7 Raspberry Pi を用いたロボットカー制御の研究

千葉　裕幸

指導教員　佐々木　建

1.　はじめに

RaspBerry Pi は安価かつポピュラーであり、温度計やドローン制御などのように、組込みシステムによく使用されている。私は授業や競技大会で学んだ組込みシステムの制御に興味があり、卒業研究で取り組んでみたいと考えた。

卒業研究のテーマも、ET ロボコン競技会で学んだ知識を生かしたいと思い、Raspberry Pi を活用しての、ロボットカー制御の研究をテーマとした。

2.　研究概要

2.1　システム概要

ハードウェアは Raspberry Pi と Kuman ロボットカーおよび USB カメラを用いた。

開発の流れは、パソコンで作成したプログラムを走行体に装着されている Raspberry Pi に書き込み、走行させるものである。

内容としては走行体が USB カメラにてコースを認識し、基本的にはライントレースし走行する。また、障害物がある場合は「止まる、回避する」パターンや、信号が赤の場合は止まるパターンなど様々な環境の変化にも対応するシステムを作成した。

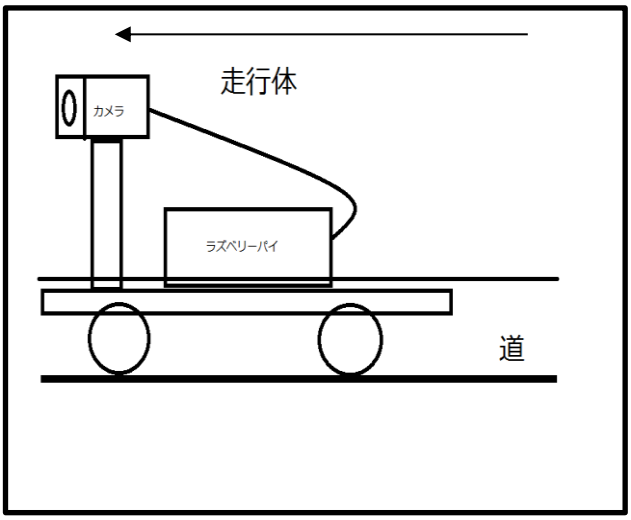

図１　走行イメージ

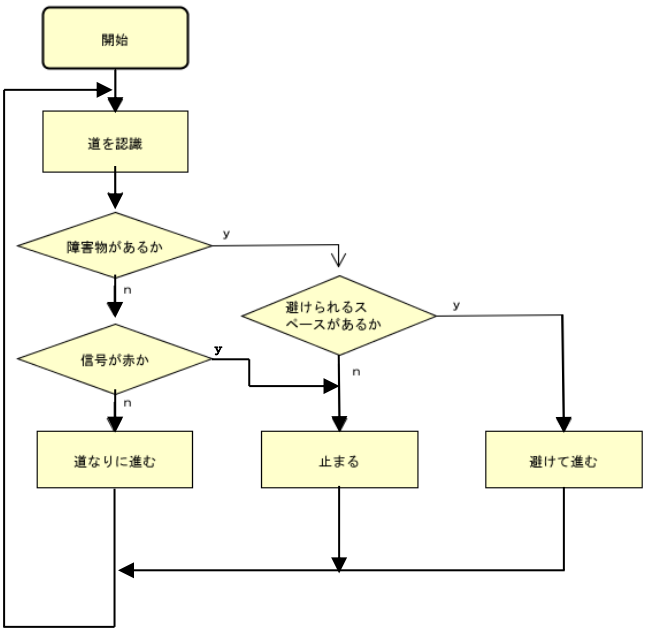

図２　システムの流れ

2.2　開発環境について

本研究での開発環境は、次ページの表１である。

実際に使用した走行体を次ページの図３、図４、図５に示す。

表 1　開発環境

・OS	Raspbian Windows10pro
・ハードウェア	Kuman　ロボットカー USB camera Raspberry Pi3 Model B
・使用言語	python3
・ライブラリ	OpenCV RPi.GPIO

図 3　走行体（正面）

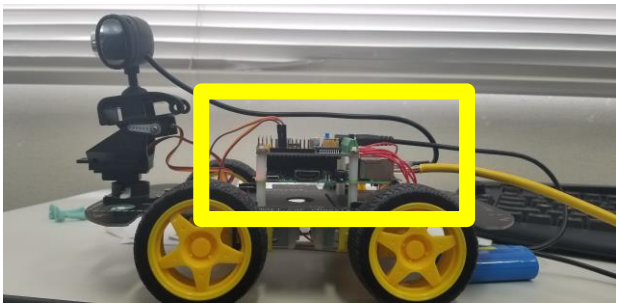

図 4　走行体（横）

図 5　走行体（上）　黄色枠の部分が Raspberry Pi

2.3　コースについて

今回の研究で使用するコースは、走るべき道とそれ以外がはっきりわかるものが好ましいため、ET ロボコン競技会で配布された練習用コースを使用する。

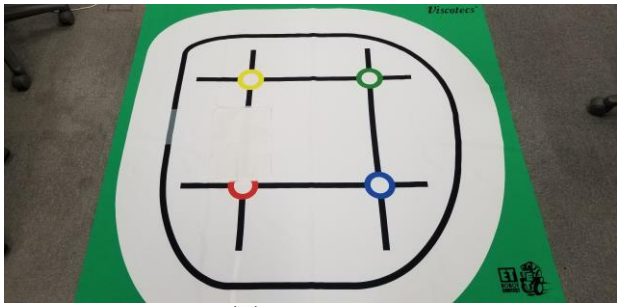

図 6　コース

3.　実装機能説明

3.1　ライントレース

今回の研究で走行の基本となるライントレースの機能について説明する。

まず、ライントレースについて、走行体が進む経路の求め方である。辿るべきラインを２値化、ラインが直線の場合はラインの中央を走行体の経路に取るのが安定した走行ができると考えられる。問題はカーブのときである。カーブを曲がるための処理は色々とあるが、辿るべきラインに一定間隔に２点を取る。その２点間を結ぶ直線を引き角度を求め、その角度の大きさによって左右のタイヤのモーター回転速度を調節しライントレースする手法で行った。

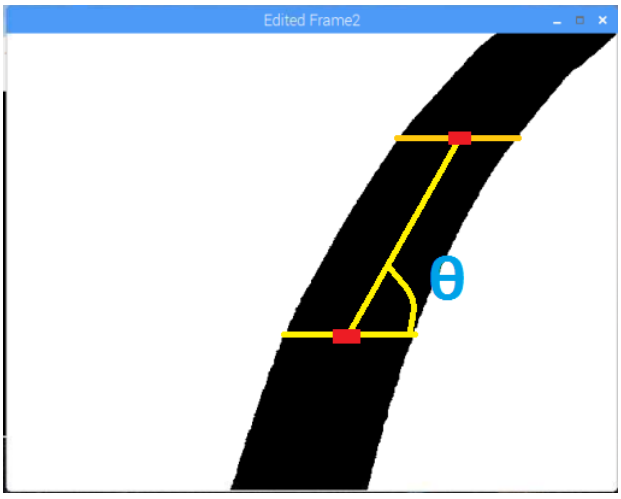

図 7　ライン経路の検出の様子

3.2　障害物の対応

障害物を回避する､回避が不可能ならば止まる､という課題を用意しているため、その機能について説明する｡実際に使用した障害物を図 8 に示す。

まず､障害物を検知するために 2 値化処理をし、輪郭を抽出する。障害物以外の輪郭が検出されないように、検出される輪郭のサイズの範囲を決め範囲外のものをノイズとして除去することで障害物を検出した。

次に、障害物に衝突することなく回避するには障害物と走行体との間に十分な距離が必要なため、それを求めるためにカメラを固定することで、ライン上の座標から距離を推測する。制御は障害物を検出したら、コース中央寄りに回避し、推測した衝突までの距離が一定以下になったらモーター出力を止める制御を行うように設定した。

図 8　障害物

3.3　信号の対応

信号が赤なら止まる、青ならば進むという機能について説明する。実際に使用した信号を図 9 に示す。

使用した信号について箱を用い、表に青、裏に赤とすることで、箱を裏返すことで信号の変化をさせることにした。信号の検出は 3.2 で説明した障害物の検出と方法は同じで、問題は障害物と信号との差別化をすることである。その手法として輪郭抽出を行った後、輪郭内部の中心付近の座標から色を検出、青や赤ならば信号、それ以外を障害物と分けることで差別化を行った。

図 9　信号

4　検証

今回、制作した走行体を制御し、良かった点と課題点について検証する。

4.1　良かった点

- OpenCV は機能が豊富で、すでにある関数を使うことで様々なことが簡単にできるという点である。実際に今回の研究で 2 値化処理や輪郭抽出の処理を行ったがどちらも OpenCV の関数を使うことで簡単にできた。
- ライントレースの機能を実装する際は、カメラを使った様々な手法を知ることができた。
- ライントレースに関してはコースアウトすることがほぼなくなった。
- 障害物の検出がしっかりとできた。
- 信号の検出ができ、信号の色によって止まる進むことができた。

4.2　課題点

- 走行体について、走行体の土台が歪んでおり左側の車輪が浮いてしまい空回りしてしまう。
- 走行体の制御に関して、モーターの制御に pwm 制御法を用いたがデューティーサイクルが 10%を下回るとモーターがパワー不足となり、動かなくなってしまう。だがデューティーサイクルを 15%以上にするとカメラで撮影している画像の変化が早すぎるため、画像の処理が追い付かなくなってしまう。そのため、カーブを曲がる際など片側のタイヤのモーターが止ってしまう。
- ライントレースをする際に片側のタイヤが止まってしまうため、細かい調節ができず左右に大きくぶれてしまう
- 障害物に関して、障害物を検出することができるが、バッテリーの残量が少なくなると回避する確率が二分の一程度になってしまう。

・障害物の外側に黒の紙を貼って２値化したさいに判別しやすくしているが、その素材が光を反射して白色に検出される時がある。
・環境光によって画像内の明度が変化してしまい環境の条件に合わせた値を設定しなくてはならない。
・４輪駆動のため動作が思い操作が重く感じられるので、２輪駆動にすると動作が軽くなる可能性がある。

5　おわりに

RaspBerry Pi を用いたロボットカー制御の研究を通して、使用した言語やライブラリに対する理解を深めつつ、様々な課題点が見えた。

当初の計画では、走行体の機能としてカメラ部分のサーボモーターを使い、コースであるラインの検出をキャリブレーションとして行うこと、及び障害物を回避する際の十分なスペースの確認を行う予定であったが、サーボモーターが動作せず今回は断念した。おそらく RaspBerry Pi 本体の pin モードの設定を変更する必要があると思われる。そのため、計画を変更しキャリブレーション及びスペースの確認の機能を省き、残りの機能を実装した。

4.2 での課題点の中で土台が歪んでいると、挙げたが修繕策として、走行体に「おもり」を載せることで、車輪の浮きをなくすという方法が考えられる。また、走行体の質量が増えるので、他の課題点である、モーターのパワー不足や、細かい調節ができないことによる左右のブレが解消されると思われる。

今回の研究での成果物を走行させての使い勝手だが、修繕点、課題点を残しているが、実装した機能は満たしていると考えられる。

今回の研究を通して、あまり触れることのなかった python やリアルタイムでの画像処理や走行体の制御を行い、一つ一つの機能を実装させ狙い通りの動きをさせることの大変さを知った。障害物の機能を例にあげると、「障害物を回避する」と言葉では簡単に聞こえるが回避するまでに障害物を検出、モーターを止める、距離を測定など、いくつもの処理を挟むので、とても考えさせられた。また、いくつもの機能が干渉しあって、予期せぬ走りを見せる時もあり、様々な環境の条件を考えなければならないということを知ることができた。

しかし、試行錯誤をしていく中で、自分の考えたものをそのまま実装できたことで、自分自身の力に自信を持つことができ、画像処理や走行体の制御について理解を深めることができたのを実感することができた。

6　参考文献・参考サイト

Python+OpenCV で Web カメラの画像を取り込んで処理して表示する話
(https://ensekitt.hatenablog.com/entry/2017/12/19/200000)
Raspberry Pi で自動運転やってみる
(https://qiita.com/stnk20/items/f614ae1471b9e555708a)
Raspberry Pi の GPIO 制御
(https://karaage.hatenadiary.jp/entry/2017/02/10/073000)
輪郭の階層情報
(http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html)