ディープラーニングを活用したロボット制御における安定性向上の研究

機械システム科　主任研究員　　堀　江　貴　雄

近年、ディープラーニングの活用は画像判別などにとどまらず、ロボット制御などにも応用されている。しかしながら、ニューラルネットによる直接的な制御手法では、特定条件で誤動作することが問題とされている。

そこで、本研究ではアーム搭載のメカナム移動ロボットを課題として、安定して自律制御することを目的とした。アーム制御と移動制御を組み合わせた複雑な制御を、end-to-end 学習で安定的に制御することを目標とする。そのために、今年度は実験用のアーム試作と、模倣学習用のデータセット設計、同時制御用ネットワークの設計と学習を行った。さらに実機によるエレベータ移動実験を行うことで、基本的なシステム構築手法の有効性を確認した。

１．緒言

近年ではディープラーニングによる画像判別や機械制御の事例も多く報告されるようになった。

これまでの研究において、筆者はNeural Network Console（ソニー社製）[1] を用いたディープラーニングの設計と学習、実装方法を確立した[2]。さらにカラー画像、距離画像、方位データ、目的地番号、ルート選択番号の各５時刻分を入力すると、車両速度、平行移動方向、回転速度を出力するニューラルネットワークを実現し移動ロボットに実装した[3][4]。同時に未学習環境を検出し、安全に停止させるために、カラー画像から深度画像を推定する Autoencoder を利用して、一定の閾値によって道環境を検出して強制停止させる手法も確立した。

本稿では、これに加えてエレベータを用いたフロア間移動を実現するため、エレベータボタンを操作可能なアームを試作し、車両とアームの制御パラメータを同時推定可能なネットワークの設計を行う。学習したネットワークの効果を実機による実験によって確認する。

２．アーム搭載メカナムロボット

２．１　ハンドアイシステム

カメラをハンド先端に取り付け、ビジュアルフィードバックによりロボットアームを制御するハンドアイシステムを、本研究ではエレベータボタンの操作のために用いる。試作したシステムを図１に示す。

使用するカメラには intel 社 RealSenseD435 を選定した。D435 は後述するメカナム台車に搭載の D455 よりも Depth 画像の最小測定距離が小さく、ハンド先端とエレベータボタン間の距離情報を取得しようとしたとき、アーム先端への実装に有利である。

ロボットアームの自由度はボタン操作に必要な押し込み動作ができれば十分と考え、４自由度とした。この構造ではエレベータボタンに対するロボットアーム基軸の位置によって、エレベータボタン面法線に平行な動作ができない。しかし実際にはボタン面に対して斜め方向の押し込みでも操作に問題はないと考えた。

アーム長さは工業技術センターのエレベータ操作ボタンに届くように上腕部，下腕部とも 250 mm とし、手先部は D435 測定レンジを考慮し 280 mm とした。

D435 で取得した映像を見ながら後述するゲームパッドを操作し、アーム先端座標系で Up down，RL，Push，Pitch 動作を指示可能である。なお、アーム先端座標系目標座標からの逆運動学計算は解析的に求め実装した。

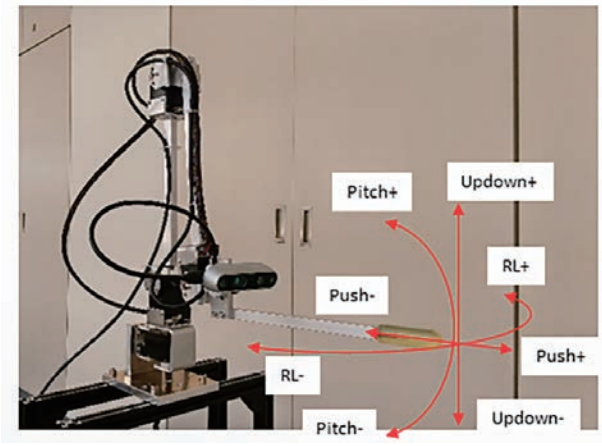

図１　ハンドアイシステム

２．２　メカナム台車

図２に試作したメカナム台車を示す。台車には４つのモータが取り付けられており各モータにメカナムホイールが取り付けられている。各モータの回転方向および速度を調整することで全方向への移動が可能と

なっている。またフロントモータはサブフレームに取り付けられており、このサブフレームはroll軸周りに回転するジョイントを介して本体フレームと結合されている。これにより路面の凹凸を吸収し、４つの車輪が設置することでスムーズな全方向移動が可能となっている。

ゲームパッドを使用しSpeed, Direction, Rotationの各パラメータを指示すると適切な各モータ回転数を決定し、台車を移動可能とするプログラムを実装した。

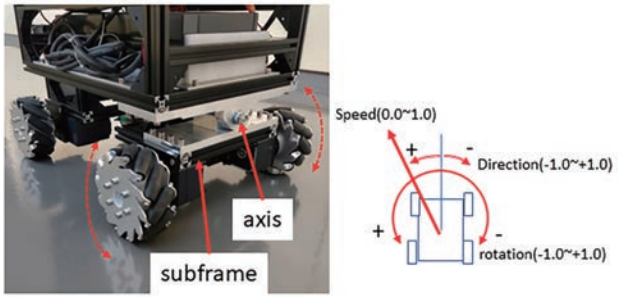

図２　メカナム駆動部

２．１のハンドアイシステムをエレベータ操作ボタンの取り付け高さにあわせて設置するために、このメカナム台車の上部にフレームを追加し、アーム基部を固定した（図３）。

車両制御用のカラー画像、深度画像撮影用にIntel製RealSenseD455センサ、磁気センサとしてWitMotion製 HWT905、モータ同期制御用のマイコンボードにArduino Uno、ニューラルネット推論および制御プログラム実行用にITX規格のPCを搭載した。

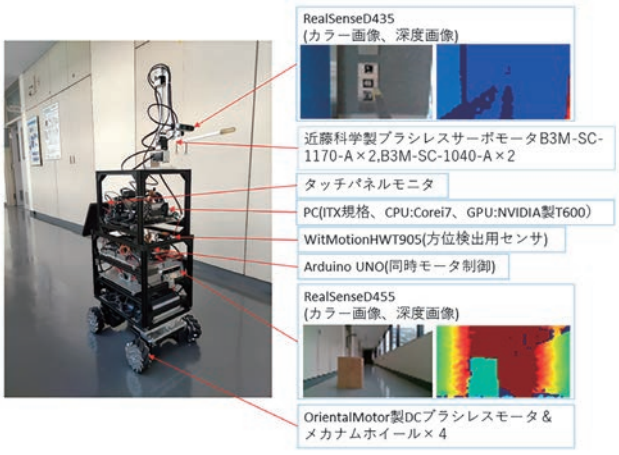

図３　アーム搭載メカナムロボット

２．３　ゲームパッド操作系

台車とアームの協調動作を手動操作で行い、データセットを生成するため、ゲームパッドを使用する。使用するゲームパッドを図４に示す。使用するゲームパッドには多数のボタンが用意されており、特に２つのスティックは操作性の観点から有用であるが、台車とアームを同時制御するには自由度が足りない。

右トリガーボタンを、「車両・アーム切り替え」とし、２つのスティックに割り当てるパラメータを切り替えることで、解決した（表１）。

図４　ゲームパッド

表１　ゲームパッド入力と
制御パラメータの割り当て関係

	右トリガーoff (車両制御モード)	右トリガーon (アーム制御モード)
左スティック 上下	前後平行移動 (directionY)と 車両速度(speedY)	アーム上下(updown)
左スティック 左右	左右平行移動 (directionX)と 車両速度(speedX)	アーム左右回転(RL)
左十字キー (上、左、右)	なし	アーム基本姿勢(pos) (正面、左、右)
右スティック 上下	なし	アーム押し引き(push)
右スティック 左右	車両回転方向と回転速度(rotation)	なし
右ショルダー	なし	アーム俯角増(angle-)
左ショルダー	なし	アーム仰角増(angle+)
Aボタン	強制データ記録	強制データ記録

３．ネットワーク設計

３．１ データセット

ロボットアームとメカナム台車の同時制御を実現す

るために、車両カラー画像、車両深度画像、アームカラー画像、アーム深度画像、方位センサデータ、ゴール番号、ルート番号から動作を推定するネットワークを学習させるために、データセットを設計した(表２)。

表２　車両、アーム同時制御用データセット

Input	Output
車両カラー画像(3×45×80)×5(5 時刻分)	車両速度 speed(1)
車両深度画像(3×45×80)×5	車両平行移動方向 direction(1)
アームカラー画像(3×45×80)×5	車両回転速度 rotation(-1～1)
アーム深度画像(3×45×80)×5	アーム push(1)
磁気センサ X(1)×5	アーム up and down(1)
磁気センサ Y(1)×5	アーム RL(1)
地点番号(9)	アーム angle(1)
ルート選択番号(1)	アーム基本姿勢 pos(4)
	車両、アーム制御選択 select(1)

なお、この５時刻分のデータセットを生成時、本ロボットシステムの制御周期は 160 ms であるが、過去時刻分データの保持は２秒おきとしている。したがって、最新時刻データとひとつ前のデータの時間間隔は０秒から２秒の間でばらついており、過去時刻４つ分の間隔は２秒固定となる。

３．２ ネットワーク設計

筆者はこれまでの研究[2-4]で、５時刻分のカラー画像、深度画像と方位センサデータ、ゴール番号、ルート番号から車両制御用パラメータを推定するニューラルネットワークを実現している。

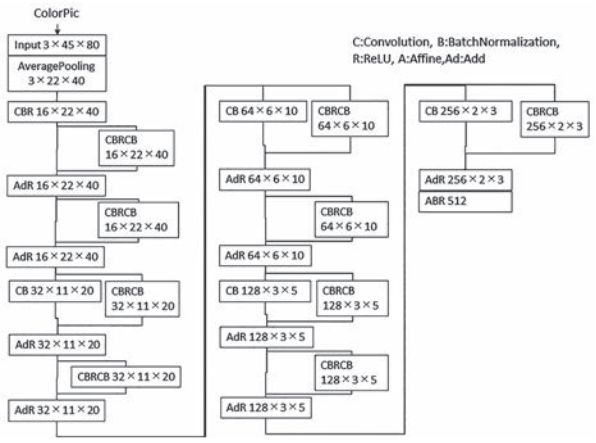

図５　PicConvUnit(カラー画像特徴抽出用)

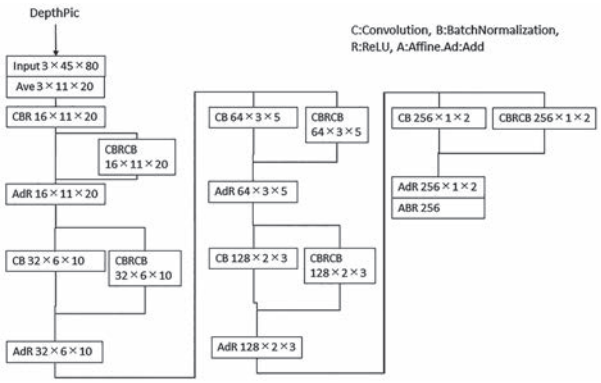

図６　DepthPicConvUnit(深度画像特徴抽出用)

このネットワークでは、カラー画像を入力すると、ResNet[5]ブロックで構成されたユニット（以後 PicConvUnit とする）で計算し、深度画像は ResNet ブロック５つを用いた、より小規模のユニット（以後 DepthPicConvUnit）で計算を行う（図５、６）。

これらユニットの計算結果を RNN[6]ブロックで集約したのち、次時刻の RNN ブロックに送る処理を時刻データごとに繰りかえす。５時刻分を処理したのちに、車両速度、車両平行移動方向、車両回転速度をそれぞれ推定する。

このネットワーク設計をベースとして、アーム制御パラメータを推定するために、車両制御推定用のネットワークと同じ設計のネットワーク構造を追加した。

アーム側ネットワークの入力値は車両パラメータと同様に、アームカラー画像、アーム深度画像、磁気センサ値、地点番号、ルート番号とした。カラー画像の処理は車両制御用ネットワークと同じ設計の PicConvUnit、深度画像には DepthPicConvUnit を用いた。

また、未学習環境における安全停止機能のために、昨年度までに開発した PTDAutoencoder[4]も追加した。これは D455 のカラー画像から深度画像を生成する Autoencoder であり、生成した推定深度画像と D455 センサで取得した深度画像の誤差を評価して未学習環境を検知し自動停止に用いる。

これら PicConvUnit, DepthPicConvUnit, PTDAutoencoder を適用して設計した車両、アームの統合学習用ネットワークを図７に示す。

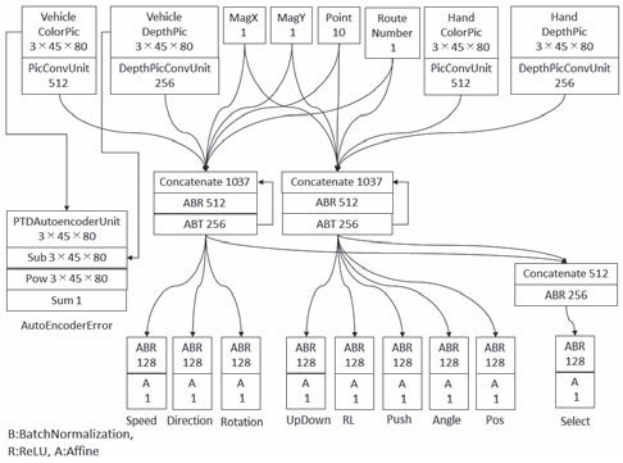

図７　統合ニューラルネットワーク

４．実験とネットワークの改良

４．１ データセット収集

工業技術センター屋内６地点に０～５の番号を割り振り、ゲームパッドの手動操作によって各地点から他の地点への移動を網羅的に行い、データセットを収集する。地点０は２階の事務室中央、１、２は２階西棟の廊下の壁側、３は２階東棟廊下の壁側、４は３階廊下の壁側、５は１階ロビーに設定した。地点０、１、２、３と地点４、５間の階層間移動ではエレベータを使用するため、ボタン操作も行う。地点番号の位置関係を図８に示す。

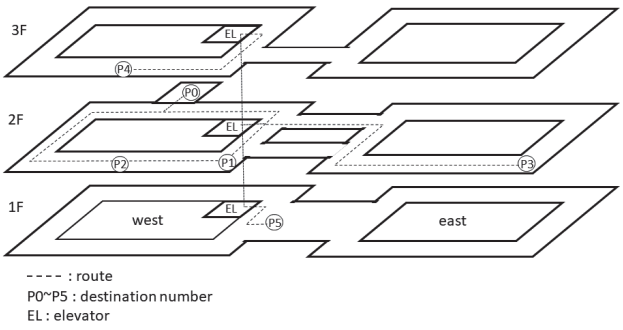

図８　地点番号と移動経路

各地点を往復する操縦データセットを合計 22 万個（約９時間 40 分に相当）収集したのち、100 epoch 学習後、Python コードに変換し、サーバーに実装した。このニューラルネットを用いて、走行実験を実施した結果、車両移動制御とアームのボタン操作制御は良好だったものの、車両、アームの制御切り替え推定と、アーム基本姿勢の推定の精度が不十分であることが判明した。精度向上のため、ネットワークの改良とデータ収集方法の見直しをおこなった。

４．２ ネットワークおよびデータ収集手法の改良

当初設計したネットワーク（図７）では、車両、アーム切り替え推定は、車両側の RNN ブロック出力とアーム側 RNN ブロック出力を concatenate で結合後、推定していた。またハンドの基本姿勢は５時刻分のアームカラー画像、アーム深度画像、磁気センサ値、目標地点番号、ルート選択番号から RNN ブロックを経由し、推定していた。

車両側 RNN、アーム側 RNN では各制御パラメータ推定用の情報抽出も担っていることから、これらの情報を共用して推定を行うのは難しいと考え、アーム基本姿勢推定と車両、アーム制御切り替え判定に特化した新たな RNN ブロックを追加した。

各時刻の車両側 PicConvUnit と DepthPicConvUnit、アーム側PicConvUnit とDepthPicConvUnit の各出力を車両側とアーム側で個別の RNN ブロックに入力し、それぞれが各時刻に出力を伝え、５時刻分を処理後に Concatenate で結合し、全結合層を介してアーム基本姿勢と車両、アーム制御切り替え判定を行う設計とした（図９）。

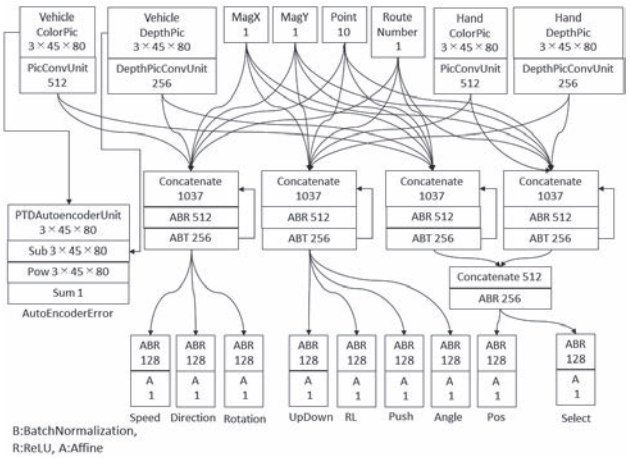

図９　改良版統合ネットワーク

データセットの見直しでは、特に誤動作の多い、エレベータ乗り込み後のアーム基本姿勢変更動作前後のデータを解析した。エレベータに乗り込んで扉が閉まった後、基本姿勢を左 90 度に変更することが正しいが、その教師データの１～２秒前に、正面姿勢を取り続ける教師データがあることが判明した。

プログラムコードを見直したところ、ゲームパッドによるデータセット記録時に、アーム切り替えボタンを押した瞬間からデータ記録するため、姿勢変更ボタンを押す前の無操作状態が不必要に記録されているこ

とが判明した。これによって、アーム基本姿勢変換前の無操作状態をニューラルネットが学習してしまっていることが分かった。

ゲームパッドの右トリガーボタン操作（アーム制御選択）かつ、左右レバーまたは左右ショルダーボタン（アーム操作量）ならび左十字キー（姿勢変更）の操作が発生したときのみデータを保存するようにプログラムを修正した。

４．３ 走行試験

これら対策後にさらにデータセットを３万個（約１時間 20 分相当）追加収集し、学習させた結果、エレベータ移動を含む全ルートの移動が可能なことを確認した。

地点０から地点４へのエレベータ操作を伴う移動例を図 10 に示す。まず、ユーザがタッチパネルから地点番号を入力後、移動開始ボタンを押すと、地点０を設定した部屋内を移動し廊下へ出る（図１０－１）。

左折後、廊下をしばらく直進、右折後にエレベータ付近で減速し、操作ボタン前で停止する（図１０－２，１０－３）。

次にアームを操作し、２階から３階に上がるために上向きのボタンを押す（図１０－４）。エレベータが到着するまで停止待機し、到着後にボタンのランプが消灯し、扉があいたのち、エレベータ内に移動する（図１０－５）。

エレベータ内で 180 度向きを変え、アームの基本ポジションを左 90 度に変更する（図１０－６）。３階に移動するために、３の数字が書かれたボタンを押す動作を行う（図１０－７）。ランプが点灯すると、ボタンから離れてランプが点灯している間（エレベータ上昇中）待機する。ランプが消えると、ハンド基本姿勢を正面に変更し、エレベータ扉が開くと移動を再開する（図１０－８）。

エレベータを出て右折、廊下を走行し突き当たりの T 字路を右に曲がる。さらに直進、地点４で減速後停止位置にアプローチを行う（図１０－９）。向きを 180 度変えて微調整を行い停止する（図１０－１０）。

以上のように、アームと車両を協調制御し、エレベータを操作して階層間移動タスクを達成することが確認できた。


１０－１


１０－２


１０－３


１０－４


１０－５


１０－６


１０－７


１０－８


１０－９


１０－１０

図１０　地点番号０から４への移動例

５．結言

本研究では、アーム搭載移動ロボットのエレベータを使った階層間移動を模倣学習するため、まずアーム搭載移動ロボットを試作し、アーム及び車両を手動操作可能とした。

次にエレベータ操作を含む階層間移動時のカメラ画像を含む各種センサデータの時系列データとその時の

操作データを記録したデータセットを作成した。

このデータセットを用いて、ResNet ブロック、RNN ブロック、Autoencoder を活用したニューラルネットを設計、学習し、実機による階層間移動実験によって、本手法が有効であることを確認した。

移動とマニピュレーションの統合制御をend-to-endで模倣学習可能なことから、本研究で用いたハードウェア構成に限らず、様々なロボット制御を現実的にend-to-endで学習可能と考える。

今後はハンドを備えた複数アームを搭載した移動ロボット制御や、少ないデータセットで移動可能なモデルの構築および、動作安定性向上の手法についても検討していきたい。

参考文献

[1] Neural Network Console, https://dl.sony.com/ja/, Accessed 2018.4.

[2] 堀江貴雄：Neural Network Console を使用したメカナム台車制御方法の開発、 ロボティクス・メカトロニクス講演会 2020, 1A1-G10, 2020.

[3] 堀江貴雄：カラー画像と距離画像を用いた模倣学習によるメカナム台車の移動制御、 第 38 回日本ロボット学会学術講演会、 3A3-06, 2020.

[4] 堀江貴雄：機械学習を用いたロボット関連製品の制御技術の開発、長崎県工業技術センター研究報告、 No.51, pp.18-22, 2022.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun ： Deep Residual Learning for Image Recognition, http://arxiv.org/abs/1512.03385, Accessed 2020.10.

[6] W. Zaremba, I. Sutskever and O. Vinyals: Recurrent neural network regularization, arXiv:1409.2329, 2015.