卒業論文

Xtion Pro Live カメラを用いた 複数移動ロボットのビジュアルフィードバック制御の基礎実験計

F112026　三木　康平

指導教員　永田寅臣　教授

2016 年 3 月

山口東京理科大学 工学部 機械工学科

概要

目次

第 1 章

緒言

近年, 消費者嗜好の多様化により, 商品生産方式が少品種大量生産から変種変量生産へと変わってきている. それに応じて, ロボットも単一工程に特化した産業用ロボットから, 人のように状況判断を行い, 幅のある業務をこなすことができるような汎用機械としてのロボットへと, ニーズが変化してきている [1]. このニーズに応えるため, さまざまな画像処理を自動で行うカメラや, ベルトコンベア上の製品を自動で仕分けするロボットビジョンなどが販売されているが, カメラや光源の位置が限られる狭い空間内において, 対象物を画像処理により検出し移動させるシステムの実現は難しい. そこで, 研究室では狭い空間内での正確な物体の検出および移動を最終的な目標としたシステムの研究を行っている. 本研究では, 現在取り組んでいるシステムの進捗について報告する.

課題に取り組むに当たって, まず開けた空間内に平面に置かれた物体を画像認識により, 対象物の角度を依らず, 正確にピック＆プレースを行うロボットシステムを開発した. この手法では, 画像内に写るオブジェクトの角度を AI で推定し, アームで Picking を行う. 具体的には, 対象物をカメラにより撮影し, まず画像認識により対象物の重心位置を計算し, その位置までアームを移動させる. その後, 角度推定用に AlexNet を基に転移学習したネットワークがその角度を推定する. そして, アームに付属するエンドエフェクタを推定角度だけ回転させ, 対象物を Picking する.

第 2 章

運動学

ロボットアームの手先を目的の位置に移動させたいとき, 各関節角度と関節間の長さ現在の手先位置が分かっていれば, 運動学を用いて必要な姿勢を計算することが出来る. 運動学とは, ロボットアームの関節角度や関節間の長さと手先位置・姿勢の関係を数式で表したものであり, 各関節角度から手先位置・姿勢を求めることを順運動学, 手先位置・姿勢から各関節角度を求めることを逆運動学という. また, ロボットアームは各関節ごとに座標系（関節座標系）を設定することができ, この座標系と台座部に設定した基準座標系は運動学に用いることで互いに変換することができる.

2.1　順運動学

順運動学とは, ロボットアームの各関節角度の変位（回転・並進）から手先位置・姿勢を求めることであり, すなわち手先位置の関節座標系から基準座標系への変換ということができる. また, 順運動学は各関節の回転と並進の要素を持つ.

2.1.1　計算方法

ここでは関節座標の手先位置を基準座標系に変換する方法について示す. まず簡単のために図 2.1 のような系について考える.

移動後の原点 $O'(x'_0, y'_0, z'_0)$ は, 移動前の原点 $O(x_0, y_0, z_0)$ から見ると次のように表される.

$$
\begin{aligned}
x'_0 &= x_0 + x_1 \\
y'_0 &= y_0 + y_1 \\
z'_0 &= z_0 + z_1
\end{aligned}
$$

上式をベクトル $\boldsymbol{r}$ を用いて表すと, 以下のようになる.

$$
O' = O + \boldsymbol{r} \tag{2.1}
$$

関節座標系から基準座標系への変換を行うには、並進移動だけでなく回転移動についても考える必要がある. そこで図 2.2 のような簡単な系の回転移動について考える.

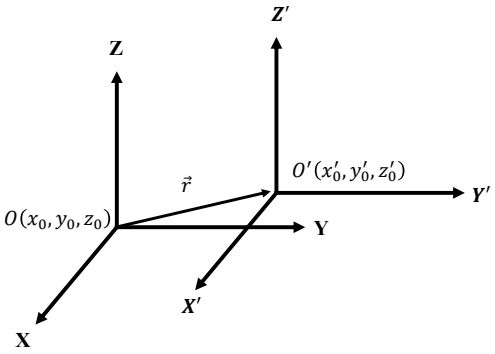


Fig.2.1. Polishing scene using a conventional industrial robot with a servo spindle.

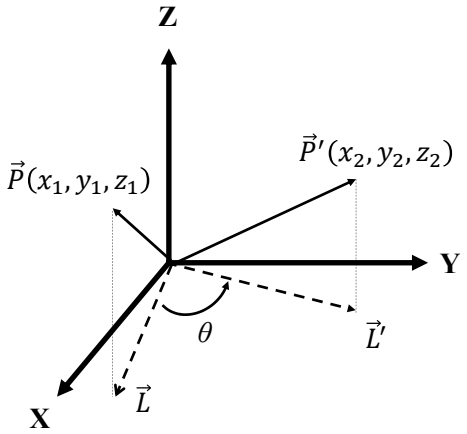


Fig.2.2. Polishing scene using a conventional industrial robot with a servo spindle.

まずベクトル $\boldsymbol{P}(x_1, y_1, z_1)$ は, 各軸方向の単位ベクトル $(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})$ を用いて次のように表される.

$$\boldsymbol{P} = x_1\boldsymbol{i} + y_1\boldsymbol{j} + z_1\boldsymbol{k} \tag{2.2}$$

次に, Z 軸を基準としてベクトル $\boldsymbol{P}$ を含む X-Y 平面を θ だけ回転した座標系を新しく X', Y', Z' 座標系とすると, その座標系におけるとすると, ベクトル $\boldsymbol{P}'(x_2, y_2, z_2)$ は単位ベクトル $\boldsymbol{i}', \boldsymbol{j}', \boldsymbol{k}'$ を用いて次のように表される.

$$\boldsymbol{P}' = x_2\boldsymbol{i}' + y_2\boldsymbol{j}' + z_2\boldsymbol{k}' \tag{2.3}$$

ここで, 単位ベクトル単位ベクトル $\boldsymbol{i}', \boldsymbol{j}', \boldsymbol{k}'$ は回転前の座標系から見ると

$$\begin{bmatrix} \boldsymbol{i}' \\ \boldsymbol{j}' \\ \boldsymbol{k}' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix} \tag{2.4}$$

と表せる. さらに式 (2.4) に式 (2.3) と式 (2.2) を代入すると

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

さらに見通しをよくするため, 単位ベクトルを省略し以下のように表す.

$$\boldsymbol{P}' = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{P} \tag{2.5}$$

すなわち Z 軸を基準としたベクトルの回転移動は, 移動前後の位置移動および相対角 θ を用いて表すことができ, これは X 軸, Y 軸においても成り立つ. この式 (2.5) 中央の行列を回転行列といい R で表すこととする. 並進移動と回転移動とを用いることにより, 図 2.3 に示すような一見複雑な座標系の変換も次のような式で表すことができる.

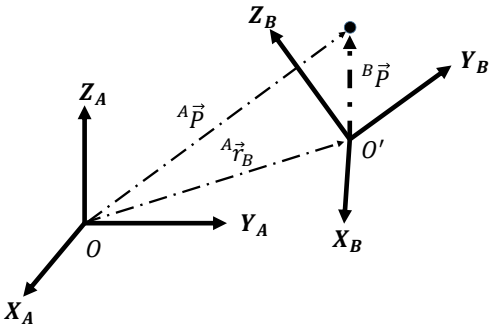


Fig.2.3. Polishing scene using a conventional industrial robot with a servo spindle.

$${}^A\boldsymbol{P} = {}^A\boldsymbol{R}_B{}^B\boldsymbol{P} + {}^A\boldsymbol{r}_B \tag{2.6}$$

ここで ${}^A\boldsymbol{R}_B$ は座標系 A から座標系 B への回転行列である. この関係を各関節ごとに用いることで, 手先位置を基準座標系からの位置として求めることができる.

2.2　逆運動学

順運動学が座標変換を行うことで解が 1 つ求まるのに対して, 逆運動学では図 2.4 の場合のように手先位置が同じ位置にある場合でも各リンクの位置と姿勢にはいくつかの解が存在することや, 解自体が存在しないこともある. このような問題があるため, 一般に順運動学より逆運動学の方が問題を解くのが難しい.

逆運動学でアームの関節角度を計算する例として, 図 2.5 のような場合を考える. まず, 角度 α と $|\boldsymbol{OP}|$ の関係は余弦定理を用いて次式で表される.

$$\cos\alpha = \frac{L_1^2 + L_2^2 - |\boldsymbol{OP}|}{2L_1L_2} \tag{2.7}$$

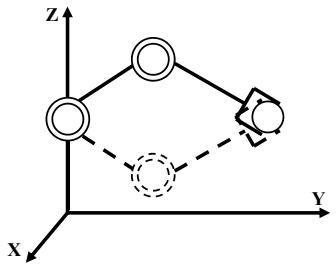


Fig.2.4. Polishing scene using a conventional industrial robot with a servo spindle.

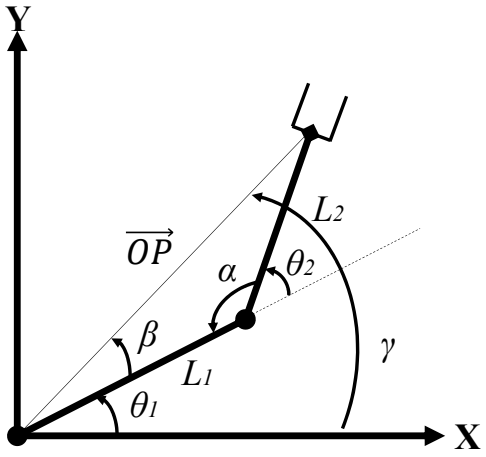


Fig.2.5. Polishing scene using a conventional industrial robot with a servo spindle.

式 (2.7) より角度 α は $\alpha = \cos^{-1}(\cos\alpha)$ と表せるので, 結果的に θ_2 は次式で表される.

$$\theta_2 = \pi - \alpha \tag{2.8}$$

同様に, 余弦定理を用いて角度 β と $|OP|$ の関係は次式で表され,

$$\cos\beta = \frac{|\boldsymbol{OP}|^2 + L_1^2 - L_2^2}{2|\boldsymbol{OP}|L_2}$$

$\beta = \cos^{-1}(\cos\beta)$ である. 最後に角度 γ は, 補助線 $|\boldsymbol{OP}|$ と手先位置の x 座標との関係から

$$\cos\gamma = \frac{x}{|\boldsymbol{OP}|}$$

であるので, $\gamma = \cos^{-1}(\cos\gamma)$ である. よって, 角度 β および角度 α を用いて θ_1 は次式で表される.

$$\theta_1 = \gamma - \beta \tag{2.9}$$

よって, 手先位置から関節角度 θ_1, $theta_2$ を求めることができた.

また別の方法として, 図 2.6 のような場合を考える. この場合, 手先位置は次式で表される.

$$\begin{aligned} x_p &= l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \\ y_p &= l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) \end{aligned} \tag{2.10}$$

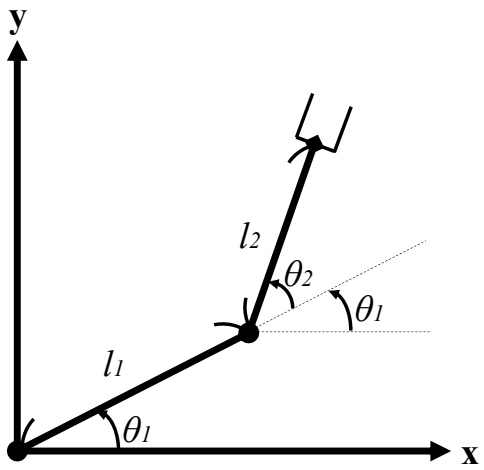


Fig.2.6. Polishing scene using a conventional industrial robot with a servo spindle.

式 (2.10) の両辺を時間で微分し, 行列で表すと

$$\frac{d\boldsymbol{P}}{dt} = \boldsymbol{J}(\boldsymbol{\theta})\frac{d\boldsymbol{\theta}}{dt} \tag{2.11}$$

ここで

$$\boldsymbol{P} = [x_p, y_p]^{\mathrm{T}}, \boldsymbol{\theta} = [\theta_1, \theta_2]^{\mathrm{T}}, \boldsymbol{J}(\theta) = \begin{bmatrix} -(l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) & -l_2 \sin(\theta_1+\theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{bmatrix}$$

である. 式 (2.12) は手先位置の速度と各関節における角速度の関係を表しており, 両辺の変換を担う行列 $\boldsymbol{J}$ を一般にヤコビ行列という. さて, 逆運動学は手先位置から各関節角度を求める問題のことであったので, 式 (2.12) を以下のように変形する.

$$\frac{d\boldsymbol{\theta}}{dt} = \boldsymbol{J}^{-1}\frac{d\boldsymbol{P}}{dt} \tag{2.12}$$

しかし, $\boldsymbol{J}^{-1}$ は必ずしも存在しない. $\boldsymbol{J}^{-1}$ が存在しない条件は, $\det\boldsymbol{J}^{-1}$ で求めることができ, その時の姿勢を特異姿勢もしくは特異点という.

第 3 章

画像認識

物体認識は, 画像認識製造業における生産ラインでのロボット視覚 (ロボットビジョン) や移動ロボットの目標追尾, 自動走行車における周辺認識など, 幅広い分野で実用化が望まれている. 文献によると, 現状の物体認識の課題は次の 2 つに大別される. 一つ目は, 対象物を正確にデータ化するための 3 次元計測に関する課題であり, 二つ目は, それをもとに対象物の位置・姿勢, 種類などを正しく認識するデータ処理に関する課題である.

3.1 画像処理

画像処理とは, アナログ画像処理方式とディジタル処理方式に分けられ, 前者はレンズ系や現像技術を用いて画像の特徴抽出や変換を行う手法であり, 後者は画像の濃度値を画素ごとに数値化し, 演算処理によって画像の特徴抽出や変換を行う手法である. 今回実験に用いた手法は, 後者に属するものである. 本節では, 画像の二値化処理について概説した後, アプリケーションに実装した処理方法について説明する.

3.1.1 色空間

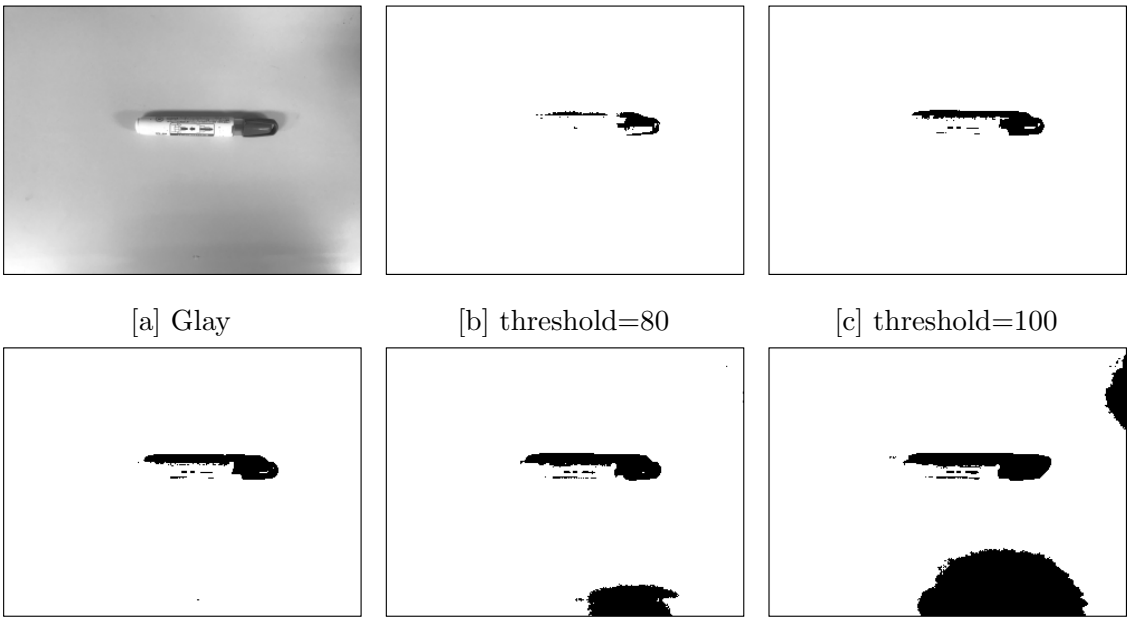
3.1.2 二値化

グレースケール画像を二値化する最も基本的な処理は, 閾値を用いる方法である. すなわち, 入力画像の画素値 $I(x, y)$, 閾値を θ とした場合,

$$I'(x,y) = \begin{cases} 255 & if(I(x,y) \geq \theta) \\ 0 & otheres \end{cases} \tag{3.1}$$

によって, 画素値を決定する.

しかし, 図のようにグレー画像全体に対して単純な閾値処理をするだけでは, 模様が潰れたり濃淡構造が部分的に失われたりする. そのため, 画素値のヒストグラムの統計量（最大値や分布など）を用いて閾値を決定する方法や, 画素ごとに閾値を変える適応的閾値処理が提案されてきた. [2, 3].



[a] Glay　[b] threshold=80　[c] threshold=100

[d] threshold=120　[e] threshold=140　[f] threshold=160

Fig.3.1. 単純二値化処理

本アプリケーションには, 単純閾値処理, Bradlry 法, 2 つの閾値を用いた閾値処理を実装した.

3.1.3　Bradley 法

Bradley 法は, 入力画像上の注目画素を中心とする, ある領域内 $(s \times s)$ の画素値の平均値を注目画素の閾値とし, その値が注目画素の画素値よりも $t\%$ 小さい場合は黒, そうでない場合は白に値を置き換える方法である. すなわち, 注目画素の閾値を式 [**?**] にて計算し,

$$S = \frac{1}{N} \sum_{i-\frac{s}{2}}^{i+\frac{s}{2}} \sum_{j-\frac{s}{2}}^{j+\frac{s}{2}} I(i,j) \tag{3.2}$$

その値を基に式 [**?**] より, 二値化を行う.

$$I'(i,j) = \begin{cases} 1 & if \ \ (I(i,j) \leq S \times \frac{100-t}{100}) \\ 0 & otheres \end{cases} \tag{3.3}$$

3.2　2 つの閾値を用いた二値化処理

この処理は, 単純二値化処理の閾値を 2 つに増やしたものであり,

3.3 Canny 法を用いたオブジェクト抽出

二値画像内において, 物体は画素値が等しいピクセルの集合として表現でき, その重心位置 $G(G_x, G_y)$ は, 画素値とピクセル座標より式 3.4 を用いて求めることができる.

$$\begin{aligned} G_x &= \frac{\sum_{p=1}^{x}\sum_{q=1}^{y} xI(p,q)}{\sum_{p=1}^{x}\sum_{q=1}^{y} I(p,q)} \\ G_y &= \frac{\sum_{p=1}^{x}\sum_{q=1}^{y} yI(p,q)}{\sum_{p=1}^{x}\sum_{q=1}^{y} I(p,q)} \end{aligned} \tag{3.4}$$

ここで, 画像の原点は左上, $I(p,q)$ はあるピクセル座標での画素値である. この式はオブジェクトが画像中に 1 つしかない場合のみ適応可能である. しかし, オブジェクトが 1 つのみの画像は稀であり, またカメラに付着した埃などにより背景に穴ができた場合も前述の式では, 重心位置がずれてしまう. そこで, 背景を除き画像内でピクセル集合（面積）が最大のものを目的のオブジェクトであると仮定し, その集合に対してのみ重心位置を求めることにする.

具体的には, まずラベリング処理により, 背景からオブジェクトを分離し, それぞれの面積を計算する. その後, 面積が最大のものに対して重心位置を計算する.

ラベリング処理とは, 2 値画像上に点在している連結成分のそれぞれに固有の名前（番号など）を付ける処理であり, これにより連結成分の個数やそれぞれの特徴（面積など）を独立して計算可能である. また, 連結成分とは, 図 3.2 に示すように中心画素とその周囲のいくつかの画素が同値で存在する領域を指し, 連結の種類には 4 連結と 8 連結がある. この 2 つの連結性

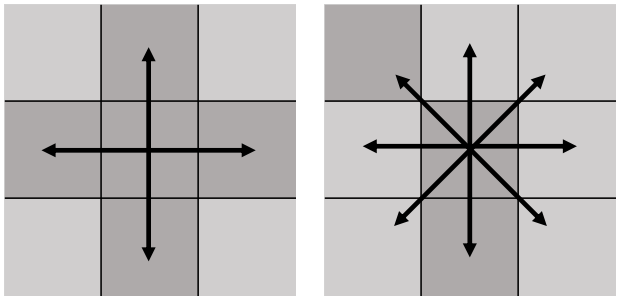


Fig.3.2. 4 connections and 8 connections

のうち, どちらか一方を背景, 他方を検出したい物体に適用することで, 背景とオブジェクトの境界付近にあるピクセルの孔の影響を抑え, それらを分離することができる. 本アプリケーションでは, デフォルトで 8 連結をオブジェクトに適用している.

分離された個々のオブジェクトの面積は, それに内包されるピクセル数を数えることで求めることができる. そして, 面積が最も大きい集合の画像全体に対する重心位置 $G'(G'_x, G'_y)$ は

次式で求められる.

$$\begin{aligned} G'_x &= \frac{\sum_{p=1}^{x}\sum_{q=1}^{y} xI'(p,q)}{S_{max}} \\ G'_y &= \frac{\sum_{p=1}^{x}\sum_{q=1}^{y} yI'(p,q)}{S_{max}} \end{aligned} \tag{3.5}$$

ここで, I' は次の条件を満たす画素値であり, S_{max} は面積の最大値である.

$$I'(i,j) = \begin{cases} 1 & if\ \ (I(i,j) \in S_{max}) \\ 0 & others \end{cases} \tag{3.6}$$

第 4 章

人工知能

4.1　人工知能

人工知能 (AI : Artifical IntelligenceI) は, 「どのようなデータを, どのようなアルゴリズムで処理すれば, 人間のような知能（認識・処理・判断など）を実現することができるのか」について研究する分野であり, 1947 年にアラン・チューリングによってその概念が提唱された.

現在の AI に関する技術 (AI 技術) や応用分野については, 文献 [4] が詳しいので, そちらを参照されたい.

4.2　機械学習

機械学習とは, 「ルールの記述が難しい問題（例えば 1 枚の画像から人の顔を見分ける問題)」に対して, 人間がルールを決めるのではなく, コンピュータ自体にルールを発見させようする手法」である. 機械学習において, コンピュータがルールを発見するまでの過程を「学習」, 発見されたルールをモデルといい, このモデルを用いて画像や音声などの判定 (認識) を行う. 学習の主なタイプとしては, 以下の 3 つが挙げられる.

- 教師あり学習
- 教師なし学習
- 強化学習

今回用いた学習方法は, 教師あり学習のみであるため, 本節では, 教師あり学習に絞って簡単に説明する. 教師あり学習とは, 学習用データすべてにあらかじめ正解となるラベルが振られており, 出力とラベルとを比較し, その誤差が小さくなるようルールを探していく方法である. 教師あり学習は, 分類問題と回帰問題に対してよく用いられる. 分類問題とは, 入力データがどのグループに属するかを予測する問題であり, 例えば, メールをスパムと非スパムに分けるものや, Web ページの内容に応じて, 「スポーツ」や「政治」などのジャンルに分類するものが挙げられる. また, 回帰問題とは, 連続した数値における予測を行う問題であり, 株価予測や人の年齢予測などに用いられる.

機械学習の分類は, 文献 [5] が詳しいので, そちらを参照されたい.

4.3　深層学習

深層学習とは, 人や動物の脳の神経回路をモデルにした, 多層のニューラルネットワークによる機械学習の手法であり, 画像認識や言語処理, ゲームなど幅広い分野で応用されている. この技術のバックグラウンドは, 図 4.1 に示すような, ニューラルネットワークと呼ばれる機械学習手法に支えられている.

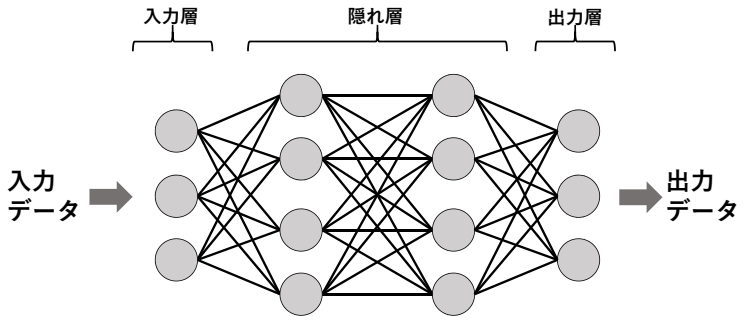
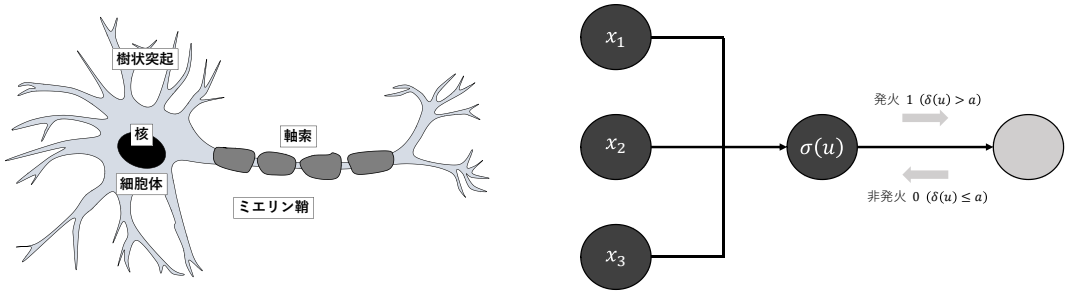


Fig.4.1. NeuralNet

4.4　ニューラルネットワーク



[a] Neuron　　[b] threshold=80

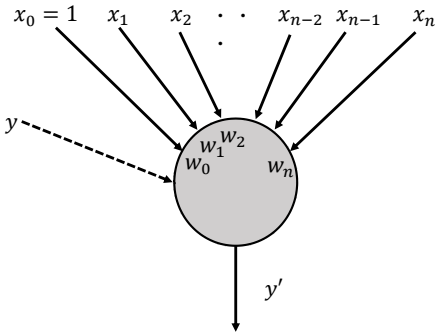
Fig.4.2. 単純二値化処理

ニューラルネットワークとは, 人間の脳内にある神経細胞（ニューロン）によって作られる神経回路網を真似て作られた数理モデルである. 人間の脳は, 図**??**(a) に示すようなニューロンが多数結合して回路網を形成している. そして, 個々のニューロンは図**??**(b) に示すように電気信号の入力側ニューロンから電気信号を受け取って蓄積し, ある閾値を超えると, 関連ニューロンに電気信号を伝達する. この閾値を超えて関連ニューロンへ電気信号が伝達されることを”発火”といい, またニューロンどうしの接続のことを”シナプス”という. シナプス強

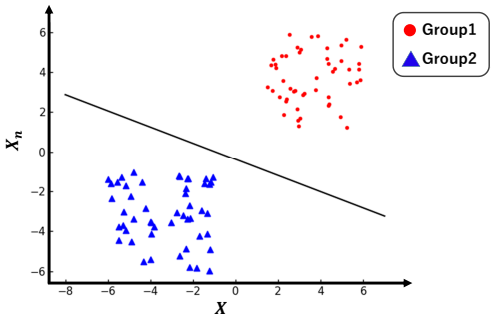
度はニューロンの組み合わせによって異なっており, 強度が高いほど伝達させる信号の強度も高くなる. このニューラルネットワークの基礎となるモデルは, 1943 年に神経生理学者の W. McCulloch と数学者の W. Pitts によって提唱された [], そして, 1957 年 F. Rosenblatt により, 学習可能なニューラルネットワークとして”パーセプトロン”が開発された.

4.4.1　パーセプトロン

パーセプトロンは, 図 4.3[a] に示すように, (n+1) 次元ベクトル $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_n)$ からなる入力を持つ. ここで x_0 は恒等的に 1 にセットされ (これはバイアス入力と呼ばれる). そして, 式 4.1 のように, 重み付け入力和が θ より多きければパーセプトロンの出力は 1 であり, 0 以下であれば出力は 0 である.



[a] perceptron



[b] 2Class Separable

Fig.4.3. パーセプトロンの概要と結果

$$y' = \begin{cases} 1 & (\sum_{i=0}^{n} w_i x_i > \theta) \\ 0 & (\sum_{i=0}^{n} w_i x_i \leq \theta) \end{cases} \tag{4.1}$$

またパーセプトロンは, 入力信号に 1 対 1 で対応する重み $\mathbf{w} = (w_0, w_1, w_2, \cdots, w_n)$ をもち, その重みは, 各信号の重要性をコントロールする要素として働く. すなわち, この重みはシナプスの結合強度を表し, 値が大きければ大きいほど, その重みに対応する信号の重要性が高くなる.

パーセプトロンの利点は, 図 4.3[b] に示すように, n 次元空間内における点を 2 つのパターンに分類できることである. ここで, それぞれの点の座標はクラス分けされる対象の属性または特徴を表す. その 2 クラスは, 単純な線形超平面 (2 次元においては直線, 3 次元空間においては平面, そして n 次元空間においては $(n-1)$ 次元平面) で互いに分離でき, この性質を持つクラスを線形分離可能という.

パーセプトロンの目標は, 入力がクラス 1 に属しているとき出力が 1, クラス 0 に属しているとき出力が 0 となるような重み $\mathbf{w}$ の集合を見つけることである. その重みは, 処理要素内に

記憶され, 式 4.2 で表されるパーセプトロン学習則に従い, 自動的に変更される.

$$\mathbf{w}' = \mathbf{w} + (y - y')\mathbf{x} \tag{4.2}$$

ここで y は入力 $\mathbf{x}$ の正しいクラス番号, y' はパーセプトロンの出力である. パーセプトロンの発明に続いて Rosenblatt は, 線形分離可能なクラスが与えられると, パーセプトロンは, そのクラスを分離する重みベクトルを有回の学習で作り出すこと (そのパターンのクラス分け作業を完璧に $(y - y') = 0$ で達成する) を証明した. この証明は重みの初期値に関係なく成り立つ.

4.5　畳み込みネットワーク

第 5 章

実機による動作実験

5.1 Dobot

TechShare 社が販売している Dobot Magician(Dobot) は, 4 自由度のロボットアームでありエンドエフェクタを取り換えることによって 3D プリンタのような積載加工, エンドミルによる切削加工ペンツールを使った印字などが行える．Dobot の質量と可搬質量はそれぞれ 3.4kg と 500g であり, 位置繰り返し精度は 0.2mm と教育ロボットとしては優れている．また Dobot 社より, C や MFC(Microsoft Foundation Class), Python など複数のプログラミング言語で API が提供されており, 個人でプログラムを組む際の敷居も低い．今回は, その中の Python API を統合開発環境である Visual Studio2019 上に実装し, Dobot 本体を動作させるプログラムの開発を行った．API とはあるコンピュータプログラム（ソフトウェア）の機能や管理するデータなどを、外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約のことである．今回はその中から Dobot 本体の姿勢を取得するための GetPose() また, Python に標準で搭載されている GUI 作成キットである Tkinter を用いて, 開発した動作プログラムを簡単に扱えるよう Dobot 制御用のユーザインタフェースの作成も行った.

5.2 学習データ

5.3 実験結果

第 6 章

今後の方針

謝辞

本研究は，山口東京理科大学大学院基礎工学研究科基礎工学専攻で行われたものである．

参考文献

[1] 井尻善久, F. Drigalski, “産業用ロボットの進化によるものづくりの近未来”, 日本ロボット学会誌, Vol. 37, No. 8, pp. 5-8, 2019.

[2] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms.”, IEEE Transactions on Systems. Man. and Cybernetics, Vol. 9, No. 1, pp. 62?66, 1979.

[3] D. Bradley, G. Roth, “Adapting Thresholding Using the Integral Image”, Journal of Graphics Tools, Vol. 12, No. 2, pp.13?21, 2007.

[4] https://www.ai-gakkai.or.jp/resource/aimap/

[5] 神嶌敏弘, 鹿島久嗣, “機械学習分野の俯瞰と展望”, 人工知能学会誌, Vol. 34, No. 6, pp. 905-915, 2019.

[6]