触抱え上げ操作の動作計画法, in 第 33 回日本ロボット学会学術講演会講演論文集, 1G3-03, 2015.

(57) 室岡 雅樹, 小原 由羽, 植田 亮平, 野沢 峻一, 垣内洋平, 岡田 慧, 稲葉 雅幸: 手動修正可能な自律認識行動システムによる未知遠隔環境下での全身マニピュレーション実現法, in 第 33 回日本ロボット学会学術講演会講演論文集, 3I3-04, 2015.

(58) 小原 由羽, 室岡 雅樹 , 植田 亮平, 野沢 峻一, 垣内洋平, 岡田 慧, 稲葉 雅幸: ヒューマノイドロボットによる認識に基づく電動工具操作の動作生成に関する研究, in 第 33 回日本ロボット学会学術講演会講演論文集, 3I3-05, 2015.

(59) Yu Ohara, Masaki Murooka, Ryohei Ueda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Configurable Autonomy Applicable to Humanoid Manipulation in Unstructured and Communication-Limited Environment, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), pp.373–380, 2015.

(60) Shunichi Nozawa, Eisoku Kuroiwa, Kunio Kojima, Ryohei Ueda, Masaki Murooka, Shintaro Noda, Iori Kumagai, Yu Ohara, Yohei Kakiuchi, Kei Okadad, Masayuki Inaba: Multi-Layered Real-Time Controllers for Humanoid's Manipulation and Lo-

comotion Tasks with Emergency Stop, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), pp.381–388, 2015.

(61) Yohei Kakiuchi, Kunio Kojima, Eisoku Kuroiwa, Masaki Murooka, Shintaro Noda, Iori Kumagai, Ryohei Ueda, Fumihito Sugai, Shunichi Nozawa, Kei Okada, Masayuki Inaba: Development of Humanoid Robot System for Disaster Response Through Team NEDO-JSK's Approach to DARPA Robotics Challenge Finals, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015),

(62) Ryohei Ueda, Masaki Murooka , Yu Ohara , Iori Kumagai , Ryo Terasawa , Yuki Furuta , Kunio Kojima , Tatsuhi Karasawa , Fumihito Sugai , Satoshi Iwaishi , Shunichi Nozawa , Yohei Kakiuchi , Kei Okada , Masayuki Inaba: Humanoid Integrated UI System for Supervised Autonomy with Massive Data Visualization over Narrow and Unreliable Network Communication for DRC Competition, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), , 2015.

参考文献

[1] Defense Advanced Resaerch Projects Agency. DARPA Robotics Challenge. `www.theroboticschallenge.org`, 2015.

[2] Richard Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *IJCAI*, pp. 608–620, 1971.

[3] Samir Alili, A Kumar Pandey, E Akin Sisbot, and Rachid Alami. Interleaving symbolic and geometric reasoning for a robotic assistant. In *ICAPS Workshop on Combining Action and Motion Planning*, 2010.

[4] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proceedings of The 2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, 2011.

[5] Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *I. J. Robotic Res.*, Vol. 28, No. 1, pp. 104–126, 2009.

[6] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *International Journal of Robotics Research*, Vol. 32, No. 9-10, 2013.

[7] Brian Coltin, Manuela M Veloso, and Rodrigo Ventura. Dynamic user task scheduling for mobile robots. In *Automated Action Planning for Autonomous Mobile Robots*, 2011.

[8] L. Mudrova and N. Hawes. Task scheduling for mobile robots using interval algebra. In *Proceedings of The 2015 IEEE International Conference on Robotics and Automation*, pp. 383–388, 2015.

[9] 比留川博久, 石綿陽一, 戸田賢二, 小原一太郎. 干渉チェックのインプリサイス実時間計

算アルゴリズム. 第 33 回日本ロボット学会学術講演会講演論文集, p. 3H21, 1999.

[10] Jane WS Liu, Kwei-Jay Lin, Wei Kuan Shih, Albert Chuang-shi Yu, Jen-Yao Chung, and Wei Zhao. *Algorithms for scheduling imprecise computations*. Springer, 1991.

[11] Jia-Ming Chen, Wan-Chen Lu, Wei-Kuan Shih, and Ming-Chung Tang. Imprecise computations with deferred optional tasks. *J. Inf. Sci. Eng.*, Vol. 25, No. 1, pp. 185–200, 2009.

[12] E. Sommerlade and I. Reid. Probabilistic surveillance with multiple active cameras. In *Proceedings of The 2010 IEEE International Conference on Robotics and Automation*, pp. 440–445, 2010.

[13] José M. Sanchiz and Robert B. Fisher. A Next-Best-View Algorithm for 3D Scene Recovery with 5 Degrees of Freedom. In *Proceedings of the British Machine Vision Conference 1999*, 1999.

[14] F. Keith, N. Mansard, S. Miossec, and A. Kheddar. Optimization of tasks warping and scheduling for smooth sequencing of robotic actions. In *In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pp. 1609–1614, 2009.

[15] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi. Visually-guided grasping while walking on a humanoid robot. In *Proceedings of The 2007 IEEE International Conference on Robotics and Automation*, pp. 3041–3047, 2007.

[16] Nicolas Mansard, Olivier Stasse, Paul Evrard, and Abderrahmane Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, 2009.

[17] S. Suzuki, Y. Tazaki, and T. Suzuki. Simultaneous optimization of timing and trajectory in sequential and parallel tasks of humanoid robots. In *Proceedings of the 2011 IEEE-RAS International Conference on Humanoid Robots*, pp. 596–601, 2011.

[18] D.McDermott. Pddl – the planning domain definition language. `ftp://ftp.cs.yale.edu/pub/mcdermott`, 1998.

[19] Franois Flix Ingrand, Raja Chatila, Rachid Alami, and Frdric Robert. Prs: A high level supervision and control language for autonomous mobile robots. In *Proceedings*

of The 1996 IEEE International Conference on Robotics and Automation, pp. 43–49, 1996.

[20] Benjamin Lussier, Matthieu Gallien, Jérémie Guiochet, Félix Ingrand, Marc-Olivier Killijian, and David Powell. Planning with diversified models for fault-tolerant robots. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *ICAPS*, pp. 216–223. AAAI, 2007.

[21] Cynthia Matuszek, John Cabral, Michael J Witbrock, and John DeOliveira. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pp. 44–49, 2006.

[22] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pp. 1012–1017, 2010.

[23] Moritz Tenorth and Michael Beetz. Knowrob—knowledge processing for autonomous personal robots. In *In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pp. 4261–4266, 2009.

[24] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14–23, 1986.

[25] Eyal Amir and Pedrito Maynard-Zhang. Logic-based subsumption architecture. *Artificial Intelligence*, Vol. 153, No. 1, pp. 167–237, 2004.

[26] Thomas B Sheridan. Telerobotics. *Automatica*, Vol. 25, No. 4, pp. 487–507, 1989.

[27] Gordon Cheng and Alexander Zelinsky. Supervised autonomy: A framework for human-robot systems development. *Auton. Robots*, Vol. 10, No. 3, pp. 251–266, 2001.

[28] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[29] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pp. 593–598, 2002.

[30] Toby HJ Collett, Bruce A MacDonald, and Brian P Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, p. 145, 2005.

[31] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and Woo-Keun Yoon. RT-middleware: distributed component middleware for RT (robot technology). In *In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pp. 3933–3938, 2005.

[32] David S Touretzky and Ethan J Tira-Thompson. Tekkotsu: A framework for aibo cognitive robotics. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20, p. 1741, 2005.

[33] Herman Bruyninckx, Peter Soetens, and Bob Koninckx. The real-time motion control core of the Orocos project. In *Proceedings of The 2003 IEEE International Conference on Robotics and Automation*, pp. 2766–2771, 2003.

[34] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully B. Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *Proceedings of The 2009 IEEE International Conference on Robotics and Automation*, Open-Source Software workshop, 2009.

[35] Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pp. 4057–4062, 2010.

[36] 小倉崇, 神崎秀, 岡田慧, 稲葉雅幸. 主体的に情報収集を行う日常生活支援ヒューマノイドの構成法. 第 25 回日本ロボット学会学術講演会講演論文集, p. 3H21, 2007.

[37] 岡田慧, 小島光晴, 稲葉雅幸. 認識行動共有知識ベースシステムにおける複数視覚特徴統合による物体認識. 日本ロボット学会誌, Vol. 26, No. 6, pp. 537–545, 2008.

[38] Kei Okada, Satoru Tokutsu, Takashi Ogura, Mitsuharu Kojima, Yuto Mori, Toshiaki Maki, and Masayuki Inaba. Scenario controller for daily assistive humanoid using visual verification, task planning and situation reasoning. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, pp. 398–405, 2008.

[39] Shunichi Nozawa, Masaki Murooka, Shintaro Noda, Kei Okada, and Masayuki Inaba. Description and execution of humanoid's object manipulation based on object-

environment-robot contact states. In *In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)*, pp. 2608 – 2615, 2013.

[40] Raoul Zöllner, Tamim Asfour, and Rüdiger Dillmann. Programming by demonstration: dual-arm manipulation tasks for humanoid robots. In *In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, pp. 479–484, 2004.

[41] Arne Lehmann, Ralf Mikut, and Tamim Asfour. Petri nets for task supervision in humanoid robots. *VDI BERICHTE*, Vol. 1956, p. 71, 2006.

[42] Freek Stulp, Andreas Fedrizzi, Lorenz Mösenlechner, and Michael Beetz. Learning and reasoning with action-related places for robust mobile manipulation. *Journal of Artificial Intelligence Research*, pp. 1–42, 2012.

[43] Radu Bogdan Rusu, Ioan Alexandru Şucan, Brian Gerkey, Sachin Chitta, Michael Beetz, and Lydia E Kavraki. Real-time perception-guided motion planning for a personal robot. In *In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pp. 4245–4252, 2009.

[44] 井上博允. 日本ロボット学会誌, No. 2, pp. 87–90, 1984.

[45] P. M. Will and D. D. Grossman. An experimental system for computer controlled mechanical assembly. *IEEE Trans. Comput.*, Vol. 24, No. 9, pp. 879–888, 1975.

[46] R. J. Popplestone, A. P. Ambler, and I. Bellos. RAPT, A language for describing assemblies. *Industrial Robot*, Vol. 5, No. 3, pp. 131–137, 1978.

[47] Dieter Fox, Wolfram Burgard, Sebastian Thrun, et al. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, Vol. 4, No. 1, pp. 23–33, 1997.

[48] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of The 1993 IEEE International Conference on Robotics and Automation*, pp. 802–807, 1993.

[49] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *In Proceedings of the 2003 IEEE/RSJ In-*

ternational Conference on Intelligent Robots and Systems (IROS'03), Vol. 2, pp. 1644–1650 vol.2, 2003.

[50] G Buttazzo, E Conticelli, Gerardo Lamastra, and Giuseppe Lipari. Robot control in hard real-time environment. In *Real-Time Computing Systems and Applications, 1997. Proceedings., Fourth International Workshop on*, pp. 152–159, 1997.

[51] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the internet. *Queue*, Vol. 9, No. 11, pp. 40:40–40:54, 2011.

[52] Matthew Johnson, Jeffrey M Bradshaw, Paul J Feltovich, Catholijn M Jonker, Birna van Riemsdijk, and Maarten Sierhuis. The fundamental principle of coactive design: interdependence must shape autonomy. In *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pp. 172–191. Springer, 2011.

[53] Twan Koolen, Jesper Smith, Gray Thomas, Sylvain Bertrand, John Carff, Nathan Mertins, Douglas Stephen, Peter Abeles, Johannes Englsberger, Stephen Mccrory, et al. Summary of team ihmc's virtual robotics challenge entry. In *Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots*, pp. 307–314, 2013.

[54] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, Vol. 12, No. 2, pp. 494–506, 2015.

[55] JohnM. Hsu and StevenC. Peters. Extending open dynamics engine for the darpa virtual robotics challenge. In Davide Brugali, JanF. Broenink, Torsten Kroeger, and BruceA. MacDonald, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, Vol. 8810 of *Lecture Notes in Computer Science*, pp. 37–48. Springer International Publishing, 2014.

[56] Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf, Anant Gupta, Oskar von Stryk, Felipe Bacim, Doug A Bowman, Alex Goins, Ravi Balasubramanian, and David C Conner. Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials. *Journal of Field Robotics*, Vol. 32, No. 3, pp. 352–377, 2015.

[57] R Darken, Kurt Kempster, and Barry Peterson. Effects of streaming video quality of service on spatial comprehension in a reconnaissance task. In *Proceedings of the meeting of I/ITSEC*, 2001.

[58] Jan BF Van Erp and Pieter Padmos. Image parameters for driving with indirect viewing systems. *Ergonomics*, Vol. 46, No. 15, pp. 1471–1499, 2003.

[59] Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Design of high torque and high speed leg module for high power humanoid. In *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pp. 4497–4502, 2010.

[60] Yoshito Ito, Takuya Nakaoka, Junichi Urata, Kazuya Kobayashi, Shunich Nozawa, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Development and verification of life-size humanoid with high-output actuation system. In *Proceedings of The 2014 IEEE International Conference on Robotics and Automation*, pp. 3433–3438, 2014.

[61] N. Hogan. Impedance control: An approach to manipulation. In *American Control Conference, 1984*, pp. 304–313, 1984.

[62] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi. Biped walking stabilization based on linear inverted pendulum tracking. In *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pp. 4489–4496, 2010.

[63] Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Controlling the planar motion of a heavy object by pushing with a humanoid robot using dual-arm force control. In *Proceedings of The 2012 IEEE International Conference on Robotics and Automation*, pp. 1428–1435, 2012.

[64] Tan Fung Chan and Rajiv V Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *Robotics and Automation, IEEE transactions on*, Vol. 11, No. 2, pp. 286–292, 1995.

[65] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pp. 2053–2058, 2007.

[66] Faruk Akgul. *ZeroMQ*. Packt Publishing, 2013.

[67] Steven Fadden, Patricia May Ververs, and Christopher D Wickens. Costs and benefits of head-up display use: A meta-analytic approach. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 42, pp. 16–20, 1998.

[68] A. Leeper, K. Hsiao, M. Ciocarlie, Leila Takayama, and D. Gossow. Strategies for human-in-the-loop robotic grasping. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp. 1–8, 2012.

[69] Defense Advanced Resaerch Projects Agency. DARPAtv Channel at Youtube. `www.youtube.com/`, 2015.

[70] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[71] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. Autonomous behavior control architecture of entertainment humanoid robot sdr-4x. In *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, Vol. 1, pp. 960–967 vol.1, 2003.

[72] S.Kajita, F.Kanehiro, K.Kaneko, K.Yokoi, and H.Hirukawa. The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation. In *Proc. Inter. Conf. on Intelligent Robots and Systems(IROS)*, pp. 239–246, 2001.

[73] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa. A realtime pattern generator for biped walking. In *Proceedings of The 2002 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 31–37 vol.1, 2002.

[74] Kazuhito Yokoi, Fumio Kanehiro, Kenji Kaneko, Shuuji Kajita, Kiyoshi Fujiwara, and Hirohisa Hirukawa. Experimental Study of Humanoid Robot HRP-1S. *The International Journal of Robotics Research"*, Vol. 23, No. 4–5, pp. 351–362, 2004.

[75] James Kuffner, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika Inoue. Online footstep planning for humanoid robots. In *Proceedings of The 2003 IEEE International Conference on Robotics and Automation*, pp. 932–937, 2003.

[76] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of The 2005 IEEE International Conference on Robotics and Automation*, pp. 629–634, 2005.

[77] James J. Kuffner Jr. and Steven M. Lavalle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of The 2000 IEEE International Conference on Robotics and Automation*, pp. 995–1001, 2000.

[78] Zeyang Xia, Jing Xiong, and K. Chen. Global navigation for humanoid robots using sampling-based footstep planners. *Mechatronics, IEEE/ASME Transactions on*, Vol. 16, No. 4, pp. 716–723, 2011.

[79] J. Garimort and A Hornung. Humanoid navigation with dynamic footstep plans. In *Proceedings of The 2011 IEEE International Conference on Robotics and Automation*, pp. 3982–3987, 2011.

[80] Philipp Michel, Joel Chestnutt, Satoshi Kagami, Koichi Nishiwaki, James Kuffner, and Takeo Kanade. Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing. In *In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pp. 463–469, 2007.

[81] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *Proceedings of the 2007 IEEE-RAS International Conference on Humanoid Robots*, pp. 196–202, 2007.

[82] J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J.J. Kuffner, , and S. Kagami. Biped navigation in rough environments using on-board sensing. In *In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pp. 3543–3548, 2009.

[83] Zhengyou Zhang. Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, Vol. 19, No. 2, pp. 4–10, 2012.

[84] Daniel Maier, Armin Hornung, and Maren Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 692–697, Osaka, Japan, 2012.

[85] Pter Fankhauser, Michael Bloesch, Christian Gehring, Marco Hutter, and Roland Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *International Conference on Climbing and Walking Robots (CLAWAR)*, p. 433, 2014.

[86] Koichi Nishiwaki, Joel E. Chestnutt, and Satoshi Kagami. Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. *I. J.*

Robotic Res., Vol. 31, No. 11, pp. 1251–1262, 2012.

[87] Joshua Bialkowski, Sertac Karaman, Michael Otte, and Emilio Frazzoli. Efficient collision checking in sampling-based motion planning. In *Algorithmic Foundations of Robotics X*, pp. 365–380. Springer, 2013.

[88] Jia Pan, Ioan Sucan, Subhashini Chitta, Dinesh Manocha, et al. Real-time collision detection and distance computation on point cloud sensor data. In *Proceedings of The 2013 IEEE International Conference on Robotics and Automation*, pp. 3593–3599, 2013.

[89] Jia Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *Proceedings of The 2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[90] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, pp. 189–206, 2013.

[91] Jr. Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, Vol. 1, pp. 500–505 vol.1, 2001.

[92] Joel Chestnutt, Koichi Nishiwaki, James Kuffner, and Satoshi Kagami. An adaptive action model for legged navigation planning, 2007.

[93] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, Vol. 56, No. 11, pp. 927 – 941, 2008.

[94] Tahir Rabbani, Frank van den Heuvel, and G Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 36, No. 5, pp. 248–253, 2006.

[95] D G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artif. Intell.*, Vol. 31, No. 3, pp. 355–395, 1987.

[96] David G. Lowe and David G. Lowe. Robust model-based motion tracking through

the integration of search and estimation. *International Journal of Computer Vision*, Vol. 8, pp. 113–122, 1992.

[97] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Fusing online and offline information for stable 3d tracking in real-time. In *In Proc. Intl. Conference on Computer Vision and Pattern Recognition*, pp. 241–248, 2003.

[98] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, pp. 1385–1391, 2004.

[99] Pedram Azad, David Munch, Tamim Asfour, and Rdiger Dillmann. 6-dof model-based tracking of arbitrarily shaped 3d objects. In *ICRA11*, pp. 5204–5209, 2011.

[100] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, Vol. 60, No. 2, pp. 91–110, 2004.

[101] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, Vol. 110, No. 3, pp. 346–359, 2008.

[102] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 32, No. 3, pp. 448–461, 2010.

[103] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, Vol. 2, No. 2, pp. 438–469, 2009.

[104] Mustafa Özuysal, Vincent Lepetit, François Fleuret, and Pascal Fua. Feature harvesting for tracking-by-detection. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV (3)*, Vol. 3953 of *Lecture Notes in Computer Science*, pp. 592–605. Springer, 2006.

[105] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, pp. 145–152, 2001.

[106] Andrew Johnson and Sing Bing Kang. Registration and integration of textured 3-d data. In *IMAGE AND VISION COMPUTING*, pp. 234–241, 1996.

[107] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison,

et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 559–568, 2011.

[108] Gilles Burel and Hugues Henocq. Three-dimensional invariants and their application to object recognition. *Signal Processing*, Vol. 45, No. 1, pp. 1–22, 1995.

[109] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on Geometry Processing*, pp. 197–206, 2005.

[110] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, pp. 3384 –3391, Baden-Baden, Germany, 2008.

[111] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of The 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[112] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *STATISTICS AND COMPUTING*, Vol. 10, No. 3, pp. 197–208, 2000.

[113] Pedram Azad, David Munch, Tamim Asfour, and Rüdiger Dillmann. 6-dof model-based tracking of arbitrarily shaped 3d objects. In *Proceedings of The 2011 IEEE International Conference on Robotics and Automation*, pp. 5204–5209. IEEE, 2011.

[114] Dieter Fox. Kld-sampling: Adaptive particle filters. In *In Advances in Neural Information Processing Systems 14*, pp. 713–720. MIT Press, 2001.

[115] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 263–270, 2011.

[116] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*, pp. 862–869, 2014.

[117] Maria Fox and Derek Long. Pddl2.1 : An extension to pddl for expressing temporal planning domains. `http://ww.dur.ac.uk/d.p.long/IPC/pddl.htm`, 2002.

[118] Stefan Edelkamp and Jorg Hoffman. Pddl2.2: The language for the clasical part of the 4th international planning competition. `http://www.informatik.uni-freiburg.de/ki/teaching/ws0607/aip/pddl2.2.pdf`, 2004.

[119] Alfonso Gerevini and Derek Long. Plan constraints and preferenced in pddl3. `http://www.cs.yale.edu/homes/dvm/papers/pddl-ipc5.pdf`, 2005.

[120] 金広文男, 稲葉雅幸, 井上博允. Statenet:障害回復機能を内蔵する行動空間の状態遷移図表現. 日本ロボット学会誌, Vol. 20, No. 8, pp. 835–843, 2002.

[121] R. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, Vol. 2, No. 1, pp. 14–23, 1986.

[122] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[123] K.S. Tso, M. Hecht, and N.I. Marzwell. Fault-tolerant robotic system for critical applications. In *Proceedings of The 1993 IEEE International Conference on Robotics and Automation*, pp. 691–696 vol.3, 1993.

[124] Ashwani Aggarwal, Susmit Biswas, Sandeep Singh, Shamik Sural, and Arun K Majumdar. Object tracking using background subtraction and motion estimation in mpeg videos. In *Computer Vision–ACCV 2006*, pp. 121–130. Springer, 2006.

[125] Ruolin Zhang and Jian Ding. Object tracking and detecting based on adaptive background subtraction. *Procedia Engineering*, Vol. 29, pp. 1351–1355, 2012.

[126] J. Kammerl, N. Blodow, R.B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time compression of point cloud streams. In *Proceedings of The 2012 IEEE International Conference on Robotics and Automation*, pp. 778–785, 2012.

[127] 内山勝, 中村仁彦. ロボットモーション. 岩波書店, 2004.

[128] 梶田秀司, 横井一仁, 比留川博久, 原田研介. ヒューマノイドロボット. オーム社, 2005.

[129] Michael Gienger, Herbert Jansen, and Christian Goeric. Exploiting Task Intervals for Whole Body Robot Control. In *In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, pp. 2484 – 2490, 2006.

[130] Tan Fung Chan and R.V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. In *Robotics and Automation,*

IEEE Transactions on, Vol. 11, pp. 286–292, 1995.

[131] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals. In *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots*, pp. 575–580, 2006.

[132] Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pp. 2053 – 2068, 2007.

[133] Larsen E., Gottschalk S., Lin M.C., and Manocha D. Fast distance queries with rectangular swept sphere volumes. In *Proceedings of The 2000 IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3719–3726, 2000.

[134] Miomir Vukobratović and J Stepanenko. On the stability of anthropomorphic systems. *Mathematical biosciences*, Vol. 15, No. 1, pp. 1–37, 1972.

[135] 原田研介, 梶田秀司, 金広文男, 藤原清司, 金子健二, 横井一仁, 比留川博久. ヒューマノイドロボットの脚腕協調における zmp 解析. 日本ロボット学会誌, Vol. 22, No. 1, pp. 28–36, 2004.

以上

1p～ 211p 完

博士論文

平成 27 年 12 月 4 日提出

東京大学大学院 情報理工学系研究科
創造情報学専攻 博士課程
植田 亮平