リアルタイム画像認識を行うAIカーの設計と実装についての研究 ： ―NVIDIA Jetson nanoを利用したJetBotの画像認識メカニズム―

著者	野津 伸治
雑誌名	鳥取看護大学・鳥取短期大学研究紀要
号	83
ページ	31-37
発行年	2021-07-01
出版者	鳥取看護大学・鳥取短期大学
ISSN	2189-8332
URL	http://doi.org/10.24793/00000329

〈研究ノート〉

リアルタイム画像認識を行うAIカーの設計と実装についての研究
―NVIDIA Jetson nanoを利用したJetBotの画像認識メカニズム―

野　津　伸　治[1]

Shinji Notsu：A Study of Design and Implementation an AI Car Which Can Image-recognize the Direction by Using Camera

AI（Artificial Intelligence）を活用する分野のひとつに画像認識がある．このための学習と推論のメカニズムの理解は重要である．これを自律走行カーのリアルタイム制御として実装する．情報科学分野専攻の短期大学生のグループ PBL の教材として利用するための AI カーの設計と実装の研究である．

キーワード：AI カー　画像認識　JetBot　Jetson nano

はじめに

AI の理論と製品化の進展にはこの 10 年目覚ましいものがある．音声認識や画像認識などの分野で新たな手法が数多く提案され，それらはスマートフォンや自動車の自動運転など様々な製品に実装されている．AI における学習もクラウド上のサービスからエッジでの実行まで様々に存在する．特に AI やデータサイエンス分野で優れたライブラリーが存在するプログラミング言語 Python はよく用いられる．

Python を利用してリアルタイム画像認識を行う題材として NVIDIA 社の提案する JetBot[1)]やJetRacer[2)]はエッジ AI コンピュータの Jetson nano[3)]をベースとし，高解像度カメラでの走行コース認識と車体を DC モータで制御する AI カーである．前者は走行スピードも相対的に低速で推論やそれに基づく挙動も低速であるが，後者は数倍の速度でコース認識と推論を行い，駆動部の制御も高速である．

鳥取短期大学でのコンピュータや AI の全体像を学ぶコースの題材としては，メカニカルな部分と電子的な部分のハードウェアと，学習や推論を司るソフトウェアを総合的に学べる好都合の題材である．任意のコースを設定して，数名でのグループで完走するまでリアルタイム画像認識の競技を行うことを計画した．

1．AI カーのハードウェア

コースのリアルタイム画像認識をして自動走行する AI カーのハードウェア構成は，CPU として NVIDIA 社の Jetson nano を，コース識別のカメラ，車体の駆動部として左右の DC モータ・ギアボックス・タイヤ，稼働状態を簡易表示する OLED 表示ユニット，学習データ等のやり取りのための WiFi ユニットから成り立つ．これらの全体像を表したものが図 1 である．

この中でも特にエッジ部分のハードウェアの構成を図示したものが図 2 である．

（1） NVIDIA 社 Jetson nano

GPU を搭載して AI 演算で有効な並列計算ユ

1　鳥取短期大学生活学科

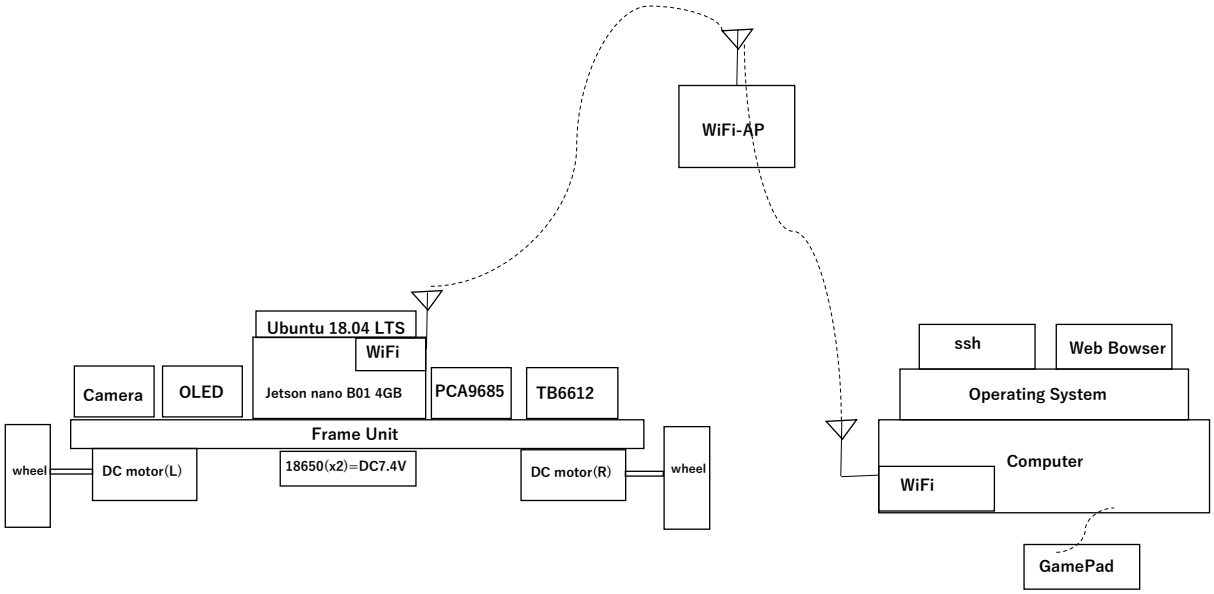

図 1. JetBot のシステム概要

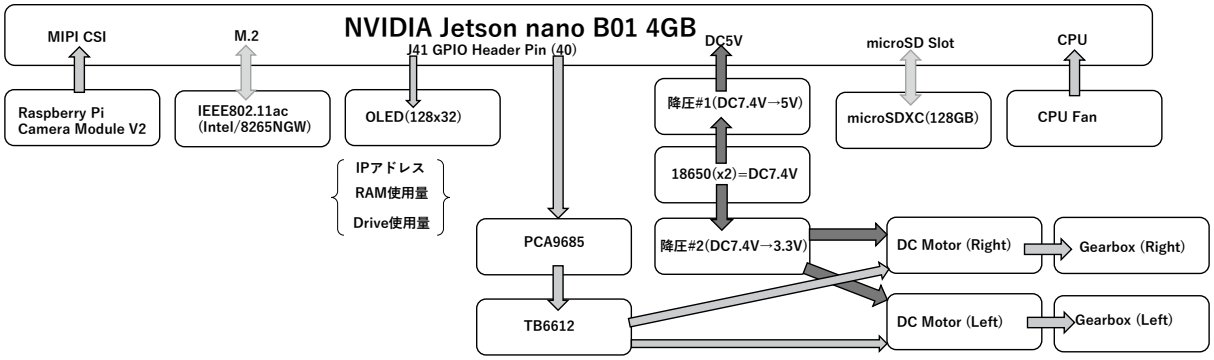

図 2. JetBot の Hardware 構成

ニットが利用できるため，NVIDIA 社の Jetson nano はエッジ AI として組み込み系システムでよく利用されるようになってきた．Jetson nano の仕様は表 1 である．

Jetson nano の J41 ヘッダーGPIO ピンのレイアウトは表 2 のとおりである．

なお，ソフトウェアレベルで Raspberry Pi の GPIO ピンと互換性を保って利用することができる．この GPIO に DC モータ制御を行うパーツや OLED などを接続する．

（2） PWM コントローラ PCA9685

Jetson nano から DC モータ，サーボモータ，ディスプレイなどを制御する場合に I2C 通信方式を用いるために PCA9685 を利用する．なお，Jetson nano とこの PCA9685 の結線は表 3 のとおりである．

（3） DC モータコントローラ TB6612

DC モータの正転・逆転・ブレーキ・ストップを PWM（Pulse Width Modulator）で制御するために 1 デバイスで 2 モータ制御できる TB6612 を利用する．PCA9685 とこの TB6612 の結線は表 4 のとおりである．なお，PCA9685 と TB6612 をひとつの HAT（Hardware Attached on Top）で実装したものを用いることが可能である．

（4） 有機 EL ディスプレイ（OLED）

有機 EL ディスプレイは，0.9 インチで 128×32

表1. NVIDIA Jetson nano B01の仕様

GPU	128-core Maxwell
CPU	Quad-core ARM A57@1.43GHz
Memory	4GB 64bit- LPDDR4 25.6GB/s
Storage	microSD
Video Encoder	4K@30 , 4x 1080p@30 , 9x 720p@30 (H.264/H.265)
Video Decoder	4K@60 , 2x 4K@30 , 8x 1080p@30 , 18x 720p@30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lans
Connectivity	Gigabit Ethernet , M.2 Key.E
Display	HDMI 2.0 and eDP 1.4
USB	3x USB 3.0 , USB 2.0 Micro-B
Others	GPIO , I2C , I2S , SPI , UART
Mechanical	100 mm x 80 mm x 29mm

表2. NVIDIA Jetson nano J41 Header GPIO Pin layout

Pi GPIO	Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO	Pi GPIO
		+3.3V	1	2	+5V		
2		SDA1 (I2C Bus1)	3	4	+5V		
3		SCL1 (I2C Bus1)	5	6	GND		
4	gpio216	AUDIO_MCLK	7	8	TXD0		14
		GND	9	10	RXD0		15
17	gpio50	UART2_RTS	11	12	DAP4_SCLK	gpio79	18
27	gpio14	SPI2_SCK	13	14	GND		
22	gpio194	LCD_TE	15	16	SPI2_CS1	gpio232	23
		+3.3V	17	18	SPI2_CS0	gpio15	24
10	gpio16	SPI_MOSI	19	20	GND		
9	gpio17	SPI_MOSO	21	22	SPI2_MISO	gpio13	25
11	gpio18	SPI_SCLK	23	24	SPI1_CS0	gpio19	8
		GND	25	26	SPI1_CS1	gpio20	7
(0)		ID_SD (I2C Bus2)	27	28	ID_SCL (I2C Bus0)		(1)
5	gpio149	CAM_AF_EN	29	30	GND		
6	gpio200	GPIO_PZ0	31	32	LCD_BL_PWM	gpio168	12
13	gpio38	GPIO_PE6	33	34	GND		
19	gpio76	DAP4_FS	35	36	UART2_CTS	gpio51	16
26	gpio12	SPI2_MOSI	37	38	DAP4_DIN	gpio77	20
		GND	39	40	DAP4_DOUT	gpio78	21

ドット表示できるものを用意して，無線ＬＡＮのIPアドレスやRAMの使用量，microSDカードの使用量を表示させる．ディスプレイとJetson nanoはI2C通信デバイスとして接続する（表5）．そのためにSDA（Serial Data), SCL（Serial Clock），デバイスの電源として3.3VとGNDの4本のケーブルを接続する．

（5） カメラモジュール（Raspberry Pi Camera Module V2)

Jetson nanoのMIPI CSI端子に接続できるカメラモジュールで画像認識に用いるため静止画像で1080ｐ（1920×1080）の画質で撮りうる800万画素のセンサーのものを用いる（表6）．またAIカーが動き回りながら静止画像を撮るので160度の視野角のあるユニットとしてRaspberry Pi専用カメラモジュールであるRaspberry Pi Camera Module V2

表 3. Jetson Nano GPIO と PCA9685 の結線

Jetson nano		PCA9685
物理ピン番号	機能	機能
1	3.3V DC	Vcc
3	I2C_2_SDA	SDA
5	I2C_2_SCL	SCL
9	GND	GND

表 4. PCA9685 と TB6612 の結線

PCA9685	TB6612
機能	機能
Vcc	Vcc
GND	GND
PWM8	PWMA
PWM9	AIN2
PWM10	AIN1
PWM11	BIN1
PWM12	BIN2
PWM13	PWMB

表 5. Jetson nano GPIO と 0.9”OLED-128 × 32 の結線

Jetson nano	OLED
機能	機能
3.3V DC	Vcc
I2C_2_SDA	SDA
I2C_2_SCL	SCL
GND	GND

表 6. Raspberry Pi Camera Module V2 の仕様

Interface	CSI
IC	Sony IMX219
静止画解像度	8メガピクセル
解像度/フレームレート	1080p (1920x1080) / 30fps 720p (1280x720) / 60fps 480p (640x480) / 90fps
外形寸法	25(W)x24(D)x9(H)mm
質量（本体のみ）	5g
動作温度	0～70℃

を Jetson nano で用いる.

（6） WiFi モジュール

ARM 版 Ubuntu 18.04 LTS 側で WiFi モジュールの標準ドライバで稼働するものは Intel 8265NGW があるので，Jetson nano 側の M.2 スロットに装着する．更に外付けアンテナを 2 本接続する．なお，このモジュールは IEEE8021.11ac と Bluetooth 4.2 の通信機能を持っている.

（7） 電源ユニット

電源ユニットとしては，リチウムイオン電池（18650）は DC3.7V，3,000mAh の仕様のものを直列で接続する．これを Jetson nano 本体への DC5.0V の給電と DC モータへの 3.3V の給電を行うことにする．それぞれ降圧コンバータ（DC7.4V から DC5.0V へ，DC7.4V から DC3.3V へ）を用意する.

（8） 車体フレーム

車体フレームを 3D プリンタで自作することが可能である．そのための stl データが JetBot 公式の Github（https://github.com/NVIDIA-AI-IOT/jetbot）で公開されている.

2．AI カーのソフトウェア

エッジ側の Jetson nano の CPU である Tegra 用 OS としては ARM 版 Ubuntu 18.04 LTS がある．これに AI ツールや各種ドライバを包含した SDK(Software Development Kit）を NVIDIA 社が JetPack として提供している（https://developer.nvidia.com/jetson-nano-sd-card-image）．さらにミドルウェアの ROS（Robot Operating System）を追加したものを JetBot として提供されている（https://drive.google.com/file/d/1o08 RPDRZuDloP_o76tCoSngvq 1 CVuCDh/view?usp=sharing）．これらを図示したものが図 3 である.

また，リモートで操作する側のソフトウェア構成を表したものが図 4 である．基本的に OS には依存せず ssh クライアントと Web ブラウザが動作することで開発と操作が行える．ただし，ワイヤレス・ゲームパッド（要 X mode 設定）での操作を行う場合はそのドライバの制約から Windows10 が必要である.

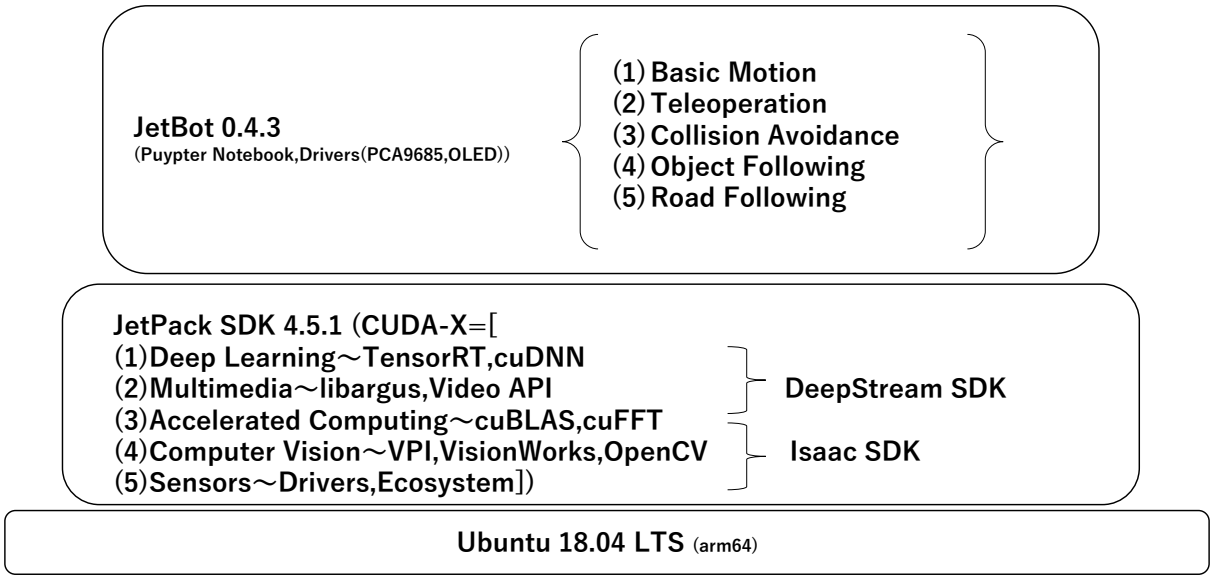

図 3. JetBot の Software 構成図

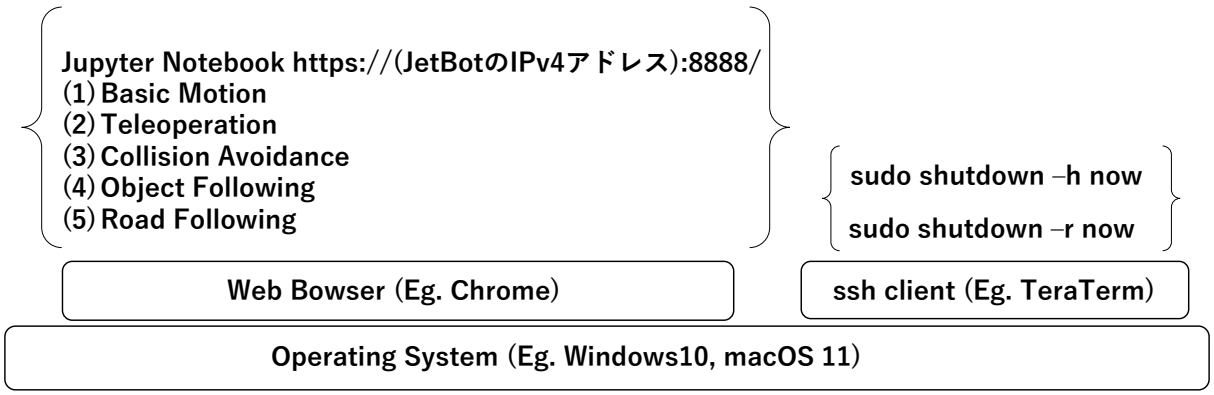

図 4. Remote Device の Software 構成

3．設定

（1） JetBot イメージを 128GB microSD へ書込

ダウンロードした JetBot のイメージファイルは SD Formatter でフォーマットした 128GB の mircoSDXC カードに Etcher を使って書き込む．また Ubuntu で起動後 Gparted を使ってパーティションをイメージファイルの 32GB から 128GB のディスク全体を利用できるように変更する．

（2） WiFi アクセスポイントへの接続

鳥取短期大学の WiFi アクセスポイント（SSID=Cygnus）への接続には，EPA 方式で「PEAP」を，フェーズ 2 認証で「MSCHAPV2」を，「CA 証明書を検証しない」に，それぞれ設定する必要がある．

（3） OS と各種アプリの更新

最初の設定のみモニタとキーボード，マウスを接続して行うが，その後はこれらを外したヘッドレスモードでリモート操作を行う．操作する側は，例えば Windows 10 の場合 TeraTerm などで ssh 接続を行う．システムを最新版に更新するために sudo apt update をしてから sudo apt upgrade を行い，sudo shutdown –r now を行う．

ダウンロードとインストールを反映するために再起動後 sudo apt dist-upgrade を実行してから再度リ

ブートする．その後に sudo apt install nvidia-jetpack をしてから sudo apt show nvidia-jetpack –a で現状を確認できる．

（4） ヘッドレスモードの JetBot をリモート操作

リモート操作される側の JetBot はバッテリ駆動で起動後，JetBot が自動的に WiFi- に接続されるので OLED に表示される wlan0 の IP アドレスを確認する．

リモート操作する側の Web ブラウザで以下のように URL を入力する：

http://（JetBot の IP アドレス）：8888

JetBot 側の Jupyter Notebook が起動するのでパスワードを入れて制御用 Python プログラムの編集と実行を行う．

（5） JetBot のハードウェアの基本動作の確認

Jupyter Notebook で左右のモータの動作や車体の前後左右の移動を確認するために以下を実行する（図 5）：/jetbot/notebooks/basic_motion.ipynb

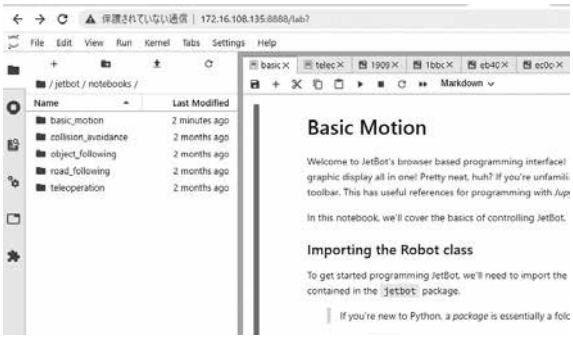

図 5. Jupyter Notebook で basic_motion.ipynb 実行

以下で具体的なコードを例示すると Robot クラスのインポートをしてから，インスタンスの初期化をする．左モータの 30% を速度で反時計回りに動かしてから停止させる．time パッケージを利用して状態を維持する時間を指定する．左右のモータの速度を変えて駆動部分の動作を確認する．

```
from jetbot import Robot
robot = Robot()
robot.left(speed=0.3)
robot.stop()
import time
robot.left(0.3)
time.sleep(0.5)
robot.stop()
robot.set_motors(0.3, 0.6)
time.sleep(1.0)
robot.stop()
robot.left_motor.value = 0.3
robot.right_motor.value = 0.6
time.sleep(1.0)
robot.left_motor.value = 0.0
robot.right_motor.value = 0.0
```

プログラムと車体の動作が逆の場合は DC モータの結線を左右入れ替える．ここまでで JetBot の駆動系の動作確認が完了する．

（6） ゲームパッドで JetBot の移動やカメラの操作

リモート操作する側の機器にワイヤレス・ゲームパッドを接続して挙動を確認する．例えば，Logicool 社の F710 を Windows10 コンピュータに X Mode に設定してからワイヤレスで接続する．各ボタンの定義を以下の URL に接続して確認する：

https://html5gamepad.com/

まずは機器の index の値を確認する．本機の場合 1 から 0 へ変更した．また左右のモータの正転・逆転が，それぞれ AXIS1 と AXIS2 に割り当てられ，カメラの撮影が B5 ボタンに対応していることを確認する．必要に応じて対応ボタンの変更をソースコードに対して行う．

ワイヤレス・ゲームッドによる JetBot のリモート操作による前後左右の移動とカメラによる撮影の動作を確認するために以下を実行する（図 6）：

/jetbot/notebooks/teleoperation.ipynb

ここまでで JetBot に走行コースを試走させて車体の状態からとるべき制御行動の推論のもととなる

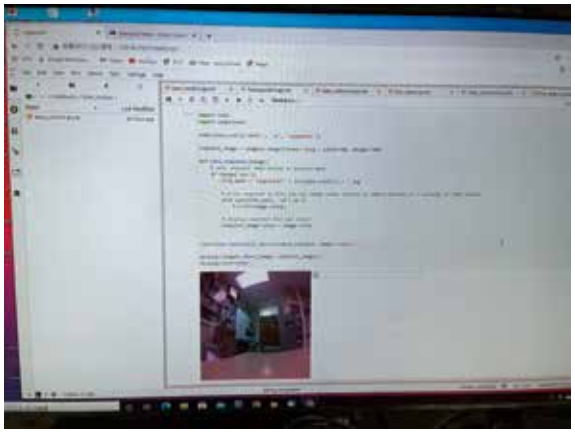

図 6. teleoperation.ipynb で学習データ撮影

学習データを蓄積することが可能となった.

（7） 衝突回避

JetBot のカメラで位置や角度を変えて物体 OB を認識している映像 100 枚と認識していない映像を 100 枚撮影してデータセットを作る（/jetbot/notebooks/collison_avoidance.ipynb）.

リモートの Windows 10 コンピュータ上 Anaconda の Pytorch 環境で Jupyter Notebook を利用できるようにする．JetBot で作成したデータセット（/dataset）を転送して同じ collision_avoidance.ipynb を実行できるようにして高速に学習をさせることで推論用モデル（best_model.pth）を導き出す.

リモート側で得た推論モデルを JetBot へ転送してから，これに基づき実際の物体 OB への衝突回避を実行させることができる．これらの一連の流れをフローチャートにしたものが図 7 である.

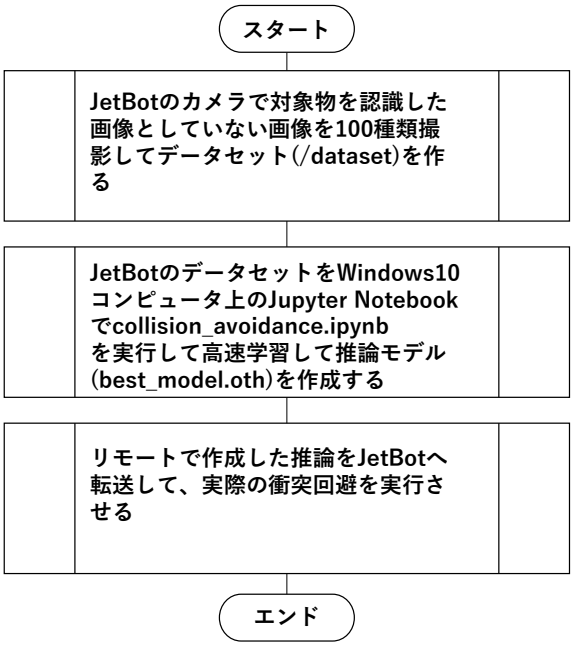

図 7. 衝突回避のフローチャート

おわりに

情報科学系短期大学生が AI 技術の一つであるリアルタイム画像認識を，自立走行車の設計と実装を通して理解するためのグループ PBL の教材として，ハードウェアとソフトウェアの総合的な理解につながる．任意に設定されたコースを自律走行の成否が推論モデルの精度とそれを導くための教師あり学習のデータの質と量である．今後任意に設定したコース走行を競技（走行時間とコース走行の精度，学習速度，学習データ数を計測比較）することで学生たちに確認させたいと考える.

引用・参考文献

1 ）JetBot https://github.com/NVIDIA-AI-IOT/jetbot（2021.3.31）.

2 ）JetRacer https://github.com/NVIDIA-AI-IOT/jetracer（2021.3.31）.

3）Jetson nano https://www.rs-online.com/designspark/jetson-nano-40-pin-gpio-1-cn（2021.3.31）.