Raspberry pi を用いた機械学習による自動運転その 2

三糸雅昂

は じ め に

筆者は，前報にて Raspberry pi を用いた機械学習による自動運転を実現する実験装置の改良を行い，最終的に装置を完成させた。主な改良点としては，ワイヤレスコントローラーの導入に加え，それに伴うプログラムの大幅な見直しを実施し，これにより操作感の向上，タイムラグの解消，さらにはプログラムの簡素化に成功した。また，今後の課題として，この実験装置を用いた実験の実施が掲げられた。本研究の目的は，Raspberry pi という低コストかつ容易に入手可能なコンピュータを用いて，学生が自動運転技術の基礎を学ぶための教育プラットフォームを提案することである。機械学習は自動運転の中核をなす技術であり，その理解を通じて学生は現代の自動車技術の最前線に触れることができると考える。

本報では，作成した実験装置を用いて学生が実施した実験結果のまとめと，その考察，さらに教育的観点からの評価について報告する。

装 置 の 概 要

本研究において作成した実験装置について，その概要を述べる。実験装置は，ワイヤレスコントローラーからの入力によって車両の操縦を行う構造となっている。この際，車両に搭載されたカメラが走行時の画像を撮影し，同時にアクセルやステアリングの状態など，車両の走行データを記録する。このデータを基に，機械学習を活用して走行コースを学習し，その運転を模倣する仕様である。

機械学習には，畳み込みニューラルネットワーク（以下，CNN）というフレームワークを使用しており，車両に搭載されたカメラから取得した画像を基に，その瞬間のステアリング値とアクセル値を算出する。具体的には，図 1 のようにカメラで撮影された画像を規定のサイズ（55×55）に整え，各画素の RGB 値を CNN で学習することにより，その地点を走行するために必要なステアリングの角度とアクセルの量を計算するための走行モデルを作成する。この走行モデルを車両に搭載することで，実験装置は自動でコースを走行できるようになる。図 2，3 に学習モデルを作成するためのプログラムを示す。

また，Raspberry Pi 用カメラモジュールは，3 D プリンターで作成した専用のマウントステー

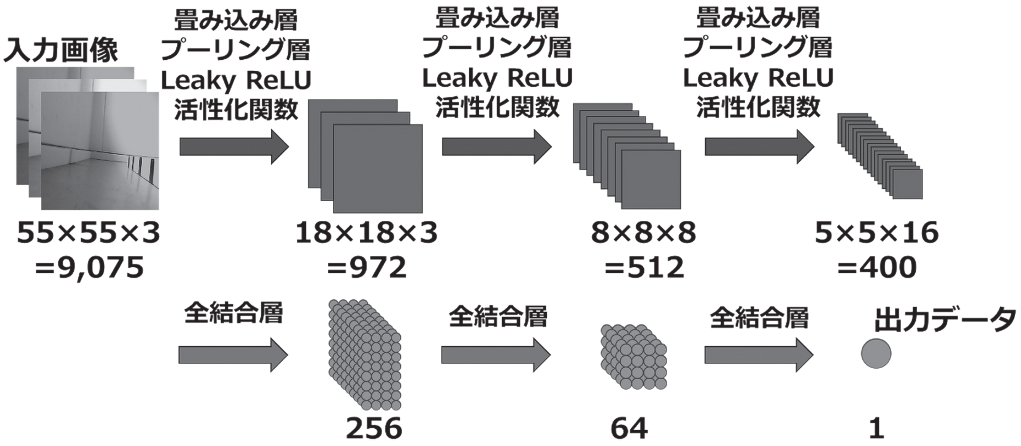

図１　畳み込みニューラルネットワークの学習イメージ

Model.py
Train.py

```python
# coding: utf-8

import numpy as np

import chainer
import chainer.functions as F
from chainer import initializers, reporter
import chainer.links as L


class CNN(chainer.Chain):
    def __init__(self):
        super(CNN, self).__init__()
        with self.init_scope():
            self.conv1=L.Convolution2D(None, 3, 3, stride=1)
            self.conv2=L.Convolution2D(None, 8, 3, stride=1)
            self.conv3=L.Convolution2D(None, 16, 3, stride=1)
            self.fc1=L.Linear(400, 256)
            self.fc2=L.Linear(256, 64)
            self.fc3=L.Linear(64, 1)

    def __call__(self, x, y):
        h = self.predict(x)
        y = y.reshape(len(y), 1)
        loss = F.mean_squared_error(h, y)
        reporter.report({'loss': loss}, self)
        return loss

    def predict(self, x):
        h = F.max_pooling_2d(F.local_response_normalization(F.leaky_relu(self.conv1(x))), 2, stride=3)
        h = F.max_pooling_2d(F.local_response_normalization(F.leaky_relu(self.conv2(h))), 2, stride=2)
        h = F.max_pooling_2d(F.local_response_normalization(F.leaky_relu(self.conv3(h))), 2, stride=1)
        h = F.leaky_relu(self.fc1(h))
        h = F.leaky_relu(self.fc2(h))
        h = self.fc3(h)
        return h


class SelfDrivingModel(chainer.Chain):
    def __init__(self):
        super(SelfDrivingModel, self).__init__()
        with self.init_scope():
            self.Accelerator = CNN()
            self.Steering = CNN()

    def __call__(self, x, y):
        y = y.transpose(1, 0)
        y1 = y[0]
        y2 = y[1]

        a1 = self.Accelerator(x, y1)
        a2 = self.Steering(x, y2)
        a = F.hstack((a1, a2))
        return a

    def predict(self, x):
        h1 = self.Accelerator.predict(x)
        h2 = self.Steering.predict(x)
        return h1, h2
```

図２　学習プロセスを設定するプログラム「model.py」

Model.py Train.py

```python
# coding: utf-8

import os
import glob
import cv2
import six
import random
import numpy as np
import chainer
import chainer.links as L
import chainer.functions as F
from chainer import cuda, Function, report, training, utils, Variable, iterators, optimizers

from Model import SelfDrivingModel

gpu = 0

if gpu >= 0:
    import cupy as cp
else:
    cp = np


def get_data():
    path = 'capture/'

    trainFile = get_file_list(path)
    trainLabel, trainData = get_label_list(trainFile)

    return trainLabel, trainData


def get_file_list(path):
    data = []
    filelist = glob.glob(path + "*.jpg")
    for filename in filelist:
        data.append(filename)
    return data


def get_label_list(filelist):
    trainlabel = []
    traindata = []
    for filepath in filelist:
        print(filepath)
        basename_without_ext = os.path.splitext(os.path.basename(filepath))[0]
        d = basename_without_ext.split('_')
        steering = np.float32(d[3])
        accelerator = np.float32(d[2])
        labels = [accelerator, steering]
        trainlabel.append(labels)

        img = cv2.imread(filepath)
        img = cv2.resize(img, (55, 55))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = np.transpose(img, (2, 0, 1))
        img = np.array(img, dtype=np.float32)
        traindata.append(img)

    return trainlabel, traindata


def main():
    print('データ作成')
    trainlabel, traindata = get_data()

    print('モデル生成')
    model = SelfDrivingModel()
    optimizer = chainer.optimizers.Adam(alpha=1e-3)
    optimizer.setup(model)

    if gpu >= 0:
        chainer.cuda.get_device_from_id(gpu)
        model.to_gpu(gpu)

    batchsize = 1000
    N = int(len(traindata) * 0.8)
    print (len(traindata))

    x_train, x_test = np.split(traindata, [N])
    y_train, y_test = np.split(trainlabel, [N])

    print('学習開始')
    for epoch in six.moves.range(1, 500 + 1):
        perm = np.random.permutation(N)
        for i in six.moves.range(0, N, batchsize):
            x_ = chainer.Variable(cp.array(x_train[perm[i:i + batchsize]]))
            t_ = chainer.Variable(cp.array(y_train[perm[i:i + batchsize]]))

            model.zerograds()
            losses = model(x_, t_)
            loss1, loss2 = losses
            loss1.backward()
            loss2.backward()
            optimizer.update()

        print('epoch={0}, AcLoss={1}, StLoss={2}'.format(epoch, loss1.data, loss2.data))

        print('epoch={0}, AcLoss={1}, StLoss={2}'.format(epoch, loss1.data, loss2.data))

    print('モデル保存')
    model.to_cpu()
    chainer.serializers.save_npz("model.npz", model)


    # 検証
    data = []
    filelist = glob.glob("capture/*.jpg")
    for filename in filelist:
        data.append(filename)
    random.shuffle(data)

    for i, file in enumerate(data):
        if i <= 10:
            img = cv2.imread(file)
            img = cv2.resize(img, (55, 55))
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = np.transpose(img, (2, 0, 1))
            img = np.array(img, dtype=np.float32)

            # 解析
            x = Variable(np.array([img], dtype=np.float32))
            ac, st = model.predict(x)

            print('st:{0}, ac:{1}, file:{2}'.format(st.data[0], ac.data[0], file))

if __name__ == '__main__':
    print('開始')
    main()
print('終了')
```

図 3　学習モデルを生成するプログラム「Train.py」

に取り付け，実験装置に搭載した。

実　　験

作成した実験装置を用いて,学生に実験を行ってもらった。実験を行った学生は本学のモータースポーツエンジニアリング学科の学生で，卒業研究として実験に協力していただいた。表１に協力していただいた学生の氏名と学生番号を記載する。

表１　実験に協力していただいた学生

学生番号	氏名
2021M1119	園山　陽輝
2021M1115	酒井　健輔
2021M1120	滝沢　隼雄
2021M1123	仲山　来成
2021M1124	西　涼輔
2021M1126	付　子豪

実験の方法は以下の５つの条件に沿って、実験装置の模擬走行を行い，それによって得られた車載画像から学習モデルを作成して，自動走行を行いその精度を評価する。

実験に使用したテストコースを図４に示す。

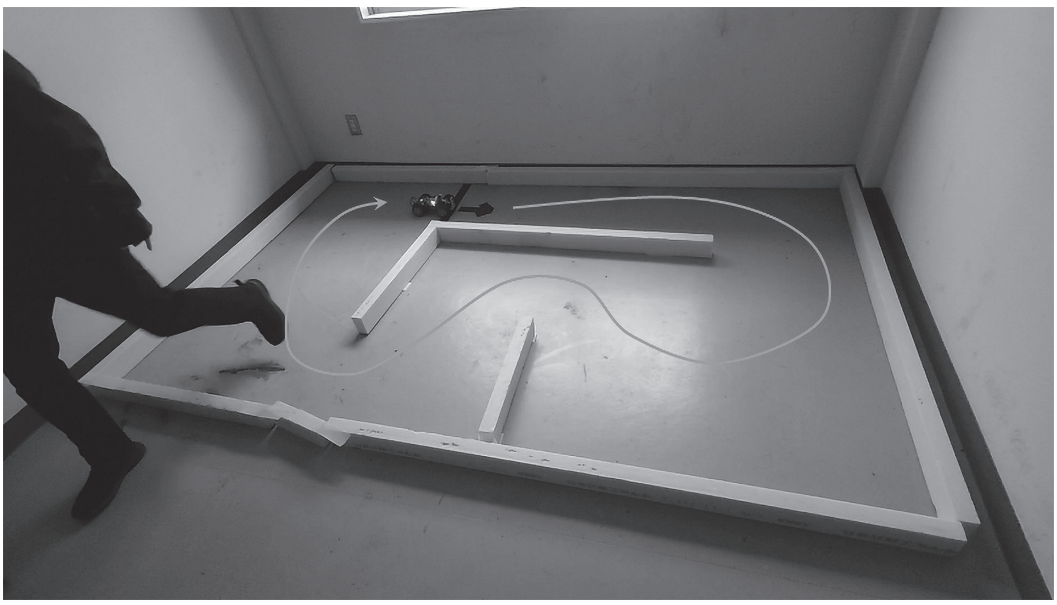

図４　実験に使用したテストコース

以下は，実験装置を用いて学生が行った 5 つの実験の結果と考察をまとめたものである。

実験 1 ：学習画像枚数の比較

■目的

学習モデルを作成する際に使用する画像の枚数が，学習モデルの性能にどのような影響を与えるかを検証する。

■方法

1 万枚と 2 万枚の画像を使用してそれぞれの学習モデルを作成し，自動運転させた。

■結果

・1 万枚の学習モデル：自動運転中にコースの半周ほどで衝突し停止。

・2 万枚の学習モデル：衝突する場面もあったが，コースを 1 周することが可能。

■考察

学習モデルの性能は使用する画像の枚数に依存することが明らかになった。より多くのデータを使用することで学習モデルの精度が向上し，自動運転の成功率が高まる。しかし，データ量が増加するにつれて学習時間も増加するため，現実的な時間内でのデータ量の上限を考慮する必要がある。

実験 2 ：カメラの比較

■目的

SunFounder PiCar-V キットに含まれる Web カメラと，新たに導入した Raspberry Pi 用カメラモジュール（Pi カメラ）の自動運転時の性能比較を行う。

■方法

2 種類のカメラ（Web カメラと Pi カメラ）を用いてそれぞれ 2 万枚の画像を取得し，同じ条件で自動運転を実行した。

■結果

どちらのカメラを使用しても自動運転の精度に顕著な違いはなく，走行結果にも大きな差は見られなかった。

■考察

Pi カメラの方が高画質で画像を記録できるが，自動運転において重要な壁や風景を捉える能力に関しては，Web カメラの画質でも十分であることが分かった。主要な特徴である壁は画質の違いに影響されず，どちらのカメラでも同様に特徴を捉えることができたため，学習には影響が出なかったと考えられる。ただし，Pi カメラのような高画質なカメラは詳細なデータを提供する可能性があり，今後の複雑な環境での自動運転には有利になる可能性がある。

実験３：カメラの取り付け位置の比較

■目的

カメラの取り付け位置（上下）が自動運転の性能にどのような影響を与えるかを検証する。

■方法

先の実験ではカメラを上側に取り付けて走行したが、今回の実験ではカメラを下側に取り付けて走行した。カメラの取り付け位置の違いを図５に示す。

■結果

カメラを下に取り付けた場合，自動運転時に走行できる距離が短くなった。

■考察

カメラの取り付け位置が下側になるほど画像に映る床や壁の割合が多くなり，特徴を捉えることが難しくなる。これにより，自動運転の性能が低下したと考えられる。今回の結果から，カメラの位置が自動運転の精度に重要な影響を与えることが確認された。

実験４：走行時のコース取りによる違い

■目的

自動運転の学習時に人間の操作によるコースの走行パターンが自動運転の精度に与える影響を検証する。

■方法

この実験では，以下の２つの走行パターンを比較した。

・レコードライン走行：最速で周回するためにコースの内側から外側へ大きく使う走行パターン（アウトインアウト）。

・端から端まで使った走行：コースの端から端までを使い，常に内側か外側かを走るパターン（インインイン，またはアウトアウトアウト）

■結果

大きな差が見られ，端から端まで使った走行パターン（後者）の方が自動運転の精度がはるかに向上した。

■考察

この結果は，コースのさまざまな場所で多様な角度からデータを取得できたことに起因すると考えられる。コースを端から端まで走ることで多様なデータが収集され，自動運転の精度向上につながった。一方，レコードライン走行では特定のラインに沿ったデータが集中し，多様な状況に対応するデータが不足したため，自動運転の精度が低下したと考えられる。コース全体を走行し，多様なデータを収集することが自動運転の精度向上に有利であることが確認された。

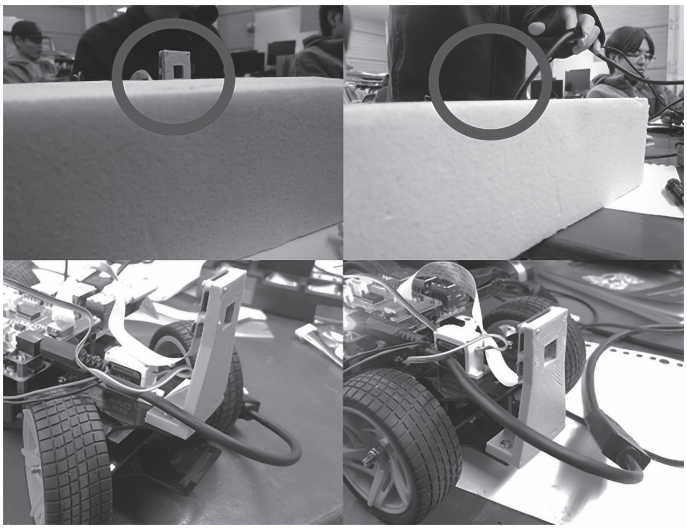

図 5　カメラの取り付け位置（左：高い，右：低い）

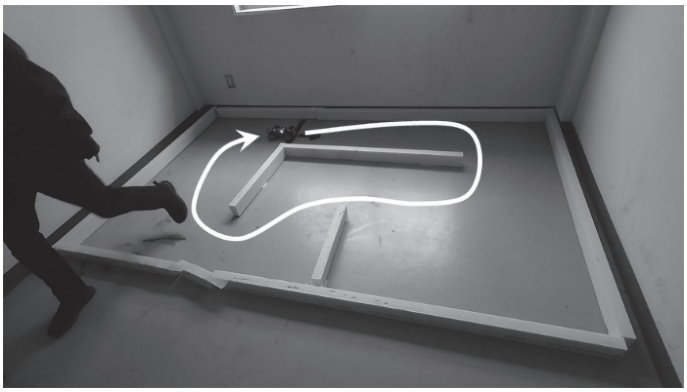

図 6　コース取り（レコードライン）

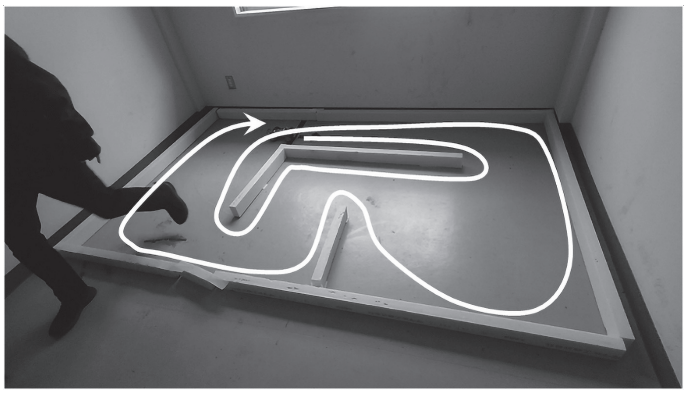

図 7　コース取り（端から端まで）

実験５：コースの柄の有無の比較

■目的

コンピュータが捉えやすい特徴をコースに追加することで自動運転の精度が向上するかどうかを検証する。

■方法

コース壁に一部に黒いガムテープで縦ラインを追加した（図８）。この縦ラインを特徴として捉えることで，自動運転の精度向上を図る。

■結果

コースに縦ラインを追加することで，自動運転の精度が向上し，車両は綺麗に１周することができた。

■考察

実験結果から，コースに目立つ特徴を追加することが自動運転の精度向上に有効であることが確認された。これは走行中の車両がコースの特徴をより正確に捉えられるようになり，結果としてステアリング値やアクセル値の判断が改善されたと考えられる。黒いガムテープで作成した縦ラインが車両の位置認識のための良い目印となり，コース上の適切なルートを維持するための判断材料となった。これにより，車両はより安定してコースを走行することが可能となった。

図８　特徴を追加したコース

教 育 的 評 価

（1）教育的価値

実験を通して，学生たちは自動運転技術の基礎とその応用について実践的な知識を得ることができた。特に以下の点で学習目標が達成された：

1．データの重要性の理解

・学習画像枚数の違いが自動運転の精度にどのように影響するのかを実感することで，大規模なデータセットの重要性を理解した。

2．ハードウェアとソフトウェアの連携の理解

・カメラの取り付け位置の変更や異なるカメラの使用による性能差を検証することで，ハードウェアの選択や配置がソフトウェアの性能にどのように影響するかを学んだ。

3．データ取得方法の影響の理解

・コースの走行パターンによる学習データの違いとその影響を比較することで，データ取得方法が学習モデルの精度に与える影響を理解した。

4．特徴抽出の重要性の理解

・コースに目立つ特徴を追加することが自動運転の精度向上に繋がることを確認することで，画像処理における特徴抽出の重要性を学んだ。

（2）実践的スキルの取得

実験を通して，学生たちは以下の実践的スキルを習得した：

1．実験デザインとデータ解析

・自動運転に関する複数の実験をデザインし，得られたデータを解析して結果を導く方法を実践した。

2．問題解決能力の向上

・カメラの位置やコースの特徴付けなど，問題に直面した際に解決策を考え実行するプロセスを通じて，問題解決能力が向上した。

（3）学生の新しい見解

実験を通して，学生たちは自動運転技術の現状と課題を深く理解することができ，新しい見解を得た：

1．データの品質と量のバランス

・大量のデータが必要である一方で，データの質も重要であることを理解し，データ収集のバランスについて考えるようになった。

2．環境の影響

・カメラの取り付け位置や周囲の特徴が自動運転の精度に影響を与えることから，実世界の環境要因が技術に与える影響について考える視点が養われた。

3．技術の限界と可能性

・現在の技術の限界を認識するとともに，工夫次第で性能が向上する可能性を見出し，自動運転技術の発展に対する興味と意欲を持つようになった。

（4）結論

今回の実験を通して，学生たちは自動運転の基本原理とその応用について深い理解を得ることができた。また，実践的なスキルを習得し，技術に対する新しい見解を得ることで，将来の技術に向けた基礎を築くことができた。この経験は，学生たちの学術的および職業的成長に大いに貢献するものとして評価される。

お　わ　り　に

本研究は，これからの学生が急速に進化する自動運転技術の革新に対応できるような教育を目指し，教育プラットフォームの提案および学生が行った実験の教育的評価を行った。本研究ではカメラを主なセンサーとして使用しているが，実際は超音波センサーやLiDARといったセンサーを複合している場合が多い。今後の課題は，実験装置に改良を加えカメラ以外のセンサーを導入し，学生が様々な実験を行うことができる環境を整えることである。本研究はまだ発展途上にあるが，学生が最新の技術に触れることができる教育課程を構築するための一助となることを期待する。

最後になりますが、本研究の遂行にあたり，ご協力を頂きました本学の学生ならびに諸先生方に厚く御礼申し上げます。

参　　考

［1］「Chainer で DonkeyCar モドキを作る 学習データ収集編（第 1 回）：株式会社アープ 開発者ブログ」
https://www.arp-corp.co.jp/blog/contents/2020/02/001.html（2023.12.25）

［2］「Raspberry Pi を使用した自動運転の研究～Chainer で DonkeyCar モドキを作る～」
中日本自動車短期大学2021年度 MSE 学科卒業論文

［3］「自動運転の研究」中日本自動車短期大学2023年度 MSE 学科卒業論文

［4］「チュートリアル ― ディープラーニング入門：Chainer チュートリアル」https://tutorials.chainer.org/ja/tutorial.html（2023.12.25）

［5］「Raspberry pi を用いた深層学習による自動運転の研究」
令和 6 年度全国自動車短期大学協会，自動車整備技術に関する研究報告誌　第53号