Raspberry Pi を用いた機械学習による自動運転その 1

三糸雅昂

は じ め に

近年，IT 技術の革新により，自動車業界においても IT 技術を活用した新技術が導入され，その技術への期待が高まっている。新技術の代表格としては自動運転が挙げられるだろう。自動運転の技術が実用レベルに達することよって，交通事故の削減や渋滞の緩和などが期待されている。2023年 4 月には自動運転車レベル 4 に相当する「運転者がいない状態での自動運転（特定自動運行）に係る許可制度」が創設され，公道での走行が認められている。こうした技術革新は100年に一度の「自動車の変革期」と言えるだろう。

これに伴い，今後の自動車整備士に要求される技術や知識は格段に高まっていくと考えられる。そこで筆者は，これからの自動車業界を担っていく学生が，自動運転の技術を少しでも多く理解できるように Raspberry Pi という小型のコンピュータを活用した自動運転の研究を行っている。

筆者はこれまでに参考［1］を参考に Raspberry Pi を用いて，決められたコースを自動で走行する自動車の模型の作成を行った。この模型は Donkey Car というラジコン向け自動運転プラットホームを模倣したもので，PiCar と呼ばれる Raspberry Pi 用のスマートロボットカーキット使用して作成し，実際に決められたコースを自動運転させることに成功した。

本報ではこれまでに行ってきた自動運転模倣装置の研究において，発生した課題を解決するためにさらなる改良を加え，より様々な実験ができるように工夫を施した。その内容について報告する。

先 行 研 究

筆者は先行研究として，市販で売られている SunFounder PiCar-V という PiCar を使用して，機械学習による自動運転を行った。（参考［2］）

市販で売られているこの PiCar にはコースを学習して，自動で運転する機能はついていないため，以下の 3 つの手順でプログラムの改変を行った。自動運転の実現の手順は以下の通りである。

1．PiCar に画像記録モードを作成する

装置を操縦しながら，走行中のカメラ映像を画像データとして記録し，同時にその瞬間のアク

セル量（アクセル値）とステアリングの角度（ステアリング値）を記録できる，画像記録モードの実装を行った。

2．デスクトップ PC に機械学習を行うためのプログラムの作成する

収集した画像データから機械学習を行い，学習モデルを作成するプログラムを作成した。機械学習は chainer というフレームワークを使用した。Raspberry Pi の性能では機械学習を行うことは困難であるため，デスクトップ PC に画像データを移して学習を行った。

3．自動運転モードの作成

作成した学習モデルを元に，同じコースを自動で走行する自動運転モードの実装を行った。実際に走行させた結果，概ね自動でコースを走行させることができた。

上記の手順により作成したこの装置を AI スマートカーと名付けた。

先行研究により，この AI スマートカーに自動運転を行わせることが可能となったが，ベースに使用した SunFounder PiCar-V に実装されていたプログラムの特性上，いくつかの課題が確認された。1 つ目はアクセル量とステアリング量が ON（100％出力）と OFF（0％出力）の 2 極化になっていること。2 つ目は Web サイトを利用した遠隔操作での操縦を行っているため，操作感にタイムラグが発生していること。3 つ目は余分な機能が実装されているため，プログラムの改変を行うのが困難であることである。本報では，上記の課題を解決するために，新しい AI スマートカー制御プログラムの設計を行った。

AI スマートカーの改良

初めにワイヤレスコントローラーの実装を行った。使用したコントローラーは PlayStation 4 というゲーム機用のコントローラーの社外品であり，ゲーム機に Bluetooth で接続するタイプのワイヤレスコントローラーである。実際に使用したコントローラーを図 1 に示す。

図 1　市販のワイヤレスコントローラー

このコントローラーを使用する理由は Raspberry Pi 自体が Bluetooth 接続を持っており，比較的導入が容易であるためである。インターネット上には，Raspberry Pi で Bluetooth 接続のワイヤレスコントローラーを使用する例が多く存在しており（参考［3］），プログラム設計時はもちろん，トラブル発生で問題が生じた場合にも，これらに関する情報を収集し易く，解決が容易であることが利点である。

ワイヤレスコントローラーを実装することで，PC を介することなく，直接 AI スマートカーを操作できるため，応答性が速く，扱いやすくなると考えられる。

1．Raspberry Pi にワイヤレスコントローラーを Bluetooth で接続する

ワイヤレスコントローラーを Raspberry Pi に接続しする方法は参考［3］［4］を参考にして行った。さらにテストプログラム「controller_test.py」を作成し，ワイヤレスコントローラーからの入力が正しく取得できるかの確認も行った。その具体的な方法を以下に示す。

・Raspberry Pi に「pyPS 4 Controller」というライブラリをインストールする。
・ターミナル上で「bluetoothctl」を開く。
・「power on」を実行する。
・コントローラーをペアリングモードにする。

今回使用しているコントローラーは PlayStation 4 用のコントローラーであるため，具体的には SHARE ボタンと HOME ボタンを同時に長押しする。

・「scan on」を実行する。
・検出された「Wireless Controller」のアドレスをコピーし，ペアリング「pair（アドレス）」を実行する。
・トラスト「trust（アドレス）」を実行する。
・Raspberry Pi を再起動する。

以降，Raspberry Pi 起動後にコントローラーの HOME ボタンを押すことで，Bluetooth 接続が可能となる。

・テストプログラム「controller_test.py」を作成し，確認を行う。

確認に使用したプログラムを図 3，実行結果を図 4 に示す。正常にコントローラーが Raspberry Pi と Bluetooth 接続され，「×」ボタンが入力されると「Hello world」と表示され，ボタンが離されると「Goodbye world」表示されていることがわかる。このプログラムを流用し，それぞれのボタンに対する処理を関数の中身に書き加えることで，Raspberry Pi に任意の制御を行わせることが可能である。

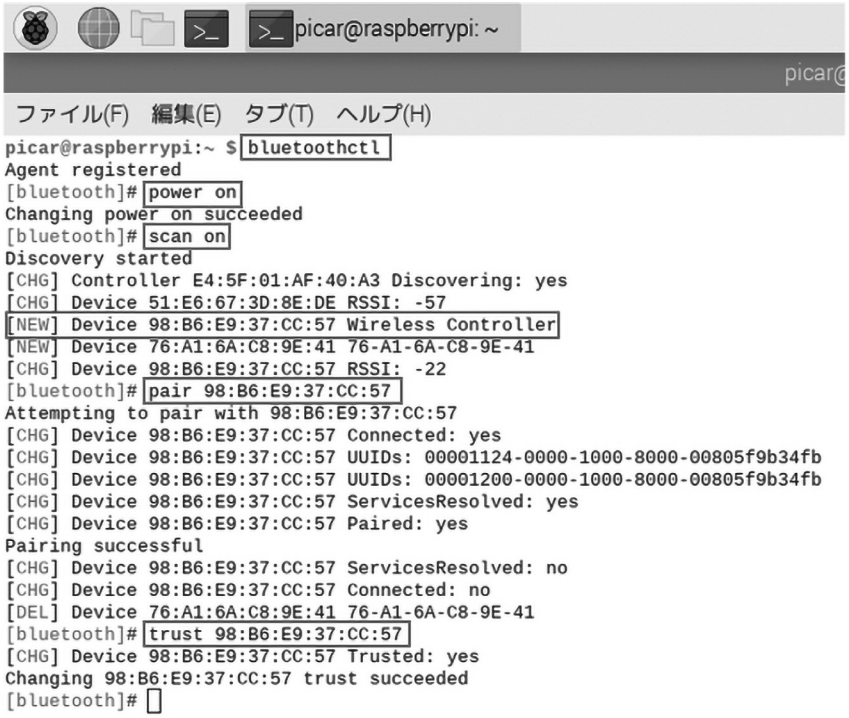

図２　Bluetooth 接続設定時ターミナルの様子

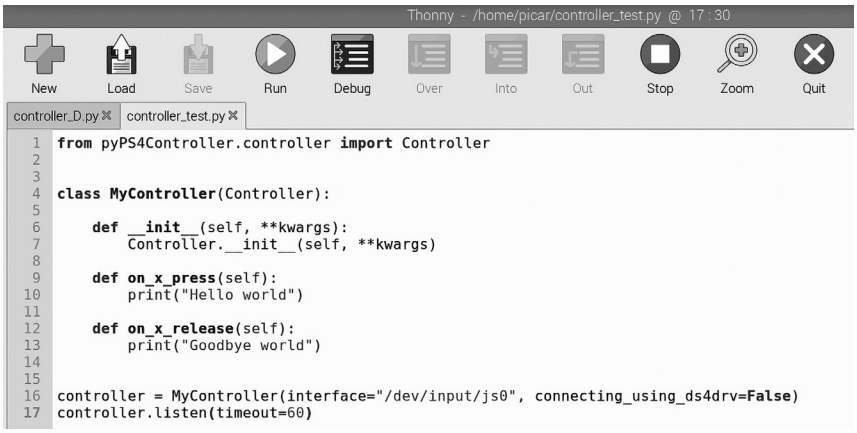

図３　テストプログラム「controller_test.py」

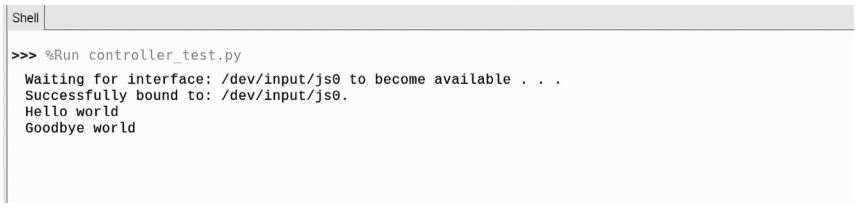

図４　テストプログラム「controller_test.py」実行結果

2．車両制御プログラム「car_control.py」を作成する

「car_control.py」はコントローラー制御プログラム「controller.py」からの指示を受け，実際にモータを制御している PiCar 既存のプログラムを利用して，AI スマートカーを制御するプログラムである。

主な機能は以下の通りである。

・走行中のカメラ画像を取得し，記録を行う「capture_task」

車両を走行させながら，カメラの画像を取得するためには，同時に 2 つ以上の処理を行わなければならないため，「multiprocessing」モジュールを使用し，並列処理を行う。処理の開始と停止はそれぞれ「cap_start」「cap_stop」で行っている。「capture_task」には実際の処理内容を記述してあり，openCV を使った画像記録モードの処理が行われる。

また，取集するデータを元に解析を行うため，記録中に車両が停止してしまうと自動運転を行う際にそのデータが反映されて車両が停止してしまう。このことを防ぐために車両の速度が 0 の時は記録を停止するように細工を施した。

・学習データを元に自動運転を行う「auto_task」

「capture_task」と同様に，自動運転の処理も並列処理で行う。処理の開始と停止は「auto_start」「auto_stop」で行い，実際の自動運転の処理は「auto_task」と「self_driving」で行う。処理が開始されると「auto_task」により，学習データを元にカメラ画像の解析を行い，それに応じてステアリング値「st」とアクセル値「ac」が算出される。その 2 つのデータを引数として「self_driving（ac, st)」関数を呼び出すことにより，自動運転が行われる。

・AI スマートカーの前輪の制御を行う「turn, turn_off」

前輪の制御は PyCar の制御プログラム中の「front_wheels」というプログラムを利用し，関数「turn（x)」が呼び出されると指定された角度 x をメモリに記録し，前輪がその角度になるようにサーボモータを制御する。切れ角は中心から左右に90度程度で，中心を90度として，最も左に切れた位置を45度，最も右に切れた位置を135度としている。角度の指定は関数の引数として x に45から135の数値を指定することで，任意の角度に操作できるように設計を行った。また，関数「turn_off」は前輪の切れ角を中心の90度に戻すように設計した。

・AI スマートカーの後輪の制御を行う「ac_on, ac_off, ac_back」

後輪の制御は PyCar の制御プログラム中の「back_wheels」というプログラムを利用し，関数「ac_on (x)」が呼び出されると指定された出力値 x をメモリに記録し，その値に合わせて後輪モータの回転速度を制御する。最大出力を100として x に 0 から100の数値を指定することで，任意

の速度で走行させることができるように設計を行った。また,「ac_on(x)」は車両を前進させ,「ac_back（x)」は車両を後退させる。「ac_off」は速度を０にしてストップさせることができるように設計した。プログラムを図５・６に示す。

car_control.py

```python
from picar import back_wheels, front_wheels
import picar
from multiprocessing import Value, Process
import datetime
import cv2
from time import sleep
import numpy as np
import chainer
from chainer import Variable
from Model import SelfDrivingModel

class CarControl():
    SPEED = 60

    AC = 0
    ST = 90
    BW_STATUS = 0

    SLEEP = 0.1

    def __init__(self):
        picar.setup()
        db_file = "/home/pi/SunFounder_PiCar-V/remote_control/remote_control/driver/config"
        self.fw = front_wheels.Front_Wheels(debug=False, db=db_file)
        self.bw = back_wheels.Back_Wheels(debug=False, db=db_file)

        self.cap_process = None
        self.task_flg_value = Value("i", 0)
        self.ac_value = Value("i", self.AC)
        self.st_value = Value("i", self.ST)
        self.bw_status = Value("i", self.BW_STATUS)

        self.auto_process = None
        self.auto_flg_value = Value("i", 0)

        self.bw.ready()
        self.fw.ready()

    def capture_task(self, task_flg, ac, st):
        if task_flg.value == 1:
            return

        task_flg.value = 1
        cap = cv2.VideoCapture(0)

        while(task_flg.value):
            _ac = ac.value
            _st = st.value
            _now = datetime.datetime.now().strftime("%Y%m%d%H%M%S%f")

            filename = "./capture/cap_{0}_{1}_{2}.jpg".format(_now, _ac, _st)

            if self.bw_status.value == 1:
                print(filename)
                ret, frame = cap.read()
                if ret:
                    cv2.imwrite(filename, frame)

            sleep(self.SLEEP)

        cap.release()
        task_flg.value = 0
        print("capture process stop")

    def auto_task(self, auto_flg):
        if auto_flg.value == 1:
            return

        auto_flg.value = 1

        modelFileName = "./model/model.npz"
        model = SelfDrivingModel()
        chainer.serializers.load_npz(modelFileName, model)
        cap = cv2.VideoCapture(0)

        while(auto_flg.value):
            ret, frame = cap.read()

            if ret:
                img = cv2.resize(frame, (55, 55))
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                img = np.transpose(img, (2, 0, 1))
                img = np.array(img, dtype=np.float32)

```

図５　車両制御プログラム「car_control.py」その１

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

                print("st:{0}, ac:{1}".format(st.data[0], ac.data[0]))
                self.self_driving(np.round(int(ac.data[0])), np.round(int(st.data[0])))


        cap.release()
        auto_flg.value = 0
        print("auto process stop")
        self.self_driving(0, 90)

    def self_driving(self, ac, st):
        if ac < 0:
            ac = 0
        elif ac > 100:
            ac = 100
        self.turn(st)
        self.ac_on(ac)

    # ========== front wheels ==========
    def turn(self, st):
        self.st_value.value = st
        self.fw.turn(st)

    def turn_off(self):
        self.st_value.value = 90
        self.fw.turn_straight()

    # ==========  back wheels ==========
    def ac_on(self, ac):
        self.ac_value.value = ac
        self.bw_status.value = 1
        self.bw.speed = ac
        self.bw.forward()

    def ac_off(self):
        self.ac_value.value = 0
        self.bw_status.value = 0
        self.bw.speed = 0
        self.bw.stop()

    def ac_back(self, ac):
        self.ac_value.value = ac
        self.bw_status.value = -1
        self.bw.speed = ac
        self.bw.backward()

    # ==========   capture    ==========
    def cap_start(self):
        print("capture start")
        self.cap_process = Process(target=self.capture_task, args=[self.task_flg_value, self.ac_value, self.st_value])
        print("capture process start")
        self.cap_process.start()

    def cap_stop(self):
        print("capture stop")
        if self.task_flg_value.value:
            self.task_flg_value.value = False
            print("cap process stop")

    # ==========     auto     ===========
    def auto_start(self):
        print("auto start")
        self.auto_process = Process(target=self.auto_task, args=[self.auto_flg_value])
        print("auto process start")
        self.auto_process.start()

    def auto_stop(self):
        print("auto stop")
        if self.auto_flg_value.value:
            self.auto_flg_value.value = False
            print("auto process stop")

if __name__ == "__main__":
    car = CarControl()
    car.ac_off()
    car.ac_on(10)
    #car.ac_back(20)
    car.turn(100)
    car.turn_off()
    car.cap_start()
    sleep(3)
    car.cap_stop()

```

図 6　車両制御プログラム「car_control.py」その 2

3．コントローラー制御プログラム（controller.py）を作成する

コントローラー制御プログラムは車両制御プログラム「car_contorl.py」を利用してコントローラーからの入力に対して車両を動かすインターフェースである。「controller_test.py」を書き換えてコントローラーの各ボタンに処理を書き加える。コントローラー操作は以下のように設定した。

図７　コントローラーの操作設定

Ｌ３スティック：ステアリング操作（左右のみ・強弱あり）
Ｒ２ボタン：前進（強弱あり）
Ｌ２ボタン：後退（強弱あり）
×ボタン：画像記録モード ON
○ボタン：画像記録モード OFF
△ボタン：自動運転モード ON
□ボタン：自動運転モード OFF

それぞれのボタンに対応する処理をプログラムに書き加え，コントローラー制御プログラム「controller.py」を作成した。プログラムを図８に示す。それぞれの処理は以下のように設定した。

controller.py

```python
from pyPS4Controller.controller import Controller
from car_control import CarControl


class MyController(Controller):

    def __init__(self, **kwargs):
        Controller.__init__(self, **kwargs)
        self.car = CarControl()

    #============ steering ============
    def on_L3_left(self, value):
        angle = int(value/32767*45+90)
        self.car.turn(angle)

    def on_L3_right(self, value):
        angle = int(value/32767*45+90)
        self.car.turn(angle)

    def on_L3_x_at_rest(self):
        self.car.turn_off()

    #========== accelerator ==========
    def on_R2_press(self, value):
        ac = int(value/32767*50+50)
        self.car.ac_on(ac)

    def on_R2_release(self):
        self.car.ac_off()

    def on_L2_press(self, value):
        ac = int(value/32767*50+50)
        self.car.ac_back(ac)

    def on_L2_release(self):
        self.car.ac_off()

    #============ capture ============
    def on_x_press(self):
        self.car.cap_start()

    def on_circle_press(self):
        self.car.cap_stop()

    #============== auto ==============
    def on_triangle_press(self):
        self.car.auto_start()

    def on_square_press(self):
        self.car.auto_stop()

    #============ release ============
    def on_x_release(self):
        return 0
    def on_circle_release(self):
        return 0
    def on_triangle_release(self):
        return 0
    def on_square_release(self):
        return 0

    def on_L3_up(self, value):
        return 0
    def on_L3_down(self, value):
        return 0
    def on_L3_y_at_rest(self):
        return 0

    def on_R3_up(self, value):
        return 0
    def on_R3_down(self, value):
        return 0
    def on_R3_left(self, value):
        return 0
    def on_R3_right(self, value):
        return 0
    def on_R3_y_at_rest(self):
        return 0
    def on_R3_x_at_rest(self):
        return 0


controller = MyController(interface="/dev/input/js0", connecting_using_ds4drv=False)
controller.listen(timeout=60)
```

図 8　コントローラー制御プログラム「controller.py」

・ステアリング操作

Ｌ３スティックの左右の入力に対し，その入力の大きさに応じて前輪の切れ角が変化するように設計を行った。今回使用しているライブラリ「pyPS４Controller」はスティック入力の大きさにより引数「value」の値が変化する。検証した結果，最大値はスティックを最大まで倒したとき，値が32767になることが分かった。検証結果を図９に示す。

ステアリングの切れ角は中心を90度として左右に45度であるため，スティックの入力値「value」をステアリングの角度「angle」に変換するための式は以下のようになる。

$$\text{angle} = \text{value} \times \frac{45}{32767} + 90$$

「angle」を引数として関数「turn_on（angle）」を実行させることで，コントローラーのＬ３スティック入力により前輪の操舵を行うことができる。またスティック入力が無くなった時，最後の値がvalue に残ってしまうため，ステアリングが中心に戻らない現象が起こる。このことを防ぐため，入力が無くなったら関数「turn_off（）」を実行するようにした。

・アクセル操作

Ｒ２ボタンとＬ２ボタンの入力に対し，その入力の大きさに応じてそれぞれ前進，後退を行うように設計を行った。ステアリングの時と同様にＲ２ボタン・Ｌ２ボタンは入力の大きさによって引数「value」の値が変化する。こちらも検証した結果，最大値は32767であることがわかる。しかし「on_Ｒ２_release」の直前が０ではなく－31079であることを考えるとおそらく最小値は－32767であると考えられる。検証結果を図10に示す。

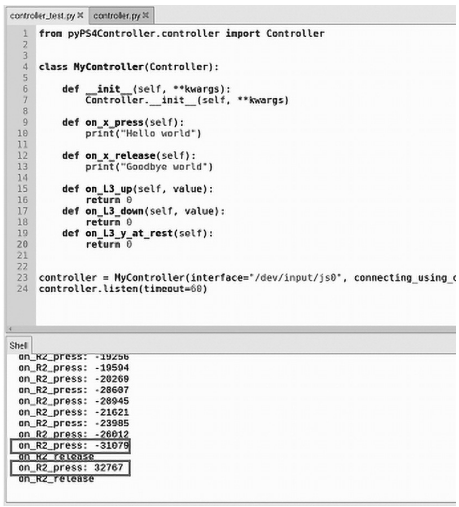

図９　Ｒ２ボタン入力値検証結果

図10　Ｌ３スティック入力値検証結果

アクセル量は出力 0 %から100%まで変化するため，ボタンの入力値「value」をアクセルの出力値「ac」に変換するための式は以下のようになる。

$$ac = value \times \frac{50}{32767} + 50$$

「ac」を引数として関数「ac_on（ac)」を実行させることで，コントローラーの R 2 ボタン・L 2 ボタンの入力により後輪の前進・後退を行うことができる。またボタンを離した時，入力が残らないように，関数「ac_off（)」を実行して出力を 0 %に戻して停止させる。

・画像記録モード

×ボタンと○ボタンの入力に対して，画像記録モードの ON/OFF 切り替えを行うように設計した。×ボタンには画像記録モードのプロセスを開始させる関数「cap_start（)」を割り当て，画像記録を開始させる。○ボタンには画像記録モードのプロセスを停止させる関数「cap_stop()」を割り当て，画像記録を停止させる。

・自動運転モード

△ボタンと□ボタンの入力に対して，自動運転モードの ON/OFF 切り替えを行うように設計した。△ボタンには自動運転モードのプロセスを開始させる関数「auto_start（)」を割り当て，自動運転を開始させる。□ボタンには自動運転モードのプロセスを停止させる関数「auto_stop()」を割り当て，自動運転を停止させる。

上記の手順により，市販のワイヤレスコントローラーを使用して AI スマートカーの操縦，画像記録，自動運転することが可能となった。これにより，改良前に発生していた操作感のタイムラグが解消され, 操作を行うのが容易となった。また，プログラムも簡潔化され容易に変更・改良を行うことができるようになり，アクセル出力も ON/OFF の 2 極化状態から入力の強度によって変化する操作の自由度が高い実験装置に変えることができた。完成した装置を図11に示す。

図11　改良した AI スマートカー

お　わ　り　に

本報では先行研究で作成したAIスマートカーに大幅な改良を加え，先行研究の課題解決の糸口を探った。今後の研究では改良したAIスマートカーを使用して実験を行い，課題が解消されているかを確認していく。また新たな試みとして，カメラの変更や他のセンサを導入することで，結果にどのような変化が起きるのかを検証してみたいと思う。

今回の改良で独自のプログラムを設計したことで，自身のAIスマートカーに対する理解も深まり，改良前に比べてプログラムを簡潔で明瞭ものにすることができた。このことは今後AIスマートカーの研究を進めるにあたって大きな進歩となるだろう。

この研究の最終目標は，実験を学生と共に行っていくことで，学生が自動運転の制御技術に触れ，理解を深めてもらうことである。まだまだ発展途上ではあるが，少しずつ最新技術にも触れられるような教育過程を築いていく懸け橋になれたら幸いである。

最後になりますが，本報での研究を行うにあたり，適切なアドバイスを頂いた中里武彦助教に心より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）: 株式会社アーブ 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］「ラズパイ4B ＋ PS4コントローラ – FRONT」https://wisteriahill.sakura.ne.jp/CMS/WordPress/2021/11/11/raspberry-pi- 4 -b-ps 4 -controller/ （2023.12.25）

［4］「GitHub - ArturSpirin/pyPS 4 Controller: Light module（less than 30KB）without any dependencies designed to provide hooks for PS 4 Controller events」https://github.com/ArturSpirin/pyPS 4 Controller（2023.12.25）

［5］「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）

［6］「【ラズベリーパイ】監視カメラの作り方｜ Python でカメラモジュールを自在に操作 ¦ sozorablog」https://sozorablog.com/camera_shooting/ （2023.12.25）