2019 年度 修士論文

機械学習を用いたパーソナルロボットハンドの開発
～上肢機能障害者の支援を目指して～

Personal Robot Hands Supported by Machine Learning and Deep Learning ~ Help the disabled in the future ~

2020 年 1 月 30 日

指導教員 小野寺宏 特任教授

東京大学大学院研究科

電気系工学専攻 37-186508

山田敦史

概要

上肢障害者の生活支援を目指し自律ロボットハンドを開発した．1 号機ではスマートフォンのカメラと CPU のみによる強化学習で特定色の物体への接近と把持に成功し，2 号機では画像認識技術と深層学習により複数の物体から対象物を識別して使用者のもとに運搬することが可能になった．本ロボットハンドは患者さんからも好評で障害者支援への実用化が期待される．

目次

図目次

表目次

第 1 章

序論

1.1 研究背景

1.1.1 ロボットとは

元来「ロボット」とは奴隷機械や人造人間という語源から来ており，人間の労働を肩代わりするものである．現在では，ロボットとはアメリカの作家 Isaac Asimov が『われはロボット』[1] で記したロボット工学三原則（Table 1.1）により定義されている．

Table 1.1: Three Laws of Robotics[1].

第 1 条	ロボットは人間に危害を加えてはならない．また，人間に危害が及ぶのを見逃してはならない．
第 2 条	ロボットは人間から与えられた命令に従わなければならない．ただし，与えられた命令が第 1 条に反する場合は，この限りではない．
第 3 条	ロボットは第 1 条および第 2 条に反しない限り自身を守らなければならない．

ロボットは機構（メカニズム）やアクチュエータ，動力源（バッテリー），センサ，コンピュータ装置（単にコンピュータと呼ぶこともある）が総合的に絡み合って動作する．これら 5 つの要素をソフトウェアで結びつけて制御を行う．例えば，人間の腕を機械で実現しようとするとき，肩・上腕・前腕・手首・手・5 指という各構成要素を組み合わせ，1 つのシステムに構成したものが機構である．機構を構成しても，それを動作させるためには駆動源が必要となる．この駆動源がアクチュエータであり，肩・肘・手首といった関節がアクチュエータである．また，ロボットの機構を動作させるアクチュエータにはバッテリーが必要である．アクチュエータを制御する際のフィードバックとしてセンサを用いる．センサは，ロボット自身の動作や環境の情報を把握する．コンピュータではセンサからの信号をもとに，アクチュエータや機構の動作を最適化するプログラムを実行する．

1.1.2 医療・福祉ロボットとその発展

現在ロボティクス技術の向上により様々な種類のロボットが開発されている．例えば Pepper[2] などの作業支援ロボットは，オフィスや商業施設など人がいる環境で，案

内，搬送などの作業を支援するサービスロボットである．形状は移動の効率を重視して 2 足歩行ではなく車輪や 4 足歩行を採用している．しかし，多くのロボットはヒューマノイド型をしており，違和感がなく人間社会に溶け込めるよう配慮されている．また，ルンバ [3] などの掃除ロボットは，障害物センサや段差センサにより自動的に掃除領域を認識し，その区域を清掃するロボットである．医療ロボットは，da Vinci[4] といった手術ロボットや介護ロボットなど，目的よって多種多様である．また福祉ロボットは，障害者や高齢者などの日常生活を補助するロボットである．自律型は少なく，マイスプーン [5] などのように障害者自身の操作によって動いているものが多い．特に医療・福祉ロボットでは高い安全性が求められる．

上記で述べたように様々な分野でロボットが活躍している．特に，医療や福祉といった人間の生活環境下で作業可能なパーソナルロボットの需要の増加は顕著である．しかし，人の生活環境下において完全に自律して動作及び人の補助ができるロボットの実現は困難な状況にある．なぜならば，ロボットが自律行動を行うためには物体や人物の認識，自己位置の認識，そして認識に基づく判断を人の生活環境下で行わなければならないからである．

パーソナルロボットとして Mobile Manipulator の開発が行われている．Mobile Manipulator は工場に限定されて使用されているが，これを家庭内でも使用できるように軽く，そして小さくすることで衝突リスクを削減する試みがされている．2018 年には，Preferred Networks は室内に散らかった家庭用品を片付けるロボットシステムを発表した [6]．部屋の全自動片付けは従来のロボットシステムでは実現困難であったが，近年の深層学習の発展によって初めて実用的なレベルとなった．物をつかむ，物を置く，動作計画を立てる，人の指示に対応するなど，ロボットが人間の生活空間で仕事をするために必要な物体認識・ロボット制御・音声言語理解技術に最先端の深層学習を用いた結果，ロボットが高速・高精度に動作できるようになった．

しかし，深層学習では多くの計算リソースが必要である．計算リソースとは深層学習を動かすハードウェア（いわゆる PC），特に GPU(Graphical Processor Unit) である．GPU は形状が大きく消費電力も大きいため，高性能な GPU を複数 Mobile Manipulator に搭載することは難しい．そこで Cloud ネットワークを利用する方法がある．Cloud を介して GPU リソースを使用する．これにより Cloud とのインターフェースを有しているデバイスであれば演算性能は低くても推論が可能となり，スマートフォンやノート PC，ボードコンピュータと言った小さなデバイスを使用することができる．しかし常時インターネットに接続している必要があるため，災害時に使用できなくなるという問題がある．一方で，Cloud を使わず Local で処理を行う Edge コ

ンピューティングという分野も近年研究開発がされている．Edge コンピューティングではより小型のデバイス（Edge デバイス）で処理を行うことを目的としており，小型，低消費電力，高精度より高速性が求められる．しかし深層学習を使用しているパーソナルロボットは高さ 1 m と大きく，重量も大きいため誤動作時のリスクが大きい．また，家庭内などで使用する分には良いが外に持ち出してどこでも使用するといった事ができない．

1.1.3　医療・福祉分野におけるロボット義手

医療分野におけるロボットの用途として期待されるものと言えばロボット義手である．義手には装飾用義手や能動義手，作業用義手などがあるが，現在主流の筋電電動義手（筋電義手）は，使用者が筋肉を動かすことで筋電位を読み取り直感的な操作を可能にする．しかし，筋電位を用いた義手には訓練が必要でリハビリテーション施設で行う必要があるが，その施設が極めて少ない [7]．また筋電位が上手く出せない患者や，腕そのものはあるが麻痺して動かせない患者に対しては使用できない．さらに非侵襲的な筋電義手では活用できる信号が限られる（通常，屈筋からの信号と伸筋からの信号との 2 チャンネル）．自由度の高い動作を可能にするロボット義手は重量が大きくなり，使用者の負担が大きくなってしまう．

筆者は茨城県立医療大学付属病院リハビリテーション科にて義手使用患者へのヒアリングを実施した．当患者は日常では能動義手を使用しており，病院でのリハビリの際に筋電義手の使用訓練を行なっている．半年のリハビリの後，ようやくペンを掴むことまでできた状況とのこと．筋電義手の使用感については，

- "筋電義手は重い"
- "（筋電が）ちゃんと伝わっているかわからない"
- "把持力の制御が難しく，（物を）落とすこともよくある"

とあまりポジティブな感想ではなかった．ヒアリングを通して得た現行の筋電義手の課題をまとめると，

- 義手本体が重い
- 筋電による義手の制御が難しい
- リハビリは大変で多くの時間を要する（その割にはできる動作は把持のみ）

である．以上のような課題があるため装着者のニーズを満たす義手がなく，実用化に

至っていないのが現状である．

1.2　関連研究

ここでは近年の福祉ロボットの開発について紹介する．

TOYOTA は障がい者や高齢者などの家庭内での自立生活をアシストする生活支援ロボット HSR（Human Support Robot）を開発した [8]．Figure 1.1 に示すような円筒形状をしており，廊下など狭い通路でも人や車椅子などと並走できる大きさである．移動用で 3 自由度，アームで 4 自由度，胴体の伸び縮みで 1 自由度の計 8 自由度の機構となっている．重量は 37 kg で最大移動速度は 0.8 km/h であり，ゆっくりとした動きで衝突リスクを考慮している．CPU ボード（2.4GHz クアッドコア），GPU ボード（Jetson）を搭載しており，さらに高負荷処理を行う場合は無線・有線 LAN を介して外部リソースを使用可能である．グリッパは 2 指対立の一般的なグリッパに加えて指先に吸盤を搭載し，硬い物体や柔らかい物体，そして平たい物体を含む 43 種類の物体の把持が可能である．インターフェースとして，タブレットを用いて HSR のカメラに映る遠把持対象物を指定して持って来させることができる．

HSR は建物内でしか使用できず，自律動作される場合は事前にマッピングした部屋でしか使えないといった課題がある．また，バッテリーおよび駆動時間の言及はされていない．

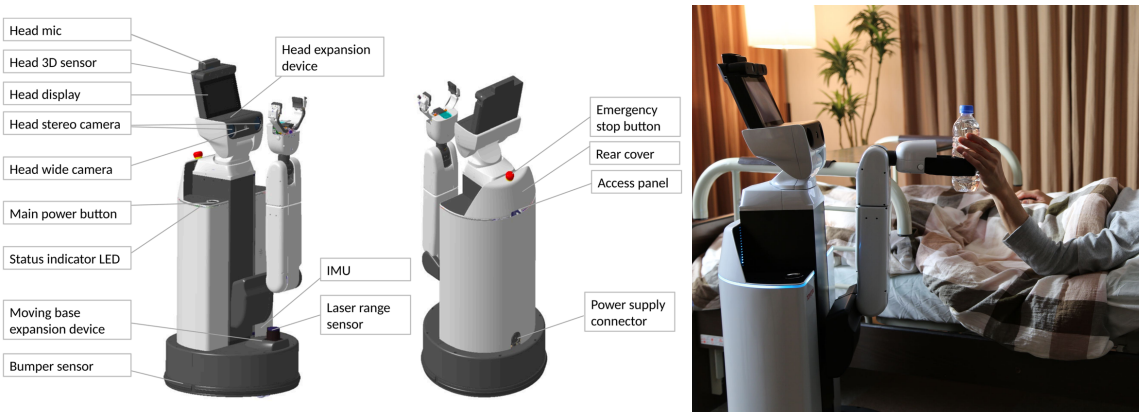


Figure 1.1: Human Support Robot (HSR) produced by TOYOTA[8,9].

筑波大の山海グループは片麻痺患者の日常生活を支援するサイバニックロボットアームを開発した [10,11]．サイバニックアームは，片麻痺者の失われた上肢機能を代替可能なロボットアームとして，あたかも使用者の腕であるかのように人の意思に基づ

いて機能することを目標に開発された．ロボットアームは 6 自由度を持ち，非麻痺側の作業領域を狭めないようにテーブルタスクの領域をカバーする最小サイズで設計してある．グリッパに静電容量センサが搭載されており，これにより把持力と重心位置を把握できる．物体にかかる外力に基づいて健側腕の動きとその動作に関するユーザの意図を推定し，推定した情報に基づいて健側腕と協働して支援動作を提供することができる．ペットボトルやプリンの蓋など 4 種類の物体に対して開封の支援を検証した．音声入力に作業内容と対象物の認識を行い，ロボットアームで対象物を固定して，健常の腕でキャップなどの開封を行える．

サイバニックアームは机に固定して使用するため，持ち運びが困難であり自宅の特定の机でしか使用できない．また，事前に対象物の最適な把持力を教示させる必要があるため，実際の生活では使用しにくい．

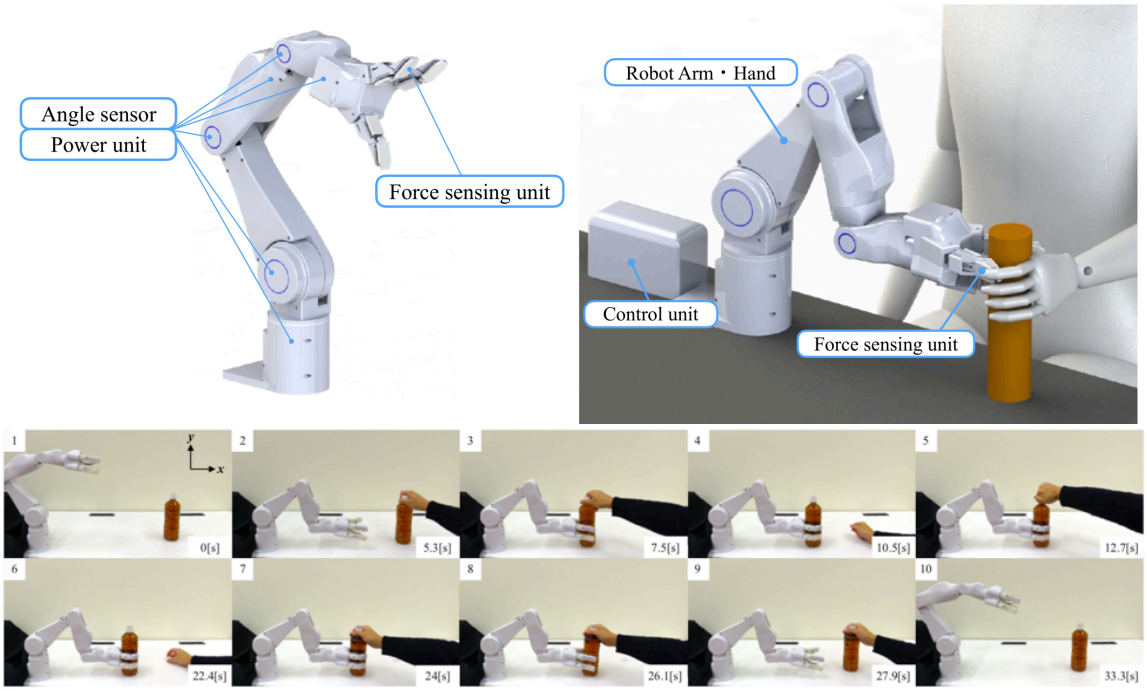


Figure 1.2: Cybernic robot arm produced by Sankai group[10, 11].

1.3 研究目的

本研究ではパーソナルロボットを小型化し，義手使用患者や片麻痺患者など上肢機能障害患者を対象としたパーソナルロボットを開発する．義手は常に身につけている

ように，パーソナルロボットも携帯できるよう小型化し腕の形をしたロボットハンドとする．制御の難しい筋電入力は避け，指令を与えると自律的に動作して対象物を把持するシステムを考える．上肢機能障害者にとってはパーソナルロボットはどんな事態でも動作する必要があるため，今回は Cloud は使用せず Edge で処理を行うこととする．また座って机で作業することが多いため使用場所を机の上に限定し，指定した物をピックアップするパーソナルロボットハンドを作製することを目的とする．

1.4 論文構成

本論文は以下の章によって構成される．

第 1 章　序論
第 2 章　深層学習とロボットへの応用
第 3 章　試作 1 号機の開発
第 4 章　試作 2 号機の開発
第 5 章　結論

第 1 章では，ロボットの背景とその中でも医療・福祉ロボットの発展について述べ，研究目的を定めた．第 2 章では深層学習・強化学習の原理について述べ，深層学習を用いた画像認識技術および強化学習を用いたロボット制御について紹介する．第 3 章ではスマートフォンを搭載し，スマートフォンの CPU とカメラを用いたロボットハンド（1 号機）の試作とその物理シミュレーションについて述べる．第 4 章では物体識別能力と把持機構を高性能化したロボットハンド（2 号機）の試作について述べる．第 5 章では本論文を振り返り，1 号機と 2 号機それぞれの達成点と課題点について述べ，次世代機の開発指針について言及し，今後の展望とする．

第 2 章

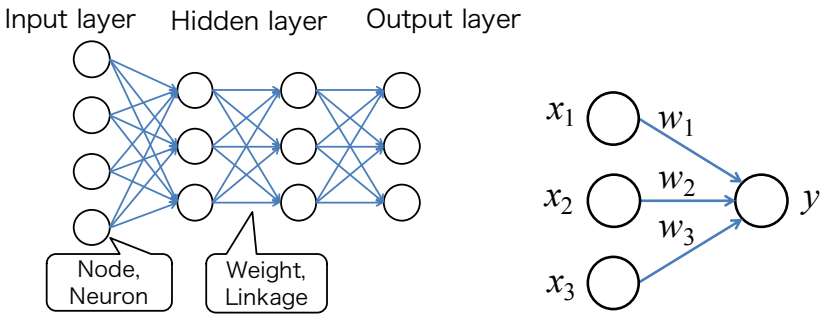
深層学習とロボットへの応用

2.1　ニューラルネットワーク

人間の脳にはニューロンと呼ばれる神経細胞が 1000 億個以上あり，それぞれが複数のニューロンが電気信号によって情報を伝達している．また脳にはシナプスという場所があり，ここで電気信号を細胞体へ受け渡す．細胞体はある閾値以上の電気信号がきた場合に他のニューロンへ電気信号を伝播させる (これを発火と呼ぶ)．このようなニューロンとシナプスで行われる演算を模倣したアルゴリズムを作ることができれば，人間のような思考や認識をコンピュータを使って再現できると考えた．そのアルゴリズムがニューラルネットワーク (Neural Network: NN) である．

2.1.1　多層パーセプトロン

ニューラルネットワークは入力層，出力層，隠れ層から構成され，層と層の間にはニューロン同士のつながりの強さを示す重みがある．非線形問題を扱うために 1986 年 Rumelhart によって考案されたのが，パーセプトロンを複数つなぎ合わせ入力と出力以外に隠れた層を持つ多層パーセプトロン (Multi-layer perceptron: MLP) である (Figure 2.1(a))．ニューラルネットワークで多層パーセプトロンの層を全結合 (fully connected: FC) 層とも呼ぶ．



(a) Muti-layer perceptron　　(b) Simple perceptron

Figure 2.1: Architecture of Muti-layer perceptron

Figure 2.1(a) における丸や矢印はそれぞれノード (またはニューロン) と重み (または結合) と呼び，ともに数値である．例えば画像を分類しようと思えば，各ピクセルの画素数を各ノードに入力する．例えば 28×28pixel のグレースケール画像であれば，

784 個のノードが必要となる．入力データ $\boldsymbol{x}$ が入力層に入ってくると，その値に重み $\boldsymbol{w}$ をかけ，活性化関数 H と呼ばれる関数に通し，結果 $\boldsymbol{y}$ を出力する．ここで，入力 $\boldsymbol{x}$，重み $\boldsymbol{w}$，出力 $\boldsymbol{y}$ を太字で表したが，これらは全てテンソルであり，1 つの層にあるノード $x_1, x_2, \cdots x_n$ を一括して $\boldsymbol{x}$ として表記している．

ここで，中間層の 1 つのノードについて考える．Figure 2.1(b) に MLP を構成する 1 ユニットである単純パーセプトロンを示した．この模式図を数式で表すと次のようになる．

$$y = H(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}) \tag{2.1}$$

$$= H\left(\sum_{i=1}^{3} w_i x_i + b_i\right) \tag{2.2}$$

ここで $\boldsymbol{b}$ はバイアスと呼ばれ，発火のしやすさを表している．中間層における活性化関数は，Eq. (2.3) に示す正規化線形関数 (rectified linear unit: ReLU) と呼ばれる関数) がよく用いられる．

$$H(x) = \max\{0, x\} = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \tag{2.3}$$

この演算を繰り返し出力層に書き出す．ここで，各層の重みの値によって出力結果は異なってくる．

出力層では，ノードの個数は区別したいクラス数分用意する．各ノードの出力値が各クラスに属している確率を表すように，活性化関数にはソフトマックス関数を用いる (ただし二値分類の場合はシグモイド関数を用いる)．ソフトマックス関数は Eq. (2.4) で表される．

$$y_i = \frac{\exp(x_i)}{\sum_{k=1}^{n} \exp(x_k)} \tag{2.4}$$

ここで y_i は，出力層が全部で n 個あるとして，i 番目の出力であることを示す．Eq. (2.4) からわかるように，入力の総和に対して 1 つのノードがどれくらいの値を持つかという割合で表されている．これにより各ノードの出力は確率として解釈できるため，値の一番大きいノードのインデックスを予測ラベルとして見ることができる．

2.1.2 畳み込みニューラルネットワーク

従来の画像認識では，画像から特徴を抽出しそれを識別器にかける手法が主流であった．古典的手法では画像から特徴を抽出するいわゆる特徴量設計が必要で，ここをいかにうまく設計するかがポイントであった．特徴抽出の方法として，HOG[12] や SIFT[13]，SURF[14] などがあり，これらによって抽出した特徴ベクトルを Support Vector Machine(SVM)[15] によって識別することが多かった．

しかし，1998 年に LeNet と呼ばれる畳み込みニューラルネットワーク (Convolutional Neural Network: CNN) が提案された [16]．CNN は畳み込み層とプーリング層からなっている．この畳み込みとプーリングの演算を通して，特徴量設計から識別までを end-to-end で行うことができる．

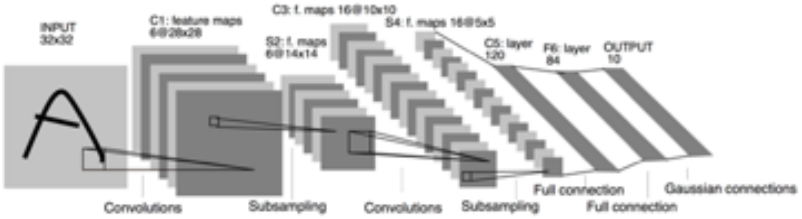


Figure 2.2: Architecture of convolutional neural network[16]

畳み込み層では，入力に対してフィルター (カーネルとも呼ばれる) を用意し，Eq. (2.5) に示す計算を行う．

$$y_{i,j} = (\boldsymbol{K} * \boldsymbol{x})_{i,j} \tag{2.5}$$

$$= \sum_m \sum_n x_{i+m,j+n} K_{m,n} \tag{2.6}$$

ここで，$\boldsymbol{K}$ はフィルター，$\boldsymbol{x}$ は入力，y は出力である．CNN ではこの演算の後に活性化関数に通す．これを図で表すと Figure 2.3 のようになる．

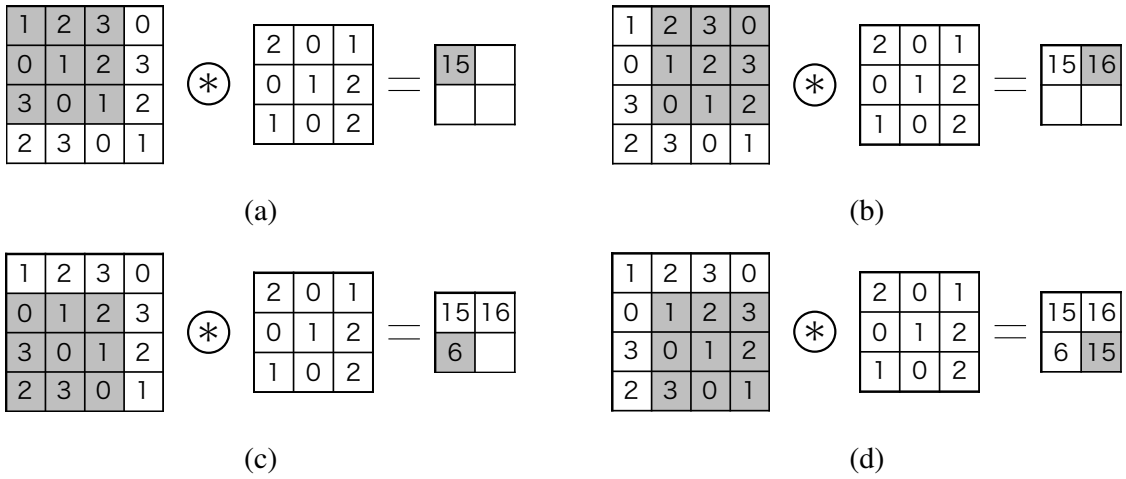


Figure 2.3: Operation process of convolution

Figure 2.3 では，フィルターのサイズは 3×3 であるが，大きさは任意である (3×3 や 5×5, 7×7 がよく用いられる)．また，フィルターは 1 マスずつ横にずらして計算を行っている．ずらし方をストライドといい，今回はストライド 1 である．CNN では多くの場合，ストライドは 1 である．このようなフィルターの畳み込み計算を行うと，フィルターごとに異なった画像の特徴を抽出して数値化することができる．

次にプーリングを行う．ここでは，画像認識で多く用いられる最大値プーリングについて述べる．Figure 2.4 に示すように，2×2 のプールサイズを用意した時，その範囲内にある最大値を取る演算である．ストライドはプールサイズと合わせ，プーリングを行った領域と被らないようにすることが一般的である．Figure 2.4 ではストライド 2 である．

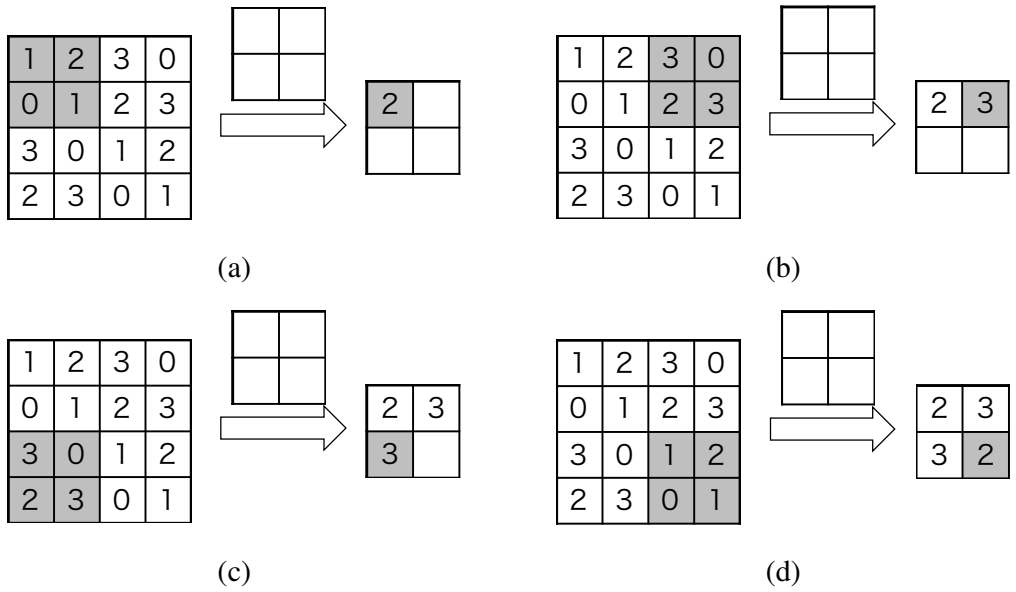


Figure 2.4: Operation process of max pooling

プーリング層では画像のサイズを小さくして (コンピュータの) 計算コストを減らし，微小な変化に対してロバストになる．

この畳み込みとプーリングを繰り返して，入力からフィルターの数だけ特徴を抽出し，この抽出した特徴マップを FC 層へ繋げて識別を行う手法が CNN である．

2.2 推論と学習

ニューラルネットワークでは推論フェーズと学習フェーズに分かれている．2.1 節は全て推論フェーズの話であり，順伝播ニューラルネットと呼ばれる．

学習フェーズでは，逆伝播ニューラルネットを用いる．逆伝播とは誤差逆伝播法 (Backpropergation)[17] から由来している．真値からの誤差を表す損失関数 (loss function) を用いて，パラメータの微分値を更新に用いる．損失関数はコスト関数，目的関数とも呼ばれる．この微分値を効率よく計算するアルゴリズムが誤差逆伝播法である．

損失関数は解く問題の目的に合わせて選ぶ必要がある．回帰問題では平均二乗和誤差 (mean squared error: MSE) が用いられる．

$$MSE = \frac{1}{N}\sum_{k}^{N}(y_k - t_k)^2 \tag{2.7}$$

また，クラス分け問題では交差エントロピー誤差 (cross entropy error) が用いられる．

$$CE = -\sum_{k}^{N} t_k \ln y_k \tag{2.8}$$

ここで，y は予測ラベルで，t は教師ラベルである．

2.2.1 最適化手法

損失関数を用いてパラメータを更新するが，更新手法にはいくつか方法があるため，ここでは本研究で使用した最適化手法について述べる．

SGD

SGD は確率的勾配降下法 (Stochastic gradient descent) と呼ばれる手法で，最も単純な最適化手法である．画像認識の分野では多く使われている．

SGD では以下の式 Eq. (2.9) でパラメータを更新する．

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial L}{\partial \boldsymbol{W}} \tag{2.9}$$

ここで，$\boldsymbol{W}$ は更新する重みパラメータ，$\dfrac{\partial L}{\partial \boldsymbol{W}}$ は $\boldsymbol{W}$ に関する損失関数の勾配である．また η は学習率と呼ばれ，実際には 0.01 や 0.001 といった値を前もって決めて使用する．SGD はパラメータの勾配を利用して，勾配方向にパラメータを更新するステップを繰り返して，徐々に最適なパラメータへと近づける手法である．

Adam

Adam[18] は adaptive moment estimation の略で，勾配の値の 1 乗和と 2 乗和の両方をパラメータ更新に用いる手法である．以下の式 Eq. (2.10) でパラメータを更新する．

$$\boldsymbol{m} \leftarrow \beta_1 \boldsymbol{m} + (1-\beta_1)\frac{\partial L}{\partial \boldsymbol{W}} \tag{2.10}$$

$$\boldsymbol{v} \leftarrow \beta_2 \boldsymbol{v} + (1-\beta_2)\left(\frac{\partial L}{\partial \boldsymbol{W}}\right)^2 \tag{2.11}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\hat{\boldsymbol{m}}}{\sqrt{\hat{\boldsymbol{v}}} + \epsilon} \tag{2.12}$$

m_t と v_t はそれぞれ、勾配の一次モーメント（平均値）と二次モーメント（分散した平方偏差）の概算値である．η，β_1，β_2 はそれぞれ学習パラメータである．また，$\hat{m}_t$，$\hat{v}_t$ はそれぞれ移動指数平均を用いた際に生じるバイアス (大きさを変えてしまっていることなど) を打ち消すために正則化しており，Eq. (2.13) で表される．

$$\hat{\boldsymbol{m}} = \frac{\boldsymbol{m}}{1-\beta_1} \tag{2.13}$$

$$\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{1-\beta_2} \tag{2.14}$$

学習パラメータはそれぞれ $\eta = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$ が最適だと言われている [18]．

この損失関数のパラメータ微分を各層で行うため，並列計算を行うとバッチ処理できて高速に学習ができる．そのため並列演算を得意とする GPU を用いて学習を行う．

2.2.2 過学習とその抑制

ディープラーニングでは過学習 (overfitting) と呼ばれる問題が多く起こる．過学習とは，訓練データに対してのみ適応し過ぎてしまい，訓練データに含まれないテストデータには精度が出ない状態を指す．過学習は，大量にパラメータを持つ表現力の高いモデルであることや，訓練データが少ないことなどが原因で起こる．ここでは過学習を抑制するテクニックを述べる．

Dropout

Dropout[19] は，ネットワークのノードをランダムに消去しながら学習する手法である．訓練時に隠れ層のノードを毎回ランダムに選択し，そのノードの出力を 0 にする．そしてテスト時には全てのノードを活性化させ，信号を伝達させる．

Dropout は，学習時にノードをランダムに消去することで，毎回異なるモデルを学習していると解釈でき，アンサンブル学習と同じ効果を擬似的に 1 つのネットワークで実現していると考えられる．アンサンブル学習とは，弱識別器を複数合わせて 1 つの強力な識別器とする手法で，現在でも有効な手法として用いられている．

Batch Normalization

Batch Normalization[20] は，ネットワークにおける各層での活性化後の出力 (アクティベーション) の分布を，適度な広がりを持つように調整する手法である．Batch Normalization では学習を行う際のミニバッチを単位として，ミニバッチごとに正規化を行う (Eq. (2.15)).

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.15}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B) \tag{2.16}$$

$$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.17}$$

ミニバッチとして $B = \{x_1, x_2, \cdots, x_m\}$ という m 個の入力データの集合に対して，平均 μ_B，分散 σ_B^2 を求め，入力データを平均 0 分散 1 となるように正規化を行う．ϵ は 0 で除算されることを防ぐもので，極小の値を用いる．

Batch Normalization を用いると，学習を速く進行させることができる，初期値にそれほど依存しない，過学習を抑制できるといった効果が期待できる．

2.3　画像認識と深層学習

Deep Learning とは Deep Neural Network(DNN) を指すことが多い．この"Deep"とは，ニューラルネットワークの層が深いことに由来している．

Figure 2.5 に画像認識タスクの精度の近年の推移を示す．これは ImageNet Large Scale Visual Recognition Challenge(ILSVRC) と呼ばれる世界的な画像認識のコンペティションである (2010 年から始まった)．カテゴリ数は 1000 クラスで，画像枚数は 120 万枚の訓練データと 15 万枚のテストデータが用意されている．

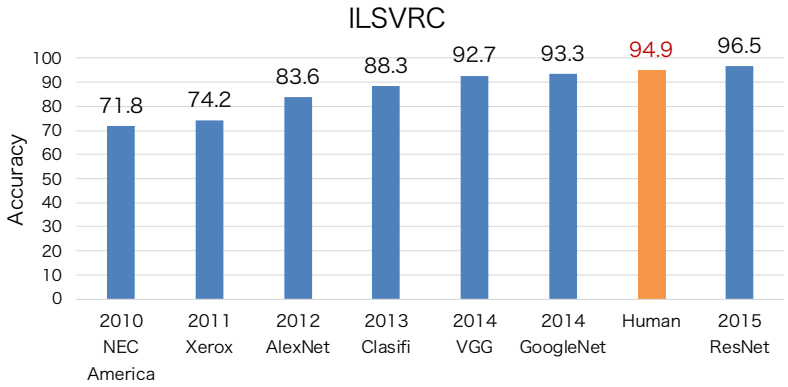


Figure 2.5: Transition of accuracy of image recognition on ILSVRC

2011 年と 2012 年は約 10% もの大差で AlexNet[21] が優勝している．これがディープラーニングの始まりである．AlexNet は 5 つの畳み込み層と 3 つの全結合層を持っている．2014 年には VGGNet[22] や GoogLeNet[23] が 9 割の精度を超えた．VGGNet は AlexNet（8 層）よりさらに深い構造（19 層）であり，GoogLeNet では 22 層もある．そして 2015 年には ResNet[24] が人間の精度をも超える認識精度を達成した．ResNet は GoogLeNet よりもさらに深く 152 層もある．CNN を複数回かけて検出を行う場合，CNN の浅い層では空間分解能はあるが抽象的な情報が少なく，深い層では意味論的な情報は取得できる（ポーズ，変形など）が空間分解能が小さいため幾何学的な情報が失われるといった特徴がある．

画像処理におけるディープラーニングでは大きく 3 つのタスクがあり，それぞれ，クラス分類，物体検出，セグメンテーションである．ImageNet はクラス分類のタスクであったが，近年ではより難しい物体検出やセグメンテーションのタスクを含む MSCOCO というデータセットが主流になってきている．

2.3.1 Classification（クラス分類）

クラス分類は画像に写っている物体が"dog", "airplane", "bird", "toy"など事前に定義されたラベルのどれに該当するかを識別するタスクである．ILSVRC はクラス分類のコンペである．この識別は，事前に定義されたラベルの特徴ベクトルと入力画像の特徴ベクトルの距離計算を行い，距離値が小さければ同一，そうでなければ否と判定する．深層学習では同一人物のペア画像間の距離値が小さく，他人の画像との距離値が大きくなるような損失関数を設計しネットワークを構築することでクラス分類問題を解いている．上述した AlexNet や VGGNet，GoogLeNet，ResNet などはクラス分類

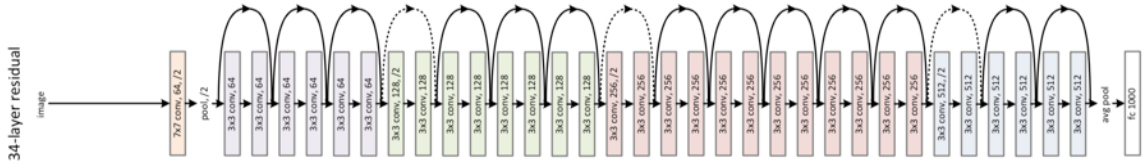


Figure 2.6: Architecture of ResNet34[24].

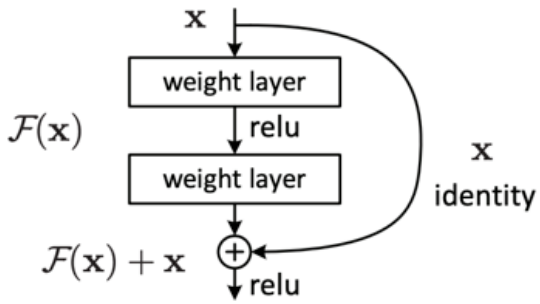


Figure 2.7: Residual block: Skip Connection[24].

を解くためのアーキテクチャである．画像認識タスクにおいてクラス分類は後述する物体検出やセグメンテーションにおいて基幹となるタスクであるため，ResNet などクラス分類を解くネットワーク必ずベースとして使われ Backbone と呼ばれる．

ここでは，Backbone としてよく用いられる ResNet について説明する．Figure 2.6 に ResNet のアーキテクチャを示す．クラス分類で使われる特徴抽出器 (Feature Extraction Layer) と識別器 (Classifier) に分けて解釈できる．畳み込み層で入力画像から特徴を抽出し，全結合層でクラス識別を行う．ResNet 以前では特徴抽出器である畳み込み層が過学習などの問題によって深くできず，良い特徴を得られなかったためどんな識別器を用いても高い精度での識別には限界があった．しかし，ResNet で提案された Skip connection によって層を深くしても過学習を抑えられるようになり，劇的に改善した．Skip connection とは Figure 2.7 に示すように，前の層の出力を次の層の出力と足し合わせる結合である．例えば前の層が良い特徴であって，次の畳み込みによって失われてしまっても，Skip connection によって良い特徴を保ったまま次の層へ進めることができるようになる．言い換えると Skip された層が有用であるかどうかを自動で判断できるようになる．

2.3.2 Object Detection（物体検出）

物体検出とは Bounding Box で物体の位置とその物体の種類を特定する方法である．物体検出で主流であるのは Faster R-CNN である [25]．Figure 2.8 に示すように，Faster R-CNN では特徴抽出器である CNN と Region Proposal Network(RPN) の 2 つからなる．CNN で得た特徴マップの各点を中心とした Bounding Box(BBox) を，1 辺の長さおよび縦横比を変えて複数用意し RPN に入力する．RPN では入力された BBox に物体か背景かを判定し，物体ならば Ground truth からどれだけずれているかを学習する．これにより入力画像内のどこに物体がいるかという物体候補領域 (Region of Interest: RoI) を決定する．CNN によって抽出した特徴マップと求めた RoI を照らし合わせ，該当する特徴マップをプーリングして物体候補ごとの固定サイズの特徴マップを作る（RoI pooling）．RoI pooling された特徴マップから識別器によってどのクラスに属するのか決定する．以上のようにして，Faster R-CNN ではモデル全体を End-to-End で学習できる．

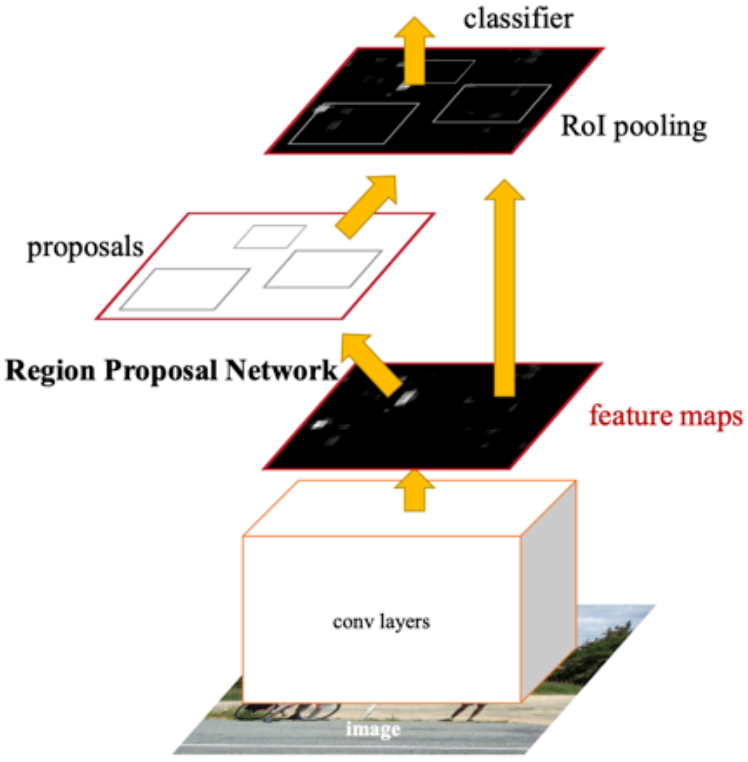


Figure 2.8: Architecture of Faster R-CNN[25].

2.3.3 Segmentation（セグメンテーション）

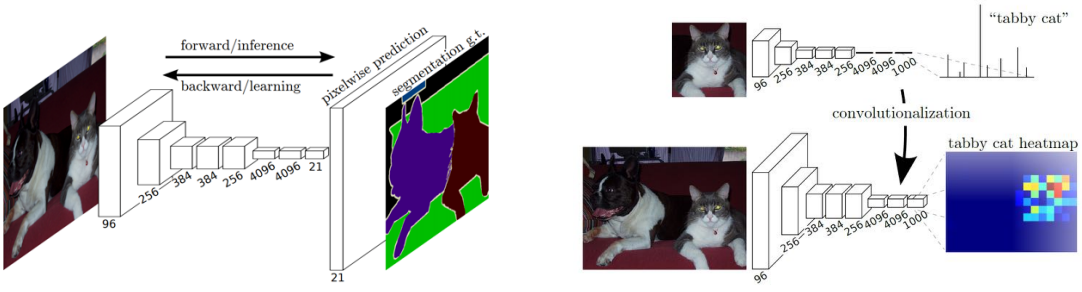
セグメンテーションには Semantic Segmentation（セマンティックセグメンテーション）と Instance Segmentation（インスタンスセグメンテーション）の 2 つがある．

セマンティックセグメンテーション

セマンティックセグメンテーションとは画像をピクセルレベルで認識することである．画像内の各画素をオブジェクトクラスに割り当てる手法である．

セマンティックセグメンテーションにおいて最初に CNN を使った手法は Fully Convolutional Network(FCN) である [26]．これはクラス分類に使われるネットワークの全結合層を畳み込み層に置き換えて，最終出力を 2 次元マップにするネットワークである（Figure 2.9）．Figure 2.9(b) のように，これによりピクセル単位でクラス分類の結果がヒートマップとして出力される．

特徴マップのサイズは Max Pooling を経て小さくなっているため，入力画像に対して特徴マップのサイズは小さくなっている．そのためこれを Up sampling して入力画像のサイズまで拡大する．Up sampling には逆畳み込み（Deconvolution）を行う．逆畳み込みとは畳み込みの逆操作によって特徴マップから畳み込む前の画像・特徴を復元するのではなく，元となる特徴マップを padding や diliate によって拡大してから畳み込む操作である．数学的にはカーネルの転置で畳み込むため転置畳み込み（Transposed Convolution）とも呼ばれる．また FCN では途中の畳み込み層の出力も使い Pooling によって失われた細かい位置情報も利用しており，これによって出力画像がぼやけずに物体の詳細な情報を捉えたセグメンテーションが可能となる．



(a) FCN framework[26].

(b) Transforming fully connected layers into convolution layers[26].

Figure 2.9: Architecture of FCN.

インスタンスセグメンテーション

インスタンスセグメンテーションとは同じクラスの物体も別として認識することである．まず入力画像に対して物体検出を行い，検出した BBox 内でセグメンテーションを行う．すなわち，インスタンスセグメンテーションは物体検出とセマンティックセグメンテーションのハイブリッドと言える．インスタンスセグメンテーションでよく使われる手法は Mask R-CNN である [27]．

Mask R-CNN は Faster R-CNN をベースとしたアーキテクチャで，RP マップで Detection & Classification branch と Segmentation branch に分岐する．Detection & Classification branch は Faster R-CNN とほぼ同様であるため説明は割愛する．Segmentation branch では FCN[26] と同じ構造であり，畳み込みを複数回行い最後に逆畳み込みを処理することで提案候補領域におけるマスクを生成する．

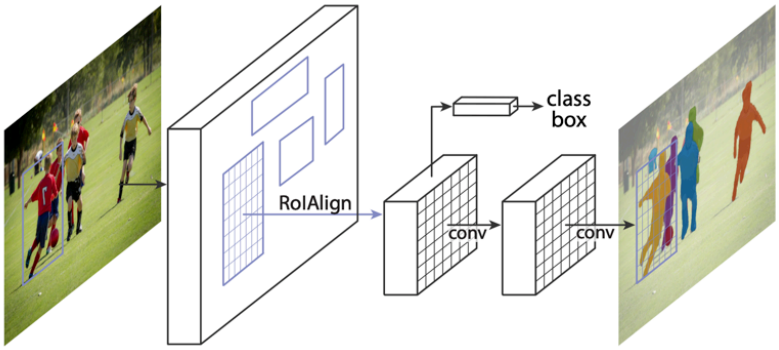


Figure 2.10: Architecture of Mask R-CNN[27].

2.3.4 画像認識における評価指標

機械学習における評価指標を説明する上で混同行列というものを用いる．Table 2.1 に 2 クラス分類における混同行列を示す．混同行列の要素は True は正解で False は不正解を示しており，Positive は予測が真，Negative は予測が偽を出力することを表している．すなわち，TP は正解ラベルを正しく予測できた，TN は正解ラベルじゃないものを正解じゃないと予測できたことを示しており，一方 FP は正解じゃないものを正解と予測してしまった，FN は正解ラベルを正解と予測できなかったことを示している．この混同行列の要素を用いて，Eq. (2.18) に示す 3 つの評価指標を定義する．

Table 2.1: Confusion Matrix on 2 class classification.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive(TP)	False Positive(FP)
	Negative	False Negative(FN)	True Negative(TN)

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + FP + TN + FN} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \end{aligned} \tag{2.18}$$

Accuracy は正解率と呼ばれ，モデルの予測結果全体と答えがどれだけ一致しているか示す割合である．一般にモデルの精度というと Accuracy を指す事が多い．Precision は適合率と呼ばれ，モデルが真と予測結果全体の中で実際に正解である割合である．つまり誤検知にどれだけ強いかを示す．Recall は再現率と呼ばれ，モデルの予測で出力されるもののうち実際に正解である割合である．つまり，見逃しにどれだけ強いかを示す．

他クラス分類では各クラスの Precision や Recall をクラス数で平均して Average Precision(AP) や Average Recall(AR) とする．

一方，物体検出やセグメンテーションにおける評価指数としては IoU(Intersection over Union) を使う．Figure 2.11 に示すように，IoU は予測結果と教師ラベルがどれだけ重なっているかを表す．数式では混同行列の要素を使って

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{2.19}$$

と表す．

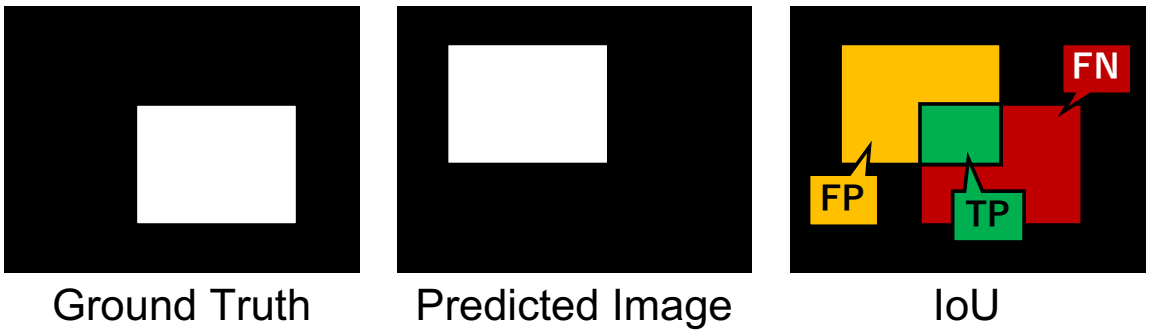


Figure 2.11: Example image of IoU.

また，物体検出やセグメンテーションでは pixel-wise で予測結果が出力されるため，AP や AR をさらに画素全体で平均を取って mean Avereage Precision(mAP) や mean Average Recall(mAR) とする．

なお，MSCOCO と呼ばれるデータセット [28] では COCO AP，COCO AR と呼ばれる独自の評価指標を用いている．Figure 2.12 に COCO の評価指標のまとめを示す．mAP や AP の決め方はひとつでなく，IoU の閾値によってそれを物体と認めるか背景とするかによって変わってくる．COCO の mAP は 101point の補間 AP であり，COCO では複数の IoU を基準とする mAP を導出し，それを平均したものを AP としたものも採用している．IoU = 0.50 : 0.95 は 0.05 刻みに 0.50–0.95 まで基準 IoU を変えた mAP の平均である．COCO AR は画像ごとの検出数 (maxDets) を 1,10,100 のいずれかに固定した時の Recall を測定する．また，AP，AR ともに small medium large に分けたオブジェクトの大きさ (Area) ごとの値も測定している．

```
Average Precision (AP):
  AP                % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50        % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75        % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small          % AP for small objects: area < 32^2
  AP^medium         % AP for medium objects: 32^2 < area < 96^2
  AP^large          % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1          % AR given 1 detection per image
  AR^max=10         % AR given 10 detections per image
  AR^max=100        % AR given 100 detections per image
AR Across Scales:
  AR^small          % AR for small objects: area < 32^2
  AR^medium         % AR for medium objects: 32^2 < area < 96^2
  AR^large          % AR for large objects: area > 96^2
```

Figure 2.12: Other metrics collected for the COCO dataset[29]

他にもいくつか細かい指標があるが，本研究では Figure 2.12 に示す AP と $AR^{max=100}$ を使用した．

2.4 強化学習

今まで述べてきたことは全て教師あり学習と呼ばれる手法であり，機械学習の 1 種である．ここからは強化学習という機械学習について述べる．

2.4.1 マルコフ決定過程

Figure 2.13 に強化学習の概念図を示す．強化学習は，エージェントと呼ばれるプレーヤーが，与えられた環境と相互作用し，探索と知識の利用を行って目標を達成する方法を求める．強化学習は教師あり学習に似ているが，教師による明確な「答え」は提示されない．代わりに「行動の選択肢」と「報酬」が提示される．ここで，強化学習においての報酬は「各行動」に対してではなく，「連続した行動の結果」に対して与えられる．この報酬を最大化することが強化学習の目的である．

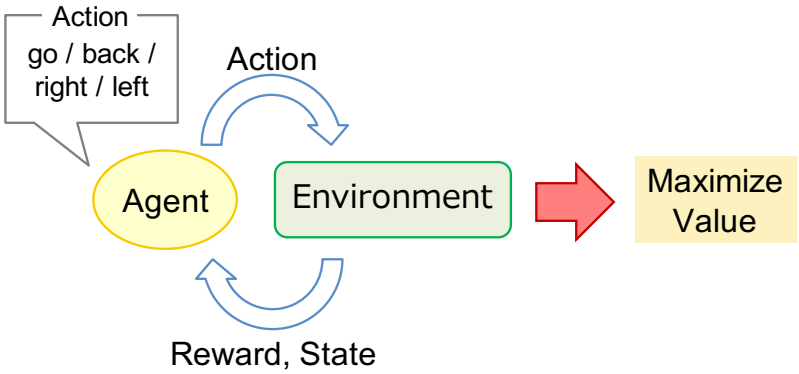


Figure 2.13: Framework of Reinforcement learning

この Figure 2.13 を表す数理モデルとして，マルコフ決定過程 (Markov Decision Process: MDP) がある．MDP は時刻 t における状態を s_t，行動を a_t，報酬を r_t，状態 s_t における行動を返す方策 $\pi(a_t|s_t)$，次の状態 s_{t+1} へ移る状態遷移確率 $P(s_{t+1}|s_t, a_t)$ によって記述される確率過程である．マルコフというのはマルコフ性という意味で，次の行動 s_{t+1} には現在の状態 s_t しか関与しないという性質を表している．1 回の報酬を指標にに最適な行動を求めてはすぐに局所解に落ち着いてしまい，後に来る大きな報

酬を得ることができないため，よい指標にはならない．そこで，ある期間で得られた累積の報酬 (収益)R_t を導入する．

$$R_t = \sum_{\tau=0}^{\infty} \gamma^\tau r_{t+1+\tau} \tag{2.20}$$

$$= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \tag{2.21}$$

と表され，これを割引報酬和 (discounted total reward) と呼ぶ．ここで $\gamma(0 < \gamma < 1)$ は割引率と呼び，未来の報酬をどの程度割り引くかを決めている．概ね 1 に近い値を用いる．この収益は，機関の中で報酬の和を取るため，ある行動が期間内における報酬の獲得に結び付いていた場合には，それがずっと後のことであっても，指標に反映することができる．

しかし収益は，区間の開始時点での状態に依存して，相互作用の内容が確率的に決定されるため，収益も確率的に変動してしまう．そのため，状態 s と行動 a を条件として収益の期待値を取り，これを行動価値 (action value) と呼ぶ．行動価値関数 $Q(s, a)$ は，ある状態から方策 π に従って行動を決定したときに得られる収益の期待値であるから，

$$Q^\pi(s_t, a_t) = \mathbb{E}\left[R_{t+1} \middle| s_t, a_t\right] \tag{2.22}$$

$$= \mathop{\mathbb{E}}_{s_{t+1}, a_{t+1}}\left[r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) \middle| s_t, a_t\right] \tag{2.23}$$

で表される．これを指標として，最適な方策 π^* に基づく最適行動価値関数 Q^* を求める．

最適価値関数を求める方法として，Temporal-Difference Learning(TD 学習) が用いられる．TD 学習は，環境のモデルを使わず経験的に学習を行い，最終結果を待たずに評価を途中で更新する．TD 学習は次のステップを待ち価値関数を更新する．

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.24}$$

$\alpha(0 < \alpha < 1)$ は学習率である．ここで，$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$ を TD 誤差と呼び，収束からの離れ具合を示す．学習が収束したとすれば TD 誤差が 0 になり，その収束値が最適価値関数となる．

2.4.2 Q-Learning

Q 学習 [30] と呼ばれる手法では TD 誤差に基づいて，次のように価値関数 Q を更新する．

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.25}$$

更新式で実際に採用した行動 a' を使っていない (方策に関わらず価値関数の最大値を与える行動を使っている) ことが特徴である．ここで，方策 π としては，まずはランダムに行動することで状態と行動のセット (s,a) を蓄積していき，その中で行動価値関数が最大となるような行動を取る．これを ϵ-greedy 方策と呼ぶ．確率 ϵ でランダムに行動することで，局所解に陥らずに最大値へ収束することができる．

端的に言えば，「どの状態で、どう行動したら、どういう報酬が得られるのか」を明らかすることが Q-Learning である．実際はこの報酬の見込みの表形式で表し，これを Q-Table と呼ぶ．

Q-Learning は Q-Table が求まれば良いが，環境から得られる状態の次元や行動の次元が大きいと求められない場合がある．そのため，Q-Learning を設計する場合は，環境から特徴を取り出し低次元にする必要がある．また，行動も連続的な行動ではなく，離散的な行動選択に絞る必要がある．

2.5 機械学習のロボット制御への応用

強化学習は戦略が必要なタスクに適用できる．ロボット制御への応用も多く研究されている．例えばロボットによるピッキングタスクを行うとすると，多くのパラメータを制御する必要があり使用する環境ごとにチューニングする必要があるが，強化学習を用いることで最適な行動をするようにパラメータを獲得でき，かつどんな環境に対してもロバストになる．ここではどのように強化学習を適用するか，先行研究を追いながら説明する．

Gu らは人間の手を介さずドアを開ける動作を初めて成功させた [31](Figure 2.14)．ここでは，Q-Learning をベースとした Normalized Advantage Function(NAF) という手法 [32] を用いている．観測する状態として，7 つの間接角度とその時間微分，ハンドエフェクタ・手・ドアのそれぞれの位置，手・ドアのそれぞれの角度の計 25 次元を入力し，Action は連続値で出力する．シミュレーション上で 4 機で非同期分散学習させた後，実世界では最大 2 機で非同期分散学習を行った．実世界では成功率 100% にな

Figure 2.14: Two robots learning to open doors using asynchronous NAF. The final policy learned with two workers could achieve a 100% success rate on the task across 20 consecutive trials[31].

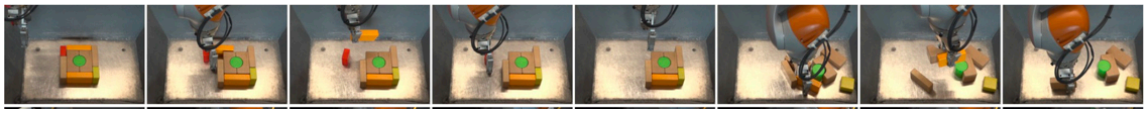


Figure 2.15: Pregrasp manipulation (a, b), grasp readjustment (c, d), grasping dynamic objects and recovery from perturba- tions (e, f), and grasping in clutter (g, h)[33].Video: `https://sites.google.com/view/qtopt`

るまでに 1 機では 4 時間，2 機では 2.5 時間という短い時間で達成した．シミュレーションから実世界への転移は難しいことが多く，Gu らは成功判定を緩めることで解決した．

また Dmitry らは，Q-Learning を改良した QT-opt という手法を提案した [33](Figure 2.15)．これはより汎用性の高いタスクをロボットに学習させることを目的として，1000 種類以上の物体を把持できるような学習を行った．QT-Opt を実行すると、より多くのオフラインデータが蓄積され，より良いデータを収集できるように，より良いモデルを訓練できるようになる．このアプローチを使い，ロボットで物を掴む事を学習させるために 7 つの実世界のロボットを使用し，4 ヶ月間にわたって合計 800 時間学習させた．学習を早めるために，最初は人間が手動で設計した 15〜30% 程度の成功率のポリシーを使用して開始した．状態として RGB のカメラ画像を取得し，行動として腕とグリッパの移動方法を返す．報酬は成功したら +1，失敗したら 0 としている．また，失敗した時の報酬を-0.05 とすると学習を早めることができる．シンプルな仮定ではあるが，QT-opt は目標物体を掴むために周りの邪魔な物体を弾いたり，掴みやすいように物体を倒したりすることを学習した．また，従来の手法より少ない学習データで高い成功率を達成した．

以上のように，強化学習での学習には多くの試行回数及び時間がかかることがわか

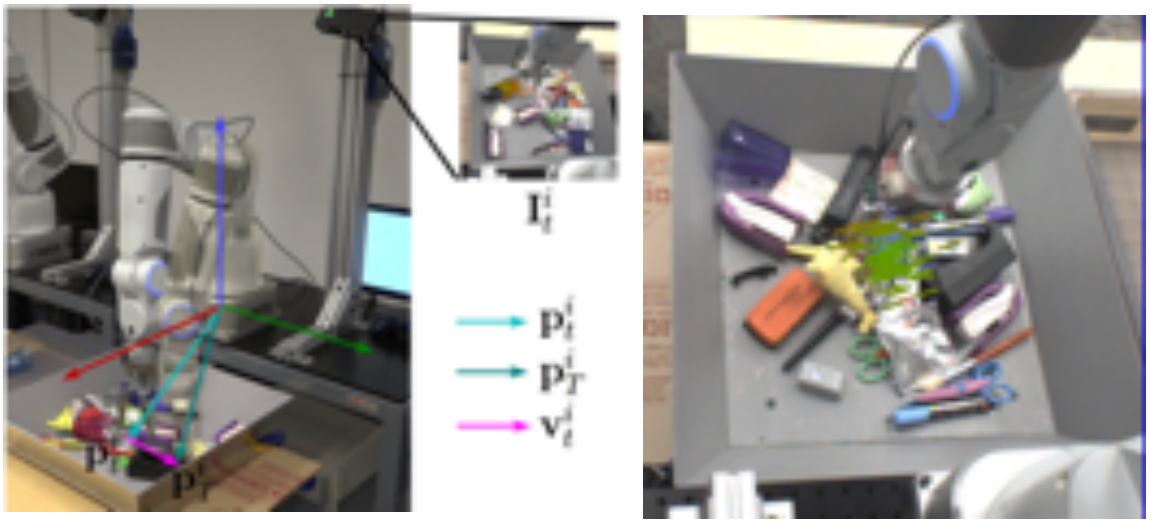


Figure 2.16: Colors indicate their probabilities of success: green is 1.0 and red is 0.0[34].

る．教師あり学習と異なり，エージェントが用意された環境から教師信号をサンプルする必要があるためである．ロボット制御においては教師データを集めることが困難な場合があるため強化学習が有効である．

一方で強化学習はそもそも最適化問題として難しく学習が困難な場合が多い．そこで教師あり学習によってロボットを制御する手法もある．

Levine らは初めて vision ベースでグラスピングタスクを成功させた [34](Figure 2.16)．この手法は教師あり学習で画像とモーターのコマンドを入力して把持可能性 (graspability) を出力するネットワークを学習し，その出力にしたがってロボットを動かす手法である．厳密には Q-Learning ではないが Q-Learning とも解釈できると言っている．最大 14 台のロボットで非同期分散学習を行い，800,000 回以上の試行を行った．

第 3 章

試作 1 号機：スマートフォン搭載ハンドの開発

3.1　要求仕様

1 号機では本体の基本デザイン・自律動作・携帯性に重点をおいて設計する．

ハードウェアとして自律移動が可能で物体を把持でき，また小型で携帯可能な重量が要求される．そのため，環境を認識できるセンサ，対象物に接近するためのアクチュエータ，把持を行うためのアクチュエータ，これらを制御するコンピュータが必要である．

ソフトウェアとして，センサ情報を Edge で計算しアクチュエータ制御を行うことが要求される．また持ち運ぶため環境にロバストな制御方式であることが必要である．ただし，消費電力と衝突リスク等を考慮するとアクチュエータの制御は高速に行う必要はなく，数ミリ秒毎の制御とした．

3.2　機構設計・機体デザイン

環境を把握するセンサとして，簡便でかつ情報量の多いカメラを採用した．また自律移動には車輪を採用し，後輪駆動の 3 輪車とした．車輪の駆動には DC モーターを用いた．把持動作にはサーボモータを使用し，1 軸動作を実装した．

上記の画像取得，認識，処理，アクチュエータ制御をすべてスマートフォンでの実現を試みた．ロボットハンドにはスマートフォンを搭載し，スマートフォンのフロントカメラで環境を認識させた．アクチュエータの制御にはワンチップコンピュータを使用し，スマートフォンとシリアル通信を行うことで各アクチュエータをそれぞれ制御した．

これらを考慮して Figure 3.1 に示すようなフレームを 3DCAD（123D Design Autodesk 社を使用）で設計した．また，ロボットハンドの向きは Figure 3.1 に示す，ソフトウェアでの画像（配列）の向きと同様の定義とした．

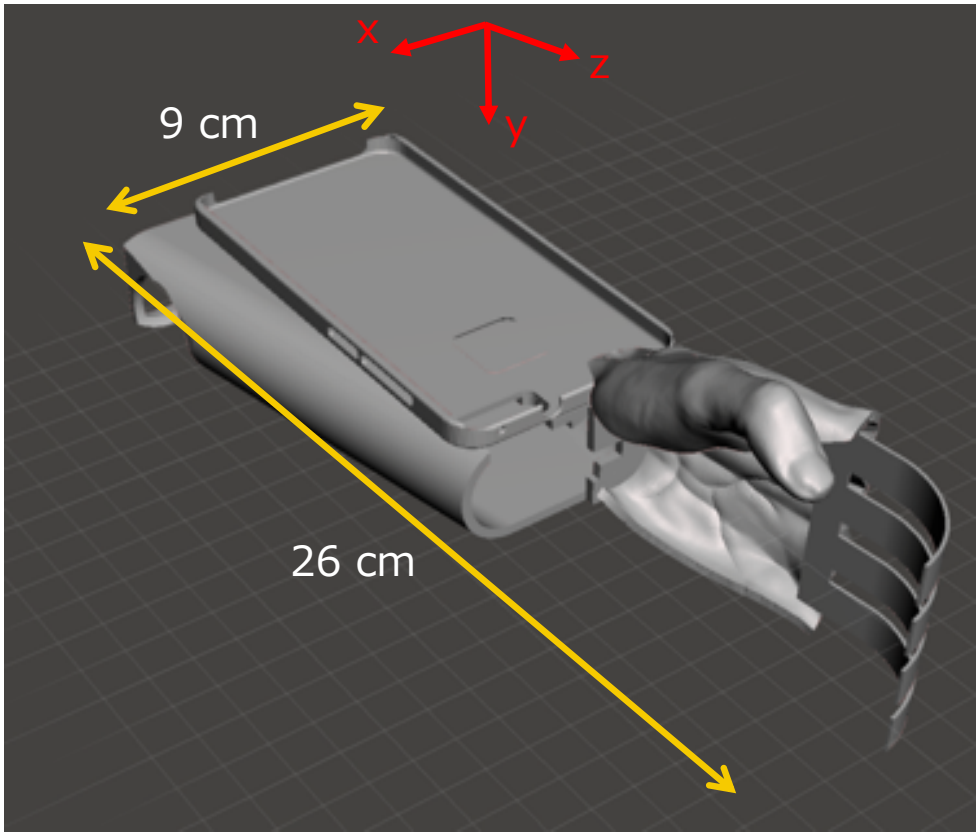


Figure 3.1: 3DCAD image of prototype No.1

3.3 制御アルゴリズム

自律移動のアルゴリズムは強化学習で行った．強化学習は環境に対してロバストであるため，実生活において有用だと考えた．強化学習の最適化としては Q-Learning を用いて学習を行い，方策として ϵ-greedy 方策を使用した．

Figure 3.2 に 1 号機の制御系ブロック図を示す．ロボットハンドの接近動作のフローを述べる．まず，フロントカメラから 480×640 pixel の画像を取得した．画像から対象物を認識するために，計算量が少ない色抽出による対象物のセグメンテーションを行った．取得した画像を HSV 空間に変換し，赤色だけを抽出しターゲットのマスク画像を得た．マスク画像からターゲットの面積（ピクセル数）と画像における重心を求め，面積を画像サイズで規格化した．この面積値と前フレームでの面積値との差分の 2 次元を状態として与えた．行動として前進，後退，右旋回，左旋回の 4 次元を与えた．報酬としてタスクが成功したら +1，1episode 以内で成功できなかったら-1，そ

の他では 0 とした．episode の終了判定に面積値と重心座標を用いて，面積が 40 以上で最接近とし，また重心座標がロボットの中心座標から 10pixel 以内であれば正面に来たと判定した．接近が完了したらサーボモータを動かして対象物を握るようルールベース化した．

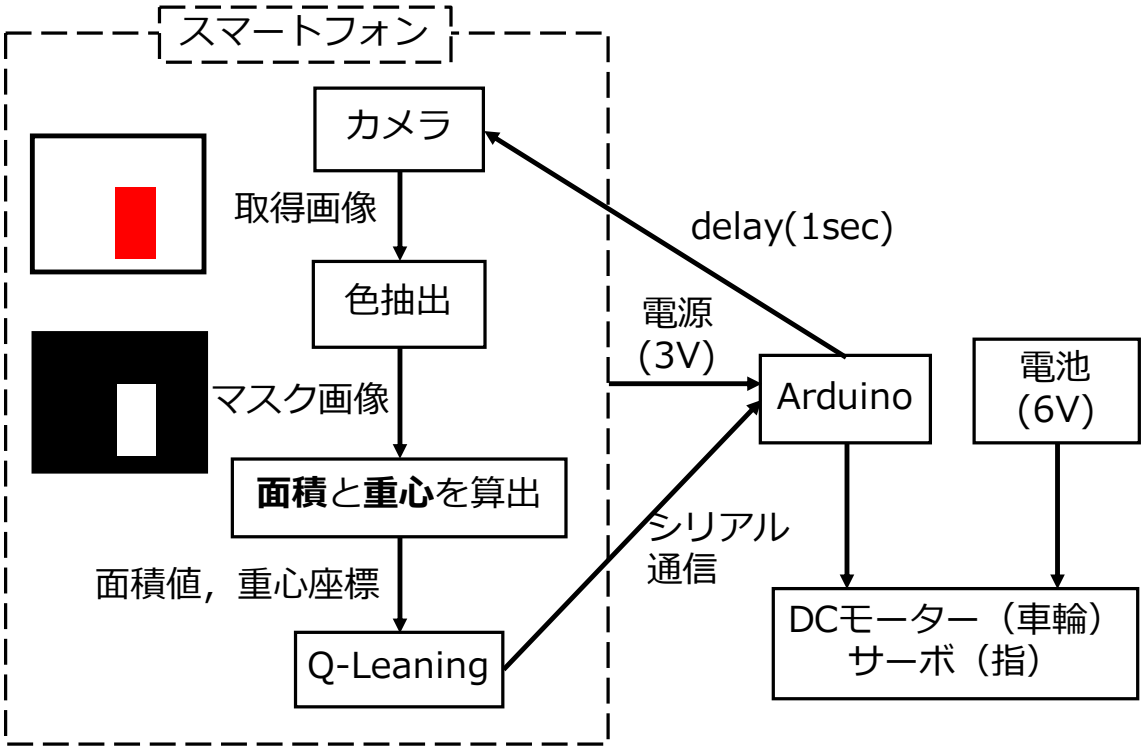


Figure 3.2: Block diagram of Prototype No.1

1 号機のプログラムは筆者の GitHub[*1]で公開している．

3.4 実機作製

1 号機作製にあたって使用した部品を Table 3.1 まとめた．

[*1] https://github.com/yumion/onodera-lab/tree/master/JSTfair/robot_hand

Table 3.1: Components of prototype No.1

本体フレーム	polymaker PolyPlus PLA Filament 750g Black
指	熱可塑性炭素繊維強化プラスティック（Carbon Fiber Reinforced Thermo Plastics: CFRTP）
スマートフォン	HUAWEI P10 lite
DC モーター	STL ギヤモータ 栄 42D 長軸型
サーボモータ	GWS サーボ MICRO/2BBMG/FP（フタバ）
タイヤ	TAMIYA 製スパイクタイヤ
コンピュータ	ArduinoProMini + FTDI
バッテリー	単 4 電池 ×4 本 + 電池パック

Figure 3.3 に作製したロボットハンドの外観を示す．ロボットハンドのフレームは簡便さと軽さを考慮し 3D プリンタ（TITAN GENKEI 社）で造形を行った．ハンドの指には熱可塑性炭素繊維強化プラスティック（Carbon Fiber Reinforced Thermo Plastics: CFRTP）を使用し，炭素繊維のバネ性能を利用して対象物の把持性能向上を期待した．スマートフォンは腕の上部に装着し着脱ができるようにした．スマートフォンのフロントカメラに鏡を角度 45 度傾けてで置くことで z 軸方向（スマートフォンの上側方向）を捉えることができるようにした．また，腕部分にバッテリーとタイヤ，そして Arduino を含む回路を収め，上から見るとスマートフォンと手のみが見えるように工夫して組み立てた．1 号機の総重量は 473.82 g であった．

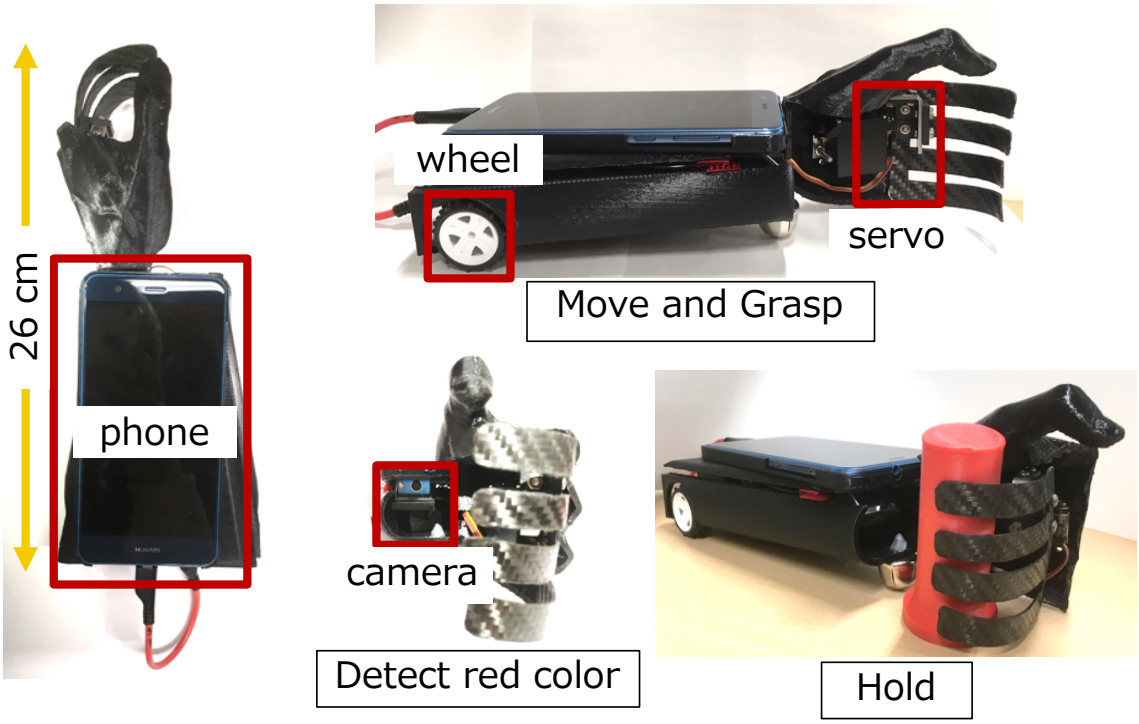


Figure 3.3: Appearance of prototype No.1. Weight is 473.82 g.

3.4.1 学習方法

1 号機の行うタスクとして接近タスクを行った．実機で 2 日間に渡り約 300episodes 学習させた．Q-learning のハイパーパラメータは，割引率は 0.99，学習率は 0.5，状態は 6 つの状態に離散化した．

3.4.2 結果

学習の中で成功した episode におけるロボットの挙動を Figure 3.4 に示す．まず，旋回することで周囲を探索する．カメラにターゲットを捉えられたらそれに向かって接近し，一定の距離まで近づいたら把持を行う．このような流れで学習が進んでいることが示唆された．

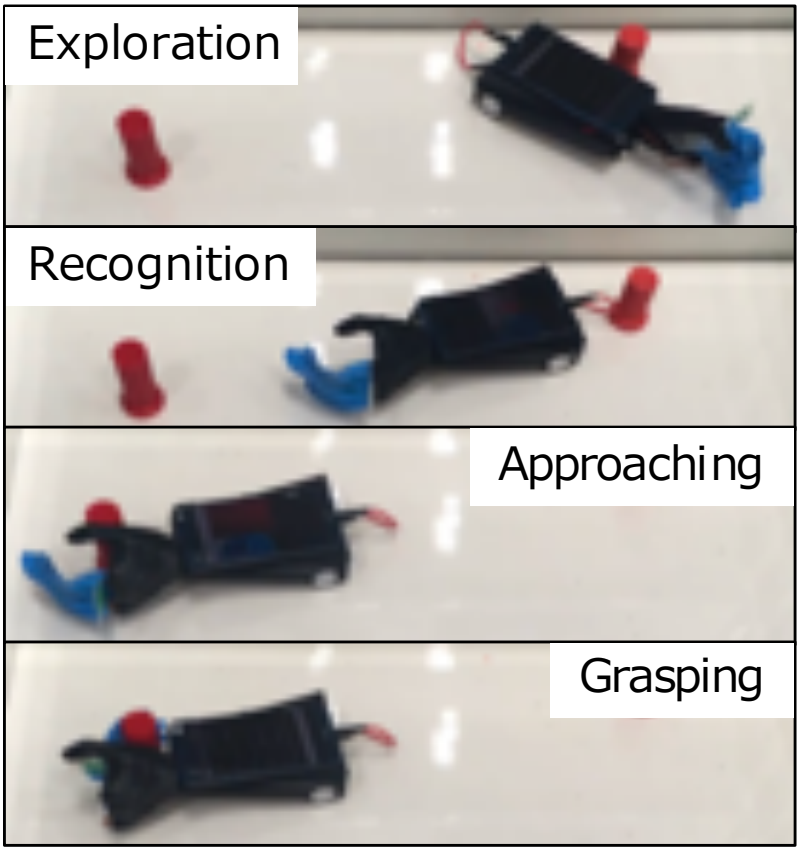


Figure 3.4: Demonstration of prototype No.1 at sucess episode.

しかしカメラのフレームレート（frame per second: fps）が低く 1fps 程度であり，各 step における行動が 1 秒程度持続してしまい，同じ場所を旋回しつづける行動が問題であった．fps が低い原因として，スマートフォンのカメラにプロテクトがかかっておりビデオが使えず，カメラの単写を使用せざるを得ないことが挙げられる．またスマートフォンの演算能力ではより高次元の状態や行動を入出力とした強化学習は難しいため，"曲がりながら進む"といった前進と旋回の組み合わせができなかった．

3.5 物理シミュレーション

強化学習では学習に多くの時間がかかる．また実機ではバッテリー残量や壁にぶつかった際に位置を人の手で直す必要がある．そこでより学習を進めかつ自動で学習ループを回すために物理シミュレーションを用いた．これにより学習過程を常に記録・参照することもできる．

3.5.1　評価手法

シミュレーション物理エンジンとして Bullet Physics Engine を用いた．実装においては API が Python で用意されている PyBullet[35] を使用した．Figure 3.5 に CAD で作製したロボットハンドをシミュレーション環境にレンダリングした画像を示す．

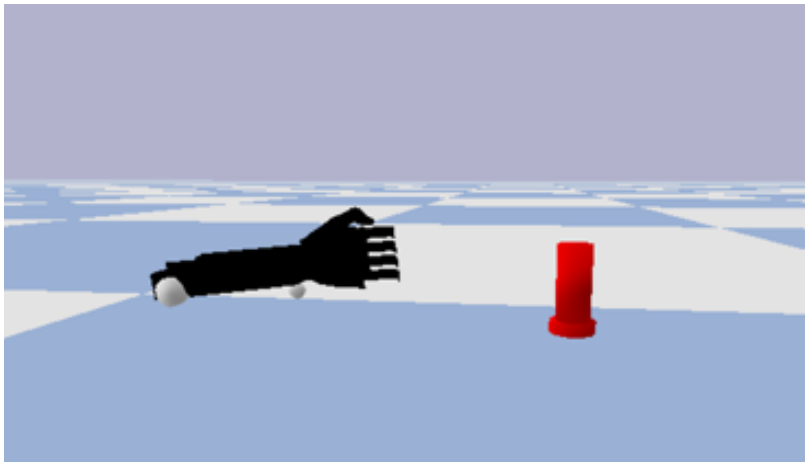


Figure 3.5: Environment of bullet physics simulation.

ターゲットをランダムに配置し，観測状態と報酬を変えて学習がどのように進むかを実験した．タスクとしては接近のみを行い，ルールベース化してある把持は省略した．性能をテストする際，ϵ-greedy 方策の探索する確率を 0 とし，greedy 方策でテストする．そして 10episode ごとに報酬の総和を計算し，1episode あたりの報酬として平均をとった．

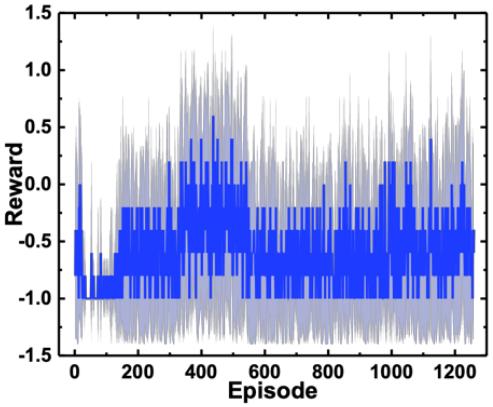
3.5.2　結果

まず，状態としてカメラから認識できるターゲットの面積値（最大を 100 に規格化），及びそのフレーム間差分の 2 次元を与えた．また報酬として，タスクが成功したら +1，1episode 以内に成功しなかったら-1，各 step では 0 とし，学習を行った（Figure 3.6(a)）．Episode が経過しても報酬に変化がないことから，学習が収束していないことがわかる．この原因としては観測が良くないかまたは報酬設計が悪いかの 2 つの可能性が考えられる．Agent の行動は環境から得られる報酬に依ることを考慮すると，多くの episode 学習を行っても収束しないということは観測状態が良くないと

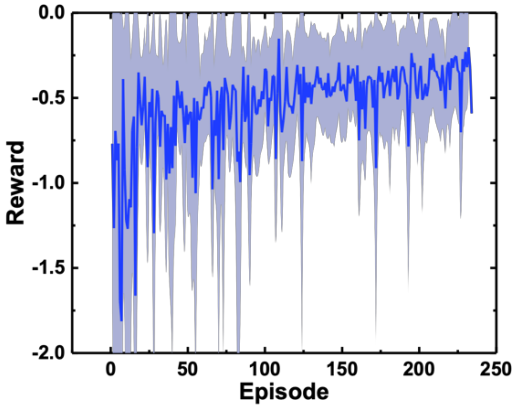
考えられる．また，報酬は連続値で与える方が各 episode ごとではなく各 step ごとにパラメータ更新が行えるため学習がスムーズに進むので，報酬を連続値に変えた．

次に，状態としてロボットハンドとターゲットとの相対位置 (x, y) 座標の 2 次元を与えた．また報酬として，ターゲットとの距離を与え，学習を行った（Figure 3.6(b)）．Figure 3.6(a) とは挙動が異なり，初めの数 10episode で急激に報酬が増加し，その後横ばいとなっていることがわかる．Episode を重ねても報酬は飽和しているため，学習が収束していることがわかる．このパラメータでレンダリングして確認してみると，学習が成功していることが分かった．

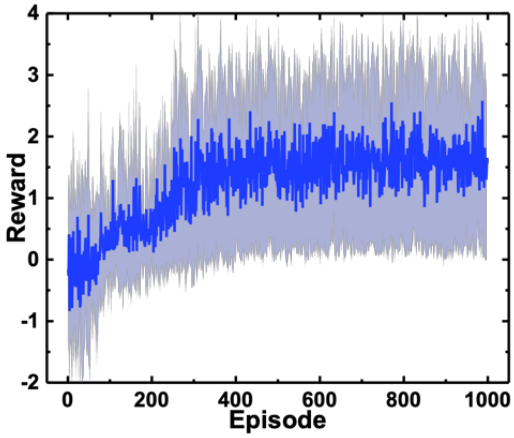
また，状態としては先ほどと同様に相対位置 (x, y) 座標の 2 次元を与え，報酬としてターゲットの面積値を与え学習を行った（Figure 3.6(c)）．Episode が進むとなだらかに報酬が増加し，400episode 付近から報酬が飽和していることがわかる．このことから学習が収束していると言える．このパラメータでレンダリングして確認してみると，確かに学習が成功していることが分かった．



(a) State: area of target, time difference of the area; Reward: success=1, failure=-1, others=0.



(b) State: relative position; Reward: distance between agent and target.



(c) State: relative position; Reward: area of target.

Figure 3.6: Learning curve of each state and reward at simulation.

3.5.3 考察

結果を踏まえると，報酬はタスクの成否にあまり関係なく，環境を正しく観測することが重要だということが分かった．Figure 3.6(a) の状態は言い換えると一人称視点であり，Figure 3.6(b)，Figure 3.6(c) は環境を俯瞰して見る三人称視点である．三人称視点では自分の周囲の環境全体を観測することができるが，一人称視点では自分が向

いている方向しか観測できない．すなわち，探索をしていく中でターゲットを捉える機会が必然的に少なくなり，報酬による行動の評価が難しくなる．したがって，一人称視点では中々学習が進まず，三人称視点ではスムーズに報酬が飽和し，学習が完了したと考えられる．

3.6 まとめ

自己完結型の自律駆動ロボットハンドを実現するため，全てのシステムを人間の手のサイズに収容することを目指してスマートフォンの CPU，アクチュエータ制御インターフェースとスマートフォンのカメラ（環境認識）を用いたロボットハンドを作製した．スマートフォンの計算リソースを考慮し，処理の軽い色抽出によって指定した色（赤色）の物体を認識し，接近し，把持するという一連の動作を行システムを開発し，ターゲットに接近し把持することに成功した．実機での学習とシミュレーション環境での学習を両方行い，外部環境をどう状態に落とし込み状態とするかが学習に大きく関わることを示した．そして，接近タスクにおいてはロボットハンド視点では学習が収束せず，ロボットハンド視点と共にターゲットを俯瞰する視点を併用して観測すると正しく学習できることがわかった．

1 号機の課題として以下が挙げられる．

- 機械的な自由度が少なく，様々な形状の物体を持ち上げて運ぶことが難しい．
- 様々な種類の物体を識別できない．
- Android のスマートフォンでは Python から制御すると内蔵カメラの動画が使えず，リアルタイム性に欠ける．
- スマートフォンの計算リソースでは重い画像処理が不可能である．

今後は，物体を把持した後に次の行為を実行させるため，より自由度を上げ複雑な動作を可能にするロボットハンドを開発する事の必要性を実感した．ロボットの制御精度と動作速度向上には，環境を認識するセンサとして，環境を俯瞰する位置に定点カメラを設置したり，作動環境内に測距センサを配置する必要がある．加えて，把持対象物が複数ある場合に，使用者が意図する物体を正しく認識し把持できる物体識別性能が重要になってくる．

3.7　ヒアリングの実施

茨城県立大学付属病院リハビリテーション科にて義手使用患者へヒアリングを行い，ロボットハンド 1 号機の感想をいただいた．

- "軽いと感じた"
- "家に帰ったらテーブルの上で作業することが多いから，ロボットハンドの有用性はあると思う"
- "自分のスマートフォンでできるのが良い"
- "クラウドを使わないから停電になったときでも使えて良い"

このように義手使用患者からも好評であり，本システムには将来性があると考える．

第 4 章

試作 2 号機：インスタンス認識ハンドの開発

4.1　要求仕様

1 号機の課題の中で特に，把持機構の自由度が小さい事と，物体の正確な識別が不可能である問題は実用化の障害となる．そこで 2 号機ではこれら 2 点の課題を克服したロボットハンドを開発する．

ハードウェアとして，1 号機では対象に近づきグリップすることをゴールとしたが，実使用を考えると物を持ち帰って次の動作をすることが要求される．そこで 2 号機では手首機構を重視して，より多彩な動きを実現するためアクチュエータを増やし，把持機構の自由度を上げる必要がある．グリッパとして，指定した点で掴む，またグリッパの中心軸で左右対称に掴む，という動作を可能にする機構が必要である．また，掴む時の把持点決定及び指定した点を掴めるようアクチュエータで調整できることが要求される．そして掴んだ後の物体の滑り防止機構が必要である．物体をピックアップするために，環境把握に加えて物体との距離を正確に捉える必要がある．

ソフトウェアとして，1 号機に搭載したスマートフォンでは環境認識に 1 秒を要したため，より高速に演算できる計算リソースが要求される．多自由度化に伴いより多くのアクチュエータを制御するため，スマートフォンではなく GPU を搭載したデバイスで学習・推論を行う必要がある．また，1 号機では色のみの認識であったが，把持対象物を"物体"として認識し，複数種類に拡張して識別できることが要求される．

4.2　機構設計・機体デザイン

環境把握と対象物との距離を測るために RGB カメラとデプスカメラを両方搭載した Intel 社の RealSense を用いた．対象物を持ち上げるために腕に関節を 1 つ加え，地面から垂直に対象物を挙上できるようにした．把持中心点を正確に掴むため，2 指対立タイプの一般的な形状のグリッパとした．把持中心位置の制御では，高精度化のためにラック&ピニオンを用いてサーボモータの回転を水平運動に変換にし，腕を進行方向と垂直な方向にスライドできる機構を搭載した．

以上を考慮して Figure 4.1 に示すようなフレームを 3DCAD（123D Design Autodesk 社を使用）で設計した．また，ロボットハンドの向きは Figure 4.1 に示す，ソフトウェアでの画像（配列）の向きと同様の定義とする．

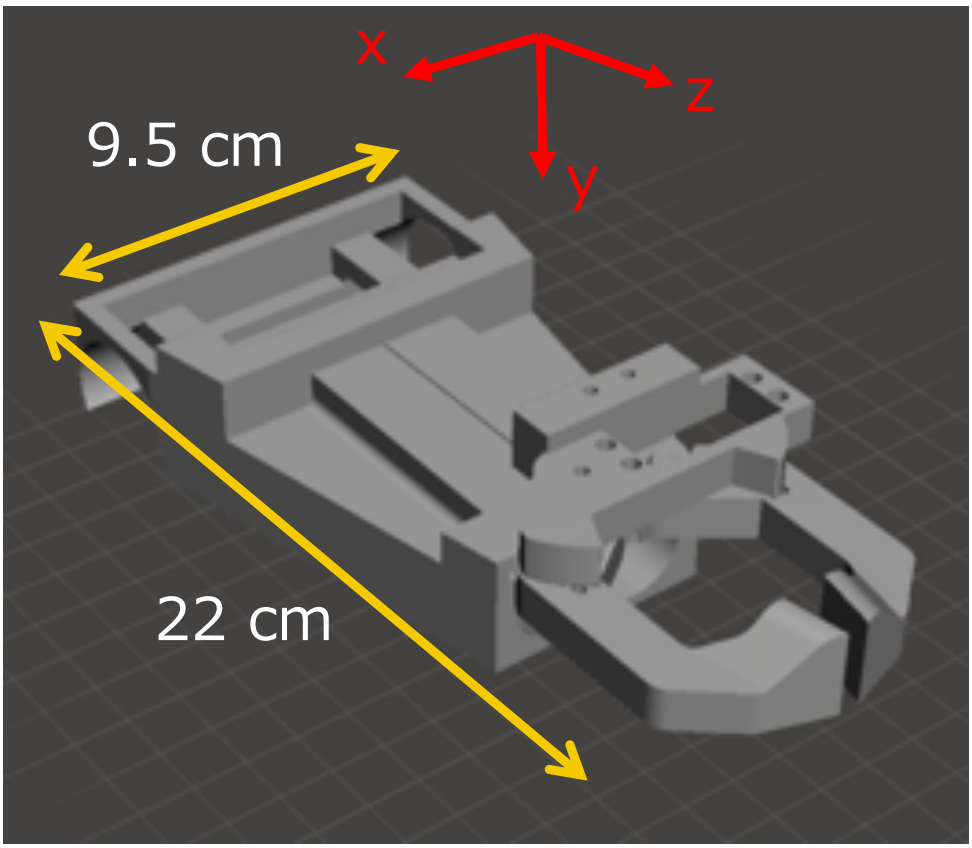


Figure 4.1: 3DCAD image of prototype No.2

4.3 制御アルゴリズム

第 3 章でロボットハンドのみの主観カメラだけでは強化学習は収束しないことを確認したため，2 号機では強化学習ではなく比例制御の考え方を利用した手法及びルールベースによって対象物のピックアップを行った．

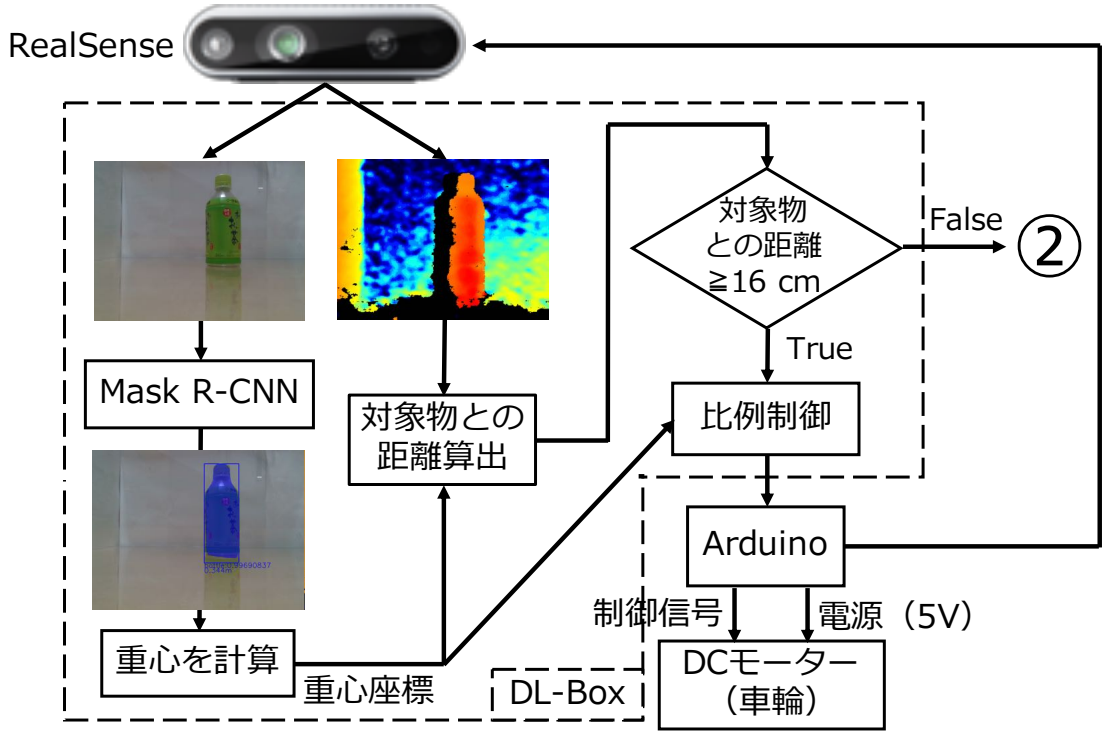


Figure 4.2: Block diagram ① of tracking objects task.

接近動作では 2 段階に分けて行った．まず対象物までの接近について述べる（Figure 4.2）．インスタンスセグメンテーションにより物体を検出し，対象物のマスクを取得し重心を計算した．その重心がロボットハンドの中心軸に来るように，左右輪のスピードをそれぞれ独立に以下の更新式によって制御しながら接近させた．

Listing 4.1: 接近アルゴリズム

```
r_motor = (1 - error_distance) / 2 * MAX_SPEED
l_motor = (1 + error_distance) / 2 * MAX_SPEED
```

ここで `MAX_SPEED` はモーターの最大スピードで今回は 100 とした．`error_distance` はロボットハンドと対象物との画像内における x 軸方向のズレを画像の横の長さの半分で規格した値であり，-1 から 1 の値をとる．そして対象物から RealSense のデプス測定限界である 20 cm 程度まで近づいたら停止させた．

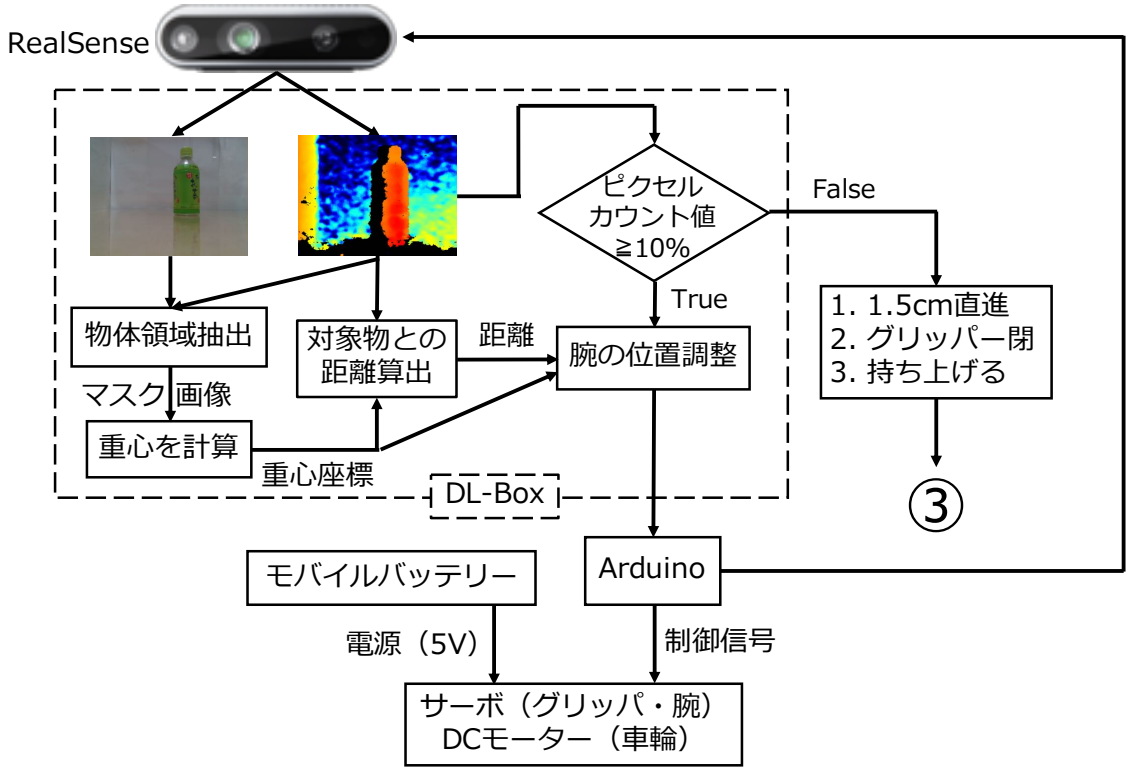


Figure 4.3: Block diagram ② of grasping objects task.

次に対象物への把持動作に向けた微調整および把持動作について述べる（Figure 4.3）．対象物の重心にハンドの把持中心が来るように横方向にラックピニオンで修正を加えた．この時，画像におけるピクセル間距離を実距離に変換し，重心とハンド中心の変位をなるべく小さくするようにサーボを動かした．そしてデプスカメラが対象物に覆われて見えなくなるまで（デプス画像の現フレームとのピクセル数比を取り，10% 以下になるまで）直進させた．そこからさらにグリッパのより内側に対象物を収めるため 4 cm 直進させ，グリッパを閉じさせた．その後腕を上げ物体を持ち上げ，この際のデプス画像を保持しておいた．そこから 1 秒ごとにデプス画像の現フレームとのピクセル数比を取り，50% 以下であったら失敗とし，5 秒落とさず維持したら把持成功とした．

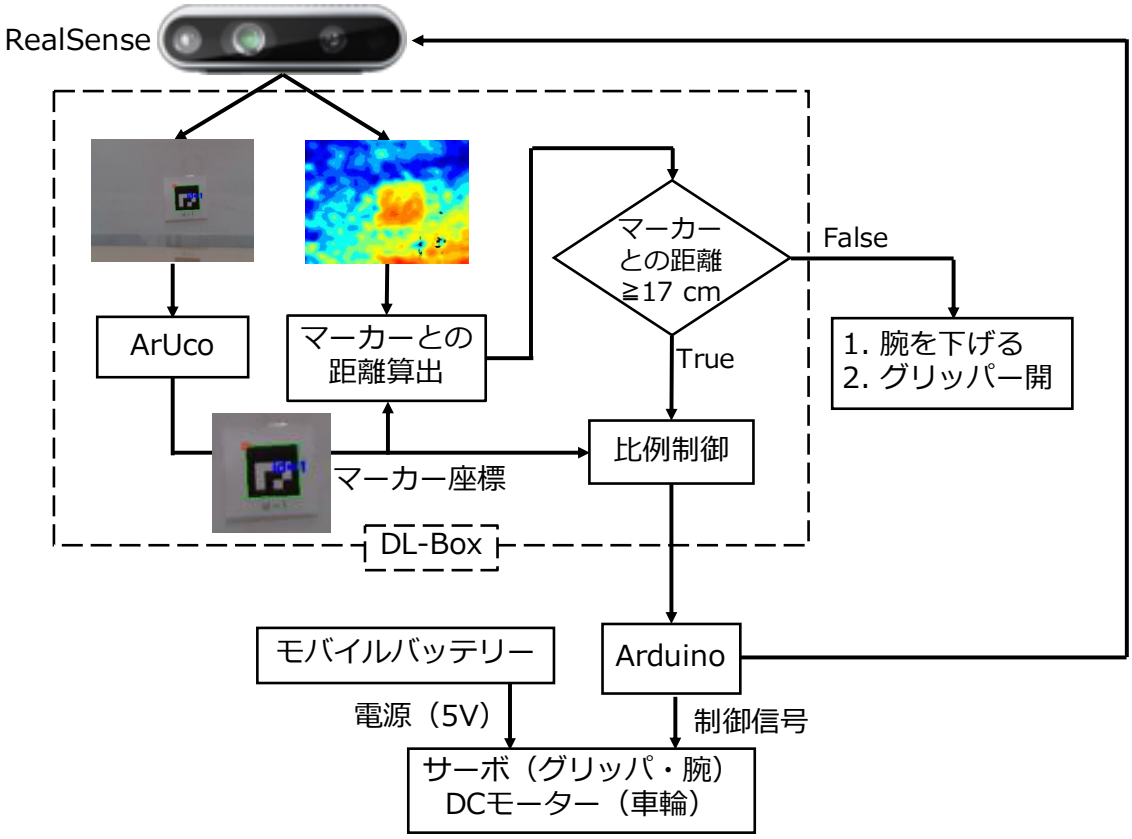


Figure 4.4: Block diagram ③ of get back home position task.

最後に，把持をした物体を運搬する動作について述べる（Figure 4.4）．把持が成功したらコード 4.1 と同様の制御でホームポジションまで戻った．ホームポジションの認識には AR マーカーを使用した．マーカーまでの距離を測定し，デプスカメラの測定限界である 17 cm より近くなったら持ち上げていた物体を下ろした．

なお，Arduino とのシリアル通信では 1 回に 2byte までしか送信できず，String 型でモーター 5 台の情報を送信すると遅延や読み飛ばしが発生した．そこで 2 号機ではモーターの取る値をデジタル化し情報を圧縮して Byte 型で送信するようにした．実際に取る値と送信 bit 信号の対応表を Table 4.1 に示す．

Table 4.1: Convert value of motor speed or servo degree to bit signal.

Actuator	Bit signal	Value @ Arduino	Value @ Python
Forward right DC motor	000	0–100(step 5)	0–20
Forward left DC motor	001	0–100(step 5)	0–20
Grip servo	010	0 / 100	0 / 1
Vertical swing servo	011	0 / 100	0 / 1
Horizontal slide servo	100	0–180 (step 10)	0–18
Terminate	101	0 / 1	0 / 1
Backward right DC motor	110	0–100(step 5)	0–20
Backward left DC motor	111	0–100(step 5)	0–20

Bit signal が駆動させるアクチュエータを 2 進数で示しており，動かす値を Value @ Python に示す 10 進数を 5 桁の 2 進数に変換してそれぞれを連結して 8 桁の 2 進数として 1byte のデータをシリアル送信した．

RealSense および Arduino の制御は G-DEP 社の DeepLearning BoxII（Ubuntu16.04）を使用し，計算リソースは NVIDIA 社の GPU（TITAN RTX）を使用した．2 号機のプログラムは筆者の GitHub[*1] で公開している．

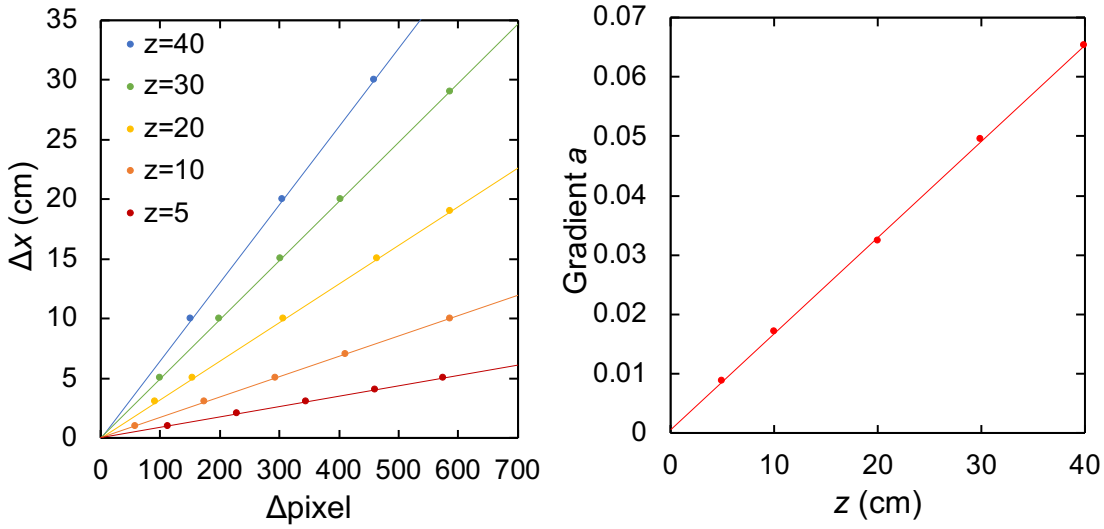
4.3.1 RealSense を使用したカメラ画像内における実距離推定

把持位置を制御する際，画像内のピクセル座標ではなく実世界における x 軸方向や y 軸方向の長さを知る必要がある（z 軸方向はデプスカメラで測距可能）．そこで，ピクセル間距離から実距離を回帰処理によって求めた．また，カメラから測る対象物までの距離にも依存するため，RealSense を用いて距離依存性についても回帰処理を行い，2 段階で求めた．

具体的には，壁に 30cm 定規を置きカメラは壁と平行になるよう設置した．壁からカメラの距離 z(cm) を変えながら定規の目盛りのピクセル座標を取得し，横軸にピクセル差分 Δpixel，縦軸にそのピクセルに対応する定規の目盛り幅 Δx(cm) をプロットした（Figure 4.5(a)）．また，各 z で線形回帰を行ったあと横軸に壁からカメラの距離 z，縦軸に傾き a をプロットし線形回帰を行い，全ての距離 z におけるピクセル間距離

[*1] https://github.com/yumion/onodera-lab/tree/master/robotHandV2

を求めた（Figure 4.5(b)）.



(a) Pixel to Distance

(b) Dependency of depth

Figure 4.5: Convert pixel-wise to distance in real.

以上から，実距離 Δx はピクセル間距離 Δpixel とカメラと対象物の距離 z を用いて

$$\Delta x \simeq (0.0016z + 0.0006) \cdot \Delta\text{pixel} \tag{4.1}$$

と推定できる.

また，ラック&ピニオンの回転から水平移動に変換するためには，サーボが 0°– 180° まで可動するため，その移動最大距離を測定しておき（4cm），各角度における 0° からの移動量をプロットし線形回帰する（Figure 4.6）ことで以下のように求められる.

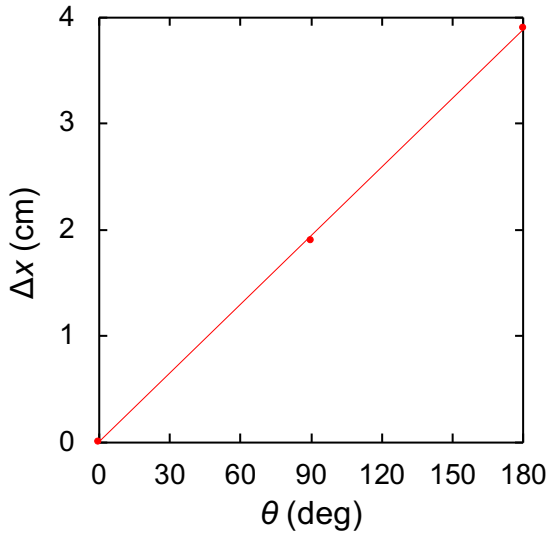


Figure 4.6: Convert servo input to distance.

$$\Delta x \simeq 0.0216\theta \tag{4.2}$$

ここで，Δx は水平移動距離 (cm)，θ はサーボの回転角度 (deg) である．Eq. (4.1) と Eq. (4.2) から

$$\theta \simeq \frac{\Delta x}{0.0216} \tag{4.3}$$

$$= \frac{0.0016z + 0.0006}{0.0216}\Delta\text{pixel} \tag{4.4}$$

$$\tag{4.5}$$

として画像からサーボを何度動かせば良いかが分かる．なお，実装上はサーボの可動範囲が 0°– 180° を考慮して

```
vertical_deg =\
(max(min(vertical_pos // 0.0216, 90), -90) + 90) // 10
```

とすることで入力の最小 0，最大を 180 とし，サーボへの入力が 0 の時に真ん中である 90° とし，それを 1/10 にデジタル化して送信する．`vertical_pos` は Δx(cm)，`vertical_deg` は θ(deg) である．

4.3.2　Mask R-CNN を用いた物体のセグメンテーションと測距

2 号機では，物体の識別のためにインスタンスセグメンテーションの手法の 1 つである Mask R-CNN を用いた．COCO2014 および COCO2017 のデータセット[*2]で学習した．学習した結果を Table 4.2 に示す．

Table 4.2: Results of Mask R-CNN validated by COCO.

	Object Detection		Segmentation	
	2014	2017	2014	2017
$AP^{IoU=0.50:0.95}_{maxDets=100}$ @Area = all	0.352	0.227	0.334	0.243
$AR^{IoU=0.50:0.95}_{maxDets=100}$ @Area = all	0.438	0.299	0.405	0.301
Inference speed (image / sec)	4.078	3.880	4.013	4.002

また，デプスカメラを用いて検出した物体の距離も同時に測定した．Mask R-CNN の出力であるマスクの重心を求め，その座標までの距離をデプスカメラで測定した．Figure 4.7 に COCO2014 で学習済みモデルを使用した物体検出およびその物体までの測距の例を示す．

Figure 4.7: Examples of Mask R-CNN inference.

このように，物体のセグメンテーションが完璧ではなくても対象物の重心はおおよそ把握できるため，IoU よりも Precision を重視し，物体検出における推論速度が速い COCO2014 の学習モデルを使用した．

[*2] http://cocodataset.org

4.4 実機作製

Table 4.3 に 2 号機作製に当たって使用した部品をまとめた．

Table 4.3: Components of prototype No.2

本体フレーム	Flashforge Filament PLA 500g レッド
カメラ	Intel RealSense D435i
マイコン	Arduino Nano
DC モーター	STL ギヤモータ 栄 42D 長軸型
サーボモータ	HS-225MG x1，HSR-5990TG x1，HS-422 x1
バッテリー	cheero Power Plus 3 mini 6700mAh
タイヤ	TAMIYA 製スパイクタイヤ
リニアガイド	MiSUMi ミニチュアリニアガイド　標準ブロック
ラック&ピニオン	MiSUMi ラックギアＬ寸固定タイプ (RGEAM0.8-300-N), MiSUMi 平歯車 m0.8 POM 青（ポリアセタール）タイプ (S80BP32B-0503)

Figure 4.8 に作製したロボットハンドの外観を示す．1 号機と同様にロボットハンドのフレームは簡便さと軽さを考慮して 3D プリンタ（FLASHFORGE 社 Adventure3）で造形した．サーボモータを 3 つ使い，それぞれグリッパの開閉（0〜50 度），腕の y 軸方向への振り上げの関節 (0〜80 度)，腕の x 軸方向へのスライド（0〜40 mm）に使用した．グリッパの先端にはゴムをつけてグリップ力を上げた．腕のスライドにはリニアガイドを用いてスムーズに腕が動くようにした．アクチュエータのバッテリーにはモバイルバッテリーを使用し，フレーム内部に組み込んだ．RealSense はロボットハンドの土台の先端にネジ止めをした．RealSense および各アクチュエータの制御は外部のデスクトップ PC と Arduino Nano を使用し，有線で接続した．2 号機の総重量は 572.27 g であった．

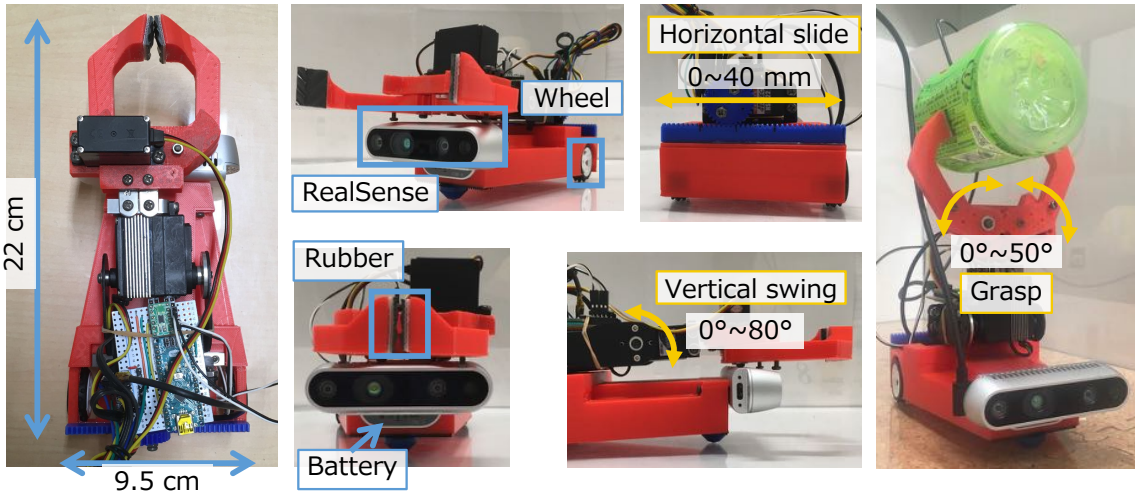
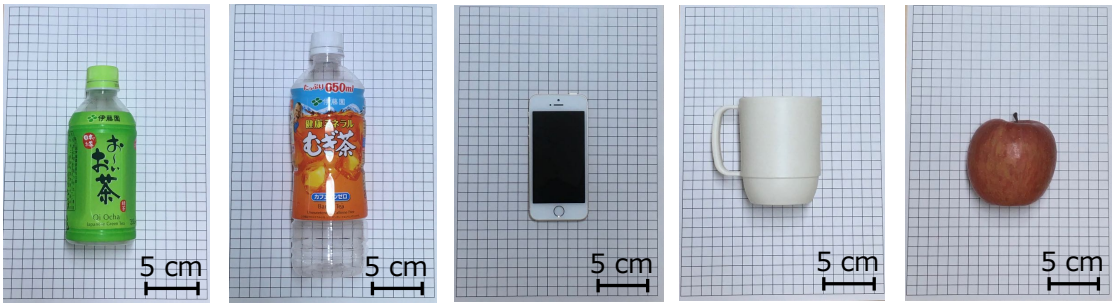


Figure 4.8: Appearance of Prototype No.2. Blue lines are expression of parts and yellow lines are motion of motors. Weight is 572.27 g.

4.5 評価方法

2 号機の改良点は対象物の識別性能とより正確な把持動作である．したがってこの 2 点に関して評価を行った．

COCO データセットの中で，机の上にあるものを RealSense カメラ（D435i）で識別・把持できるかを検証した．今回，対象とした物体は Figure 4.9 に示す 5 つの物体（4 種類）である．また，Table 4.4 に各対象物の詳細を示す．ロボットハンドの機構上，地面から高さが低い物体は把持ができないため，各対象物は正立させた状態で配置した．"cell phone"は背面にクリップを付けて直立させた．



(a) bottle(350ml) (b) bottle(350ml) (c) cell phone (d) cup (e) apple

Figure 4.9: Objects.

Table 4.4: Details of objects.

	Size	Weight	# of Training data	Notes
bottle(350 ml)	16.5 × ϕ 5.9 cm	26.28 g	8880	Empty
bottle(650 ml)	22.0 × ϕ 6.9 cm	29.94 g		Empty
cell phone	0.78 × 5.9 × 12.0 cm	127.51 g	5017	iPhone SE
cup	9.8 × ϕ 7.9 cm	82.50 g	9579	Empty
apple	ϕ 8 cm	263.32 g	1662	Raw

まず，4.3.2 節で学習を行った COCO2014 の学習済みモデルを使用して Mask R-CNN の識別精度を検証した．また各対象物において，ロボットハンドとの距離すなわち画角によって Mask R-CNN の検出精度およびそのクラス分類の精度が異なるかを検証した．

次に，各物体に対して接近成功率および把持成功率を検証した．接近成功率とは，上記実験で決めた物体検出の限界距離まで近づきかつ対象物を正面に捉えて静止したら成功，対象物の正面で止まれなかったら失敗とし，10 回試行したときの成功率とした．把持成功率とは 5 秒間持ち上げ続けたら成功，そうでなければ失敗とし，10 回試行したときの成功率とした．接近および把持のタスクは以下のように定めた．タスク 1 が接近タスクでタスク 2 が把持タスク，タスク 1+ タスク 2 が接近・把持の一貫タスクである．

- タスク 1

  離れたところに対象物を置き，上記実験で決めた物体検出の限界距離（4.6.1 よ

り 16 cm）まで接近する．

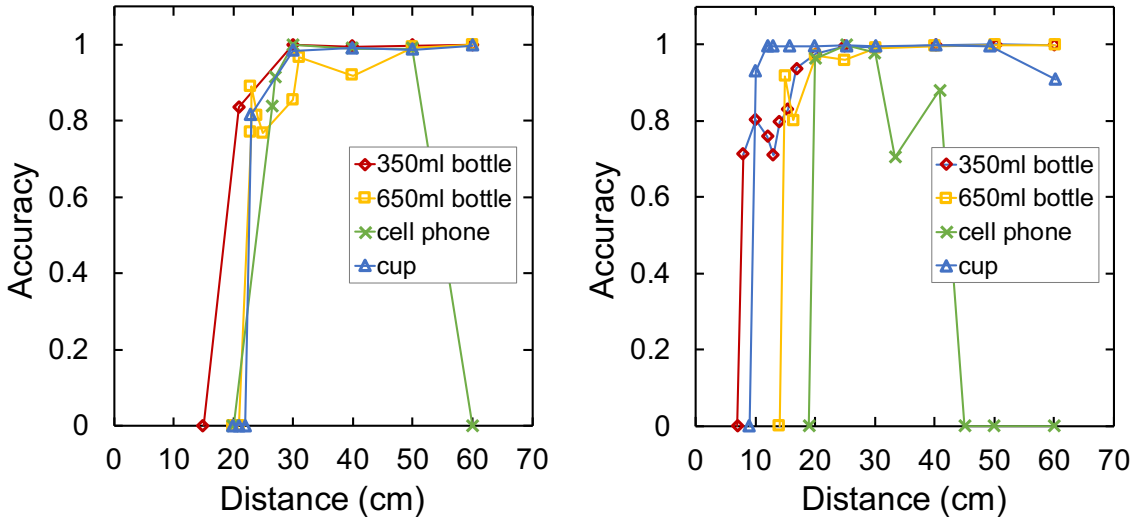
- タスク 2
  対象物の向かい（物体検出の限界距離 16 cm）の場所から把持動作を行う．
- タスク 1+ タスク 2
  離れたところに対象物を置き，物体検出の限界距離（16 cm）まで接近し，そこから把持動作を行う．

4.6 結果

4.6.1 識別性能評価

正立させた"bottle"は向きに関わらず検出できた．"cell phone"はエッジ部分だけでは検出できず，画面側と背面のみ検出できた．"cup"は取っ手が写っている角度では検出できたが，取っ手が写らない向きでは"bottle"と誤認識することがあった．"apple"はどの向きにおいても検出できなかった．

各対象物において，識別精度の対象物依存性を Figure 4.10 に示す．なお，"apple"は全ての場所・向きにおいて認識できなかったため載せていない．



(a) Input raw image.

(b) Input zero padding image.

Figure 4.10: Dependency of distance between camera and object.

カメラ画像をそのまま Mask R-CNN の入力とした結果が Figure 4.10(a) である．対象物とカメラとの距離が 23 cm を境により遠い位置での識別精度が向上し，30 cm 以上では精度がほぼ 100% であった．しかし，"cell phone"は 50 cm よりも遠いと検出できなくなった．

背の高い"bottle(650 ml)"は, 距離が近いと画像内に物体全体が収まらず検出精度が下がると予想できるが, 背の低い"cup"も同じように精度が下がった．これは COCO2014 の学習データには対象物が大きく写っている画像が少ないことが原因と考えられる．そこで，入力画像を zero padding して対象物を小さく写るようにして入力した結果が Figure 4.10(b) である．padding をすることで大幅に距離依存性が改善され，16 cm まで近づいても検出されるようになった．したがって接近タスクでは 16 cm を閾値とし，そこまで対象物に接近するタスクとした．

4.6.2 把持タスク評価

各タスクにおける成功率を Table 4.5 に示す．

Table 4.5: Success rate on grasping objects tasks.

Objects	Task1	Task2	Task1 + Task2
bottle (350 ml)	60%	90%	50%
bottle (650 ml)	60%	80%	50%
cell phone	60%	50%	40%
cup	70%	50%	30%

"apple"は認識ができなかったため，全てのタスクにおいて成功率は 0 であった．

失敗例では，掴み方が悪いため掴んでも落とした，掴んで持ち上げたがロボットハンド自体が横転した，持ち上げたがズレ落ちた，などが原因であった．

またホームポジションに戻る動作も実装した．AR マーカーを原点の壁に設置しておき，把持が成功したら旋回して AR マーカーを探索し，マーカーを捉えたらそこへ直進し，対象物を下ろすことができた．

4.7　考察

まず，認識性能について述べる．Mask R-CNN の検出精度における対象物との距離依存性では，対象物の高さに関係なく 30 cm から精度が落ち始め，20 cm では全ての物体で検出できなくなった（Figure 4.10(a)）．背の高い"bottle(650 ml)"は，距離が近いと画像内に物体全体が収まらず検出精度が下がると予想できるが，背の低い"cup"も同じように精度が下がった．これは学習データに大きく写っている画像が無いことが原因と考えられる．そこで，入力画像を zero padding して対象物を小さく写るようにして入力した結果が Figure 4.10(b) である．padding をすることで大幅に距離依存性が改善され，16 cm まで近づいても検出されるようになった．

次に，把持性能について述べる．把持成功率より接近成功率が低い原因は，2 号機の RGB カメラの位置によると考えられる．2 号機に搭載したカメラ RealSense は左端に RGB カメラ，真ん中右寄りにデプスカメラが付いている．そのため，対象物がロボットハンドから距離が離れていれば問題ないが，ロボットハンド近接時において対象物が右側に位置すると死角となり，認識不能となり接近できない．また，"cell phone"と"cup"は把持成功率が低く，"bottle"の把持成功率が高い理由としては，把持対象物の対称性が考えられる．bottle は円筒対称であるため，どの角度からアプローチしても同じ把持が可能だが，"cell phone"や"cup"は"bottle"より対称性が低いため，ルールベースだけでは正しい位置で把持できなかった．これは各物体に対して把持位置を学習するなど，より高度なアルゴリズムが必要となる．また，非対称性の複雑な形状をもつ物体の把持が可能な Jamming 転移型グリッパ [36] や多指グリッパ，吸盤グリッパなどへの変更も選択肢となる．

"apple"が認識できなかった原因は，学習データ数の偏りにあると考えられる．Table 4.4 に示したように，"apple"だけ他の対象物よりも訓練データ数が 3 分の 1 以下と少ない．したがって今回使用したモデルは"apple"の認識が弱いモデルだったと考えられる．

4.8　まとめ

物体の識別（インスタンス認識）をし，対象物の把持およびその次の動作を可能とするロボットハンドを実現するため，機構の多自由度化を行い，深層学習を用いた物体検出を用いたロボットハンドを作製した．物体検出では，データセットの特性を考え

て入力画像を padding することで，対象物とカメラの距離が近くても物体を検出できることができるようになった．日常生活によくある物体 3 種類の物体を認識し，接近・把持を行い，物体を持ち上げて運搬することを可能とし，ペットボトルの把持成功率は 9 割に達した．

2 号機の課題として，計算リソースの都合上，GPU を搭載したデスクトップ PC と有線で接続されているため，携帯性が低下した．また，ロボットハンドの重量が軽いために，ロボット本体重量以上の重い物体を持ち上げることができなかった．

第 5 章

結論

5.1　総括

本研究では，上肢機能障害者支援を目指し，強化学習・深層学習をロボット制御に適用した自律型ロボットハンドの試作を行った．携帯可能な重量と人間の手と同等のサイズを目指して小型軽量化し，Cloud は使用せず Edge のみで処理を実現した．今回開発したロボットハンドは指定した物体を識別し，接近し，把持し，使用者のもとへ運搬するタスクを達成できた．Figure 5.1 に今回試作したロボットハンドを並べて載せる．

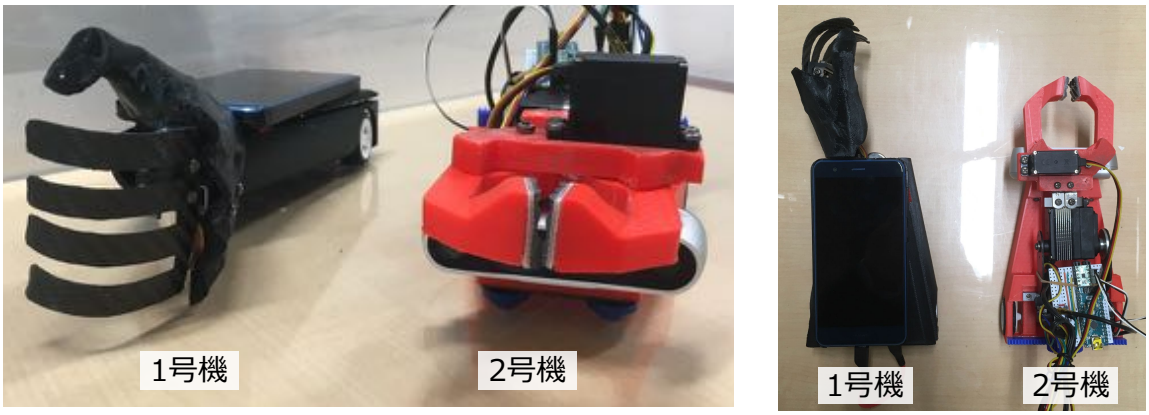


Figure 5.1: Apperarance of robot hands.

試作したロボットハンドそれぞれの特徴と課題について，以下にまとめる．

試作 1 号機

1 号機ではスマートフォンを搭載し，スマートフォンのカメラと CPU のみによる強化学習で，特定の色の物体への接近と把持に成功した．重量も一般的なパーソナルロボットよりも軽く（474 g），携帯する際使用者の負担にはならないと言える．リハビリテーション科に通う義手使用患者へヒアリングを実施した際も 1 号機は好評で，障害者支援への実用化が期待できる．

一方，計算リソースの制約から対象物の識別は不可能であり，事前に指定した物体の色のみの判断に留まった事が課題である．

試作 2 号機

2 号機では 1 号機と比べて，画像認識技術と深層学習により複数の物体がある中でも対象物を識別できるよう改善した．また，ロボットハンドの運動性能の自由度を増やし，物体を持ち上げて運搬することを可能にした．3 種類の物体の識別を可能とし，ペットボトルの把持成功率は 9 割に達した．

一方，ロボットハンドの重量が軽いために，ロボット本体重量以上の重い物体を持ち上げることができなかった．これは本体の重心位置の最適化と本体重量増加で対応可能である．また，計算リソースの都合上 GPU を搭載したデスクトップ PC と有線で接続されているため，携帯性が低下した．これは次節で述べるように，より高性能の Edge コンピュータを用いることで携帯性は大きく改善できる．

5.2　今後の展望

ロボットハンドの本体のデザイン，アクチュエータや計算リソースのハードウェア，アクチュエータの制御や画像処理を行うソフトウェアの 3 つの観点から，改善指針として以下にまとめる．

デザインの改善

今回のロボットハンドでは，地面から直立している物体のみの把持は可能だが，紙や皿のような平たい物体や細長く自立困難な物体の把持は不可能であった．そのため，腕の関節と手首の関節を増やすことで地面に対して垂直にアプローチできるように改善することで，把持できる物体形状の幅が広がる．また，今回のグリッパは一般的な形状のものであったが，様々な物体を把持可能な Jamming 転移型グリッパ [36] など，グリッパ形状についても検討する必要がある．

ハードウェア（計算リソース）の改善

2 号機では識別能力を向上させるために，外部の計算リソースを使用していたため GPU 搭載 PC と有線で接続されていたことが実用化においては課題となる．そこで次世代機では Edge デバイス（例えば NVIDIA 社の JetsonNano ボード）を使用することで，ロボットハンド自身で GPU を使用した推論が可能となる．さらに JetsonNano

を用いると直接アクチュエータの制御が可能となり，よりシンプルなロボット開発環境が実現できると考えられる．ただし，大きな消費電力が課題である．

ソフトウェアの改善

接近タスクは比例制御で十分有効であることがわかったが，今回のロボットハンドでは一般的な 2 指対立タイプのグリッパを使用したこともあり，把持タスクはルールベースでは物体の形状に対称性が無いものは把持成功率が低かった．把持タスクには Vision ベースの教師あり学習を行い最適な把持点を掴むことで改善できる．さらに識別タスクに用いた Mask R-CNN と合わせてモデルを構築できれば，End-to-End に学習ができ精度改善や推論速度向上が期待できる．

また，今回タスクの指令はコマンドラインから行ったが，実際に使用するときは外部インターフェースが必要となる．インターフェースとしては音声コマンドやスマートフォンを使ったタッチパネル入力などが有用と考えられるが，使用者の障害に応じて検討する必要がある．

参考文献

[1] I. Asimov. *I, Robot*. Robot series. Bantam Books, (1950).

[2] A. K. Pandey and R. Gelin. *IEEE Robotics and Automation Magazine*, Vol. 25, No. 3, pp. 40–48, (2018).

[3] ロボット掃除機 ルンバ | アイロボット公式サイト. https://www.irobot-jp.com/roomba/. visited on 2020/01/28.

[4] da vincii の紹介 : da vinci について | 日本ロボット外科学会 j-robo -japan robotic surgery society-. https://j-robo.or.jp/da-vinci/. visited on 2020/01/28.

[5] マイスプーン（食事支援ロボット）｜医療・介護｜防犯対策・セキュリティのセコム. https://www.secom.co.jp/personal/medical/myspoon.html. visited on 2020/01/28.

[6] Autonomous tidying-up robot system - preferred networks. https://projects.preferred.jp/tidying-up-robot/. visited on 2020/01/28.

[7] 隆明陳. *The Japanese Journal of Rehabilitation Medicine*, Vol. 49, No. 1, pp. 31–36, (2012).

[8] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase. *ROBOMECH Journal*, Vol. 6, No. 1, (2019).

[9] トヨタ自動車、生活支援ロボットの実用化に向けて研究機関等と技術開発を推進するコミュニティを発足| トヨタ自動車株式会社 公式企業サイト. https://global.toyota/jp/detail/8709536. visited on 2020/01/28.

[10] H. Toyama, H. Kawamoto, and Y. Sankai. Basic research of upper limb work support system "My cybernic robot arm" for hemiplegic persons. In *2017 IEEE International Conference on Robotics and Biomimetics, ROBIO 2017*, Vol. 2018-Janua, pp. 1–7. Institute of Electrical and Electronics Engineers Inc., (2018).

[11] H. Toyama, H. Kawamoto, and Y. Sankai. *2019 41st Annual International*

Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1645–1650, (2019).

[12] N. Dalal and B. Triggs. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1, pp. 886–893, (2005).

[13] D. G. Lowe. *Int. J. Comput. Vision*, Vol. 60, No. 2, pp. 91–110, (2004).

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346–359, (2008).

[15] C. Cortes and V. Vapnik. *Machine Learning*, Vol. 20, pp. 273–297, (1995).

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, (1998).

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Nature*, Vol. 323, No. 6088, pp. 533–536, (1986).

[18] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, (2015).

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Journal of Machine Learning Research*, Vol. 15, pp. 1929–1958, (2014).

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456. JMLR.org, (2015).

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, p. 2012, (2012).

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, (2015).

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, (2015).

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, (2016).

[25] S. Ren, K. He, R. Girshick, and J. Sun. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, pp. 1137–1149, (2017).

[26] E. Shelhamer, J. Long, and T. Darrell. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 4, pp. 640–651, (2017).

[27] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, Vol. 2017-October, pp. 2980–2988. Institute of Electrical and Electronics Engineers Inc., (2017).

[28] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8693 LNCS, pp. 740–755. Springer Verlag, (2014).

[29] Coco - common objects in context. `http://cocodataset.org`. visited on 2020/01/28.

[30] C. J. C. H. Watkins and P. Dayan. *Machine Learning*, Vol. 8, No. 3, (1992).

[31] S. Gu, E. Holly, T. Lillicrap, and S. Levine. *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3389–3396, (2017).

[32] P. Kormushev, S. Calinon, and D. G. Caldwell. *Robotics*, Vol. 2, No. 3, pp. 122–148, (2013).

[33] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. No. CoRL, pp. 1–23, (2018).

[34] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. *The International Journal of Robotics Research*, Vol. 37, No. 4-5, pp. 421–436, (2017).

[35] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, (2016–2019). visited on 2020/01/28.

[36] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger. *Proceedings of the National Academy of Sciences*, Vol. 107, No. 44, pp. 18809–18814, (2010).

謝辞

本研究は，東京大学大学院工学系研究科電気系工学専攻小野寺宏特任教授のご指導の下で行われました．この 2 年間研究を進めるにあたり，研究室内外の多くの方に多大なるご協力をいただきました．この場を借りて謝辞を述べさせていただきます．

小野寺宏 特任教授

指導教員として 2 年間ご指導いただきました．充実した研究環境を与えていただき，また幅広い先生のご紹介をしてくださり研究の異分野連携の重要性を知ることができました．毎週研究の進捗の報告でご指導いたたき，研究に行き詰まってもアイディアをたくさんいただき，研究を進めることができました．研究だけでなく今後の進路についてや医療業界の知識などを普段から教えていただき人生の選択肢を広げることができました．大変ありがとうございました．

染谷隆夫 教授

研究の進捗の発表の場を定期的に設けていただき，研究で困っていた部分や疑問に思うところについて染谷研究室の学生を含め，ディスカッションをすることができ研究を前に進めることができました．大変ありがとうございました．

横田和之 准教授

研究の方向性を心配くださりありがとうございました．また，実験 TA では電子回路の深い知識と考察や実験の注意点を丁寧に教えていただきました．大変ありがとうございました．

九州工業大学 長隆之 准教授

強化学習のノウハウについて丁寧に一から教えていただきました．東大所属でなくなってもメールでのやりとりを続けさせて頂き，とても助かりました．大変ありがとうございました．

茨城県立医療大学付属病院 四津有人 准教授
リハビリテーション科の見学を快諾してくださいました．ヒアリングをさせて頂き，研究のモチベーションを再確認できました．大変ありがとうございました．

武田伊織 博士
3D プリンターの使い方を丁寧に教えていただいたり，日々の研究で困っている時にアイディアをいただきました．学会発表や研究発表の際に発表資料の作成をたくさんご指導していただきました．大変ありがとうございました．

松崎博貴 氏
深層学習に関するプログラミングについてアドバイスいただきました．食事をよく共にして，本郷周りの美味しい飯屋を教えていただきました．またベンチャーやがんセンターの紹介等，プライベートでもたくさん道を開いてくださって感謝しています．大変ありがとうございました．

多川友作 氏
研究において活発な議論をさせていただきました．また筋トレに付き合っていただきました．大変ありがとうございました．

竹内雅樹 氏
C 言語周りのアドバイスをいただきました．大変ありがとうございました．来年の修論頑張ってください．

矢谷研究室 チェ シウク 氏
アルゴリズム面でクリティカルなアドバイスをいただきました．チェさんのおかげで問題点がいくつも解決できました．大変ありがとうございました．

鷹野玲美 学術支援専門職員
電子工作に関してたくさんご指導いただきました．また一緒に SFP に参加し YUBIBO を完成させ，テックコンテストに出場したこと，とても嬉しかったです．大変ありがとうございました．

田中麻美 技術員
普段からの雑談で研究をする元気を与えてくれました．いつも田中さんの笑顔に癒されていました．大変ありがとうございました．

両親や友人には，健康に気を使ってくれたり様々な面で支えていただきました．ありがとうございました．

研究業績

1.（口頭発表）山田敦史，松崎博貴，樽茶好彦，武田伊織，小野寺宏 「頭足類吸盤構造の 3 次元解析に基づく吸着システムの造形」第 36 回日本ロボット学会学術講演会（2018 年 9 月）

2.（口頭発表）山田敦史，松崎博貴，武田伊織，小野寺宏「強化学習を用いたパーソナルロボットハンドの開発」 電気学会センサ・マイクロマシン部門（E 部門）2019 年 バイオ・マイクロシステム研究会（2019 年 7 月）

3.（口頭発表）山田敦史，松崎博貴，武田伊織，小野寺宏 「上肢機能障がい者のための強化学習を用いた自律型ロボットハンドの開発」第 37 回日本ロボット学会学術講演会（2019 年 9 月）