.

コンピュータ理工学特別研究報告書

題目

Raspberry Pi を用いた模型自動車の制御

-車載カメラ画像の鳥瞰変換を用いたステアリング制御-

学生証番号　1245126

氏名　水船　克洸

提出日　平成 28 年 2 月 1 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

本研究ではRaspberry Piを用いた模型自動車の制御を行う.模型自動車の前方を単眼カメラで撮影し,路面上の車線を検出する.この結果に応じてタイヤ角度の制御と前進・後退の制御を行い車線に沿って滑らかに走行させる.

単眼カメラで撮影した路面は，手前ほど狭く奥ほど広い範囲を撮影している.そのため,画像上で見た路面と自動車の位置関係は単純ではない．この問題を解決するために，撮影した画像上での位置情報に鳥瞰変換を施し，路面を真上から見た状態での位置情報に変換する.これにより検出した車線の正確な位置を把握することができる.

自動車はタイヤを一定の角度に保ったまま進むと一点を中心とする円運動を行う.このときの円運動はホイールベースと前輪のタイヤの角度の関係から求めることができる.このことを利用すると現在の車両位置からのすべての走行経路を計算することができる.

開発したプログラムはカメラを使って路面を撮影し車線を検出する.検出した車線から一つの目標点を決定し，この目標点に向かって模型自動車を走らせる.目標点に達するためのタイヤ角度は，計算した全ての走行経路と目標点との距離を比較し,最も距離の近い走行経路のタイヤの角度として計算した.車線のカーブが大きく曲がりきれないと判断した場合には,ある程度後退させてからまた前進させることで曲がりきれるようにした.実際に走らせた結果，模型自動車は車線の上をスムーズに走行することができた.課題としては,車線の検出が上手くいかない場合があるため，ノイズが多い路面では安定して走ることができなかった.

目次

1章 序論

コンピュータと言えば，ノートパソコンのように，ディスプレイやキーボード・マウスを備えたものを想像する人が多いが，家電のような専用装置にもコンピュータが利用されている.つまり，私たちの身の回りには，コンピュータとは見えない形で,多くのコンピュータが存在する.パソコンがさまざまな用途に使うことができるのに対し，専用装置は用途が決まっている.コンピュータが組み込まれた専用装置を「組み込み機器」といい，組み込み機器を制御するコンピュータシステムを「組み込みシステム」という.また,組み込み機器に利用する小型CPUを「マイクロコントローラ（マイコン)」とよぶことがある.

自動車にもコンピュータが組み込まれており，高度な制御が可能になっている.自動車に搭載されているCPUの数は非常に多く，その数も時代が進むにつれ増えている.1980年代には約8個,2000年には約30個,2005年には約50個,そして現在,数の多いもので約80個ものCPUを搭載している自動車がある.このことから自動車の性能向上にはコンピュータ技術の影響が大きいことがわかる.

最近の自動車の進化として注目すべきは自動運転である．自動運転は，自動車産業や自動車社会に革新的な変化をもたらす可能性がある技術である.自動運転は路面状況や周囲の車両や歩行者などの環境をカメラやレーダ,ソナーなどの複数のセンサを用いて認識し,認識結果に基づいて自動車を制御することで実現できる.自動運転には多くの技術が必要であるが，中でもカメラを用いて周囲を撮影し,それを画像処理することで周囲の状況を認識する技術が大きな役割を果たしている.

自動運転における車載カメラの役割に，車線の認識や周囲の車両・歩行者など障害物の検出がある．単眼の車載カメラで車線や障害物を検出し，その結果に基づいて自動車を制御するためには，画像に映った車線や障害物の見かけの位置を，自動車との実際の位置関係に換算する必要がある．これを行う一つの方法として，車載カメラ画像を鳥瞰変換することが知られている[2]．本研究では，車載カメラで検出した車線情報を鳥瞰変換したのち，自動車の進行方向を制御する方法について研究した．本研究の目的は，車線検出とそれに基づく自動運転の基本的な技術を自作することによって，画像処理技術，組み込みシステム技術，自動運転に利用されるコンピュータ技術などを取得することである.その中でも，画像を鳥瞰変換し車線の正確な位置関係を把握した上で適切なステアリング制御を行うことが本研究の目標である.

今回の研究のために実車を利用することは容易でない.そこで,Raspberry Piにカメラモジュールを接続したものを，車載カメラと車載コンピュータに見立てた．実車のモデルとして，TAMIYA 社のバギー工作セットに自作の加工を加え

たものを利用した．地面として大きな紙面を用い，紙面に車線を模擬する線を描いた．そして，紙面に描いた車線を抽出し，車線を自動的にトレースするプログラムを開発した．

本論文は 2 章で作成した模型自動車について説明し,3 章で鳥瞰変換とステアリング手法について説明する.次いで,4 章ではカメラで撮影した路面の車線検出とステアリング制御により自動運転を行う実験と,その結果について説明する.5 章で結論を述べる.

2 章 Raspberry Pi を用いた模型自動車

2.1　模型自動車のシステム構成

模型自動車のシステム構成を図 1 に示す．また,模型自動車の外観を図 2 に示す.模型自動車は前方を観察するカメラを備えている．このカメラで撮影した画像を Raspberry Pi で処理し，車線を検出する．検出した車線を滑らかにトレースするように，模型自動車のステアリング（前輪の方向）と駆動輪を制御する．

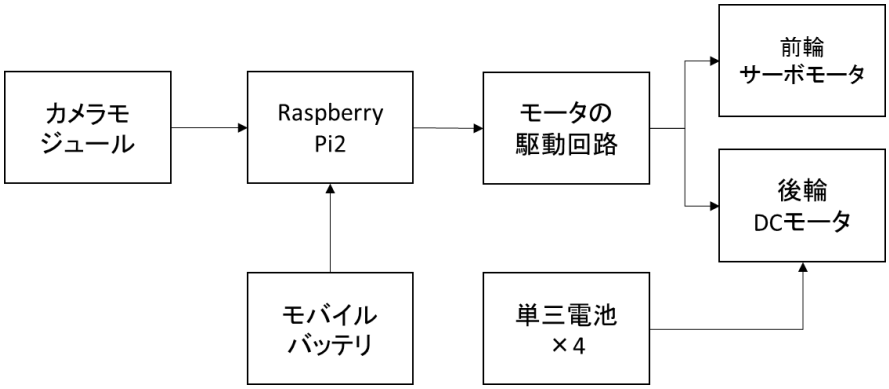


図 1　模型自動車のシステム構成図

図 1 を構成する要素について説明する.

(1) Raspberry Pi

Raspberry Pi はカードサイズのコンピュータで，Linux 系の OS が動作する．ディスプレイ・キーボード・マウス・ネットワークを接続することで通常のパソコンとして利用できる(図 3)．また，GPIO 端子を用いて専用回路を制御することで,組み込み機器のコントローラとして利用することができる.その場合，ディスプレイ・キーボードなどは，用途に応じて，取り去る.

(2)カメラモジュール

5 メガピクセルの画像センサとレンズから構成されており,2592×1944 画素の静止画と,1920×1080 画素の動画を記録することができる．図 2 からわかるように，カメラは模型自動車の上方から斜め下に向かって撮影しているため，車体先端から約 9cm が死角になる.

(3)モータ

前輪のステアリング用にサーボモータを用いる(図 4).前輪をステアリングさせるための機構は,アッカーマンタイプステアリング機構を用いており，タイロッドを左右に動かすことでタイヤの向きを変える.L 型の金具をタイロッドに半

田付けすることでサーボホーンとタイロッドをつなげている.

ステアリングの可動範囲において，サーボモータの軸角度を左右に 14 段階調節可能である．従って，タイヤの角度も 14 段階調節することができる. しかし 1 段階で変化するタイヤの角度は十分に小さいため, 2 段階ずつ調節するようにした. よって左右に 7 段階ずつ調節できる.

車両の前進・後退については, 後輪を駆動する DC モータの回転方向を変えることで行う．サーボモータと DC モータのドライブ回路を図 5 に示す.

(4) 電源

システム全体は DC・5V で動作する．当初，5V・1A のモバイルバッテリだけで動作させる計画であったが，動作が安定しないため，後輪の DC モータのみ単三×4 本のバッテリボックスを追加しそこから電力を供給する.

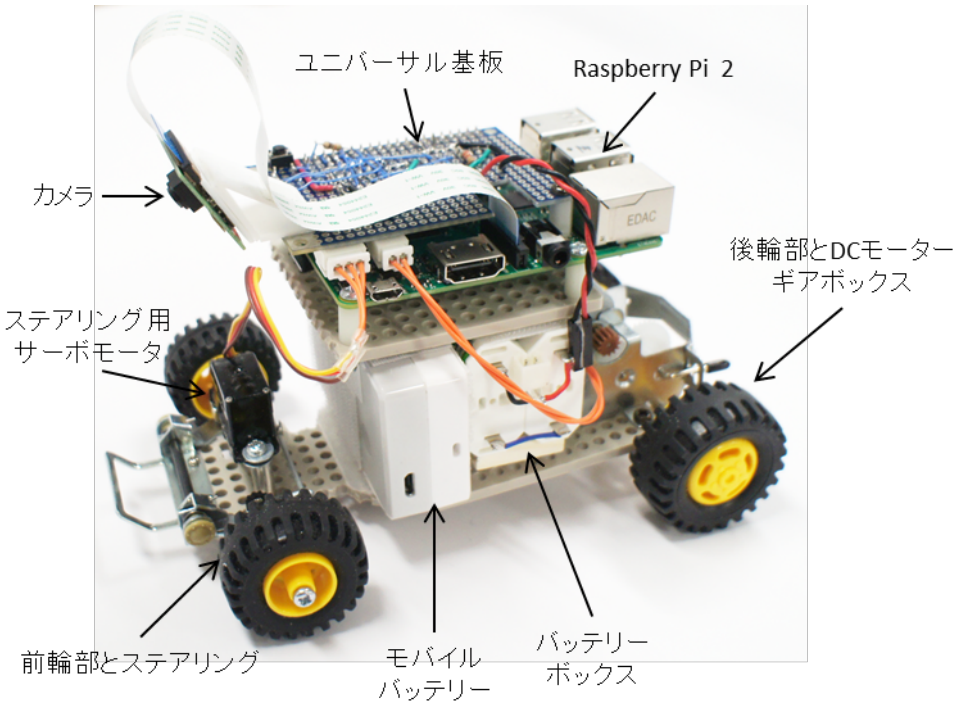


図 2 模型自動車の外観

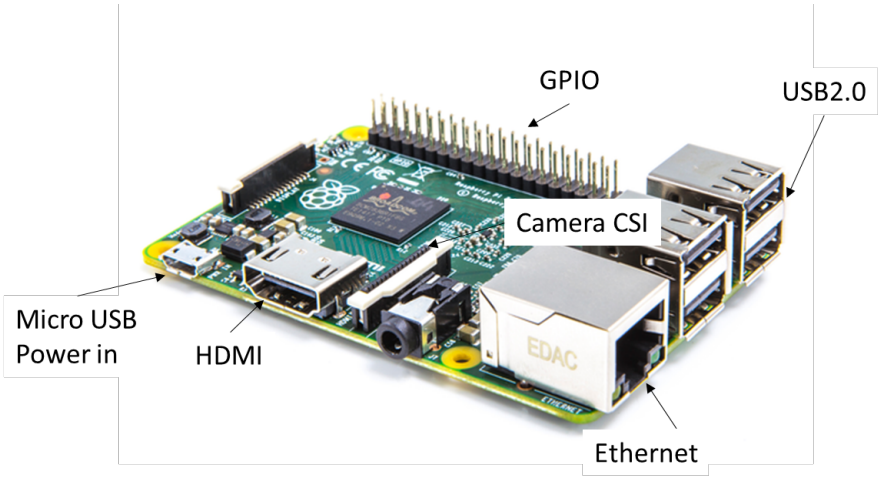


図 3　Raspberry Pi2 の外観

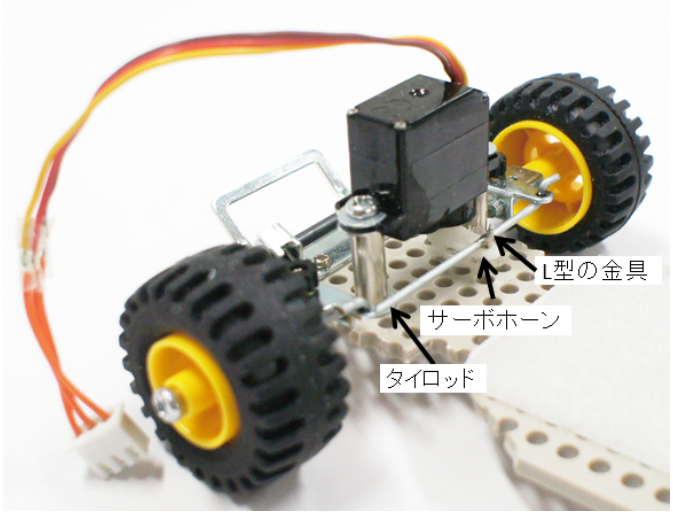


図 4　サーボモータとステアリング部

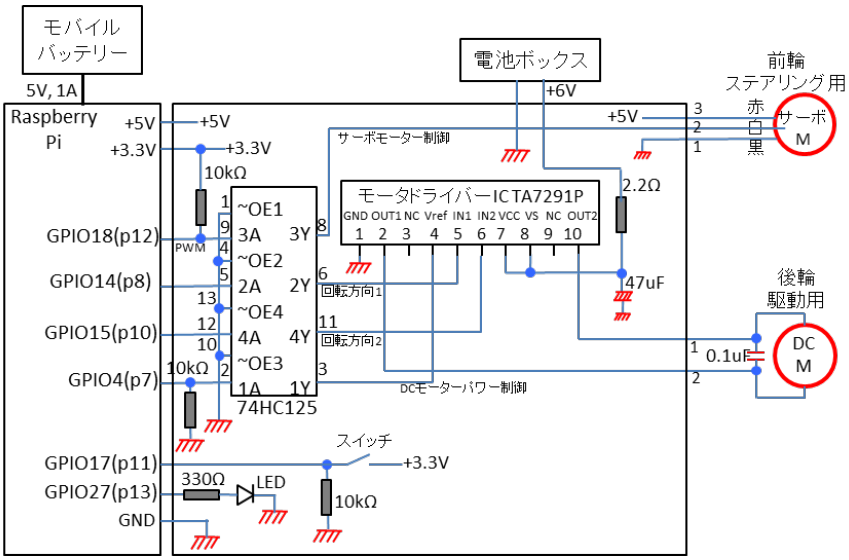


図 5　モータドライブ回路

2.2　ソフトウエアの開発環境

開発にあたって使用した環境は次の表 1 の通りである.

表 1　開発環境

プログラム言語	C 言語
ライブラリ	WiringPi, OpenCV
リモートデスクトップ	xrdp, Microsoft Remote Desktop, MacBookAir

(1) プログラム言語

プログラムの開発に C 言語を用いた. Raspberry Pi のプログラム開発は Python を使うことが多い. しかし, Python はスクリプト言語であるため, 今回の研究で扱う画像処理のような処理の重い作業には向いていない. そのためコンパイラ言語である C 言語を用いて開発を行い, 処理の高速化を図った.

(2) WiringPi[3]

Raspberry Pi の GPIO を C 言語から制御するライブラリとして WiringPi を用いた.

(3) OpenCV

インテルが開発した, 画像処理・画像解析および機械学習等の機能を持つ

C/C++, Java, Python, MATLAB 用ライブラリである.

(4)リモートデスクトップ

Raspberry Pi を操作するためには,ディスプレイ・キーボード・マウスを接続しなければならない.テストのために模型自動車を動作させる場合, これらの機器は邪魔になる.そこでネットワークに接続された Raspberry Pi を別のコンピュータから操作するリモートデスクトップを利用した.なお,Raspberry Pi には有線 LAN のインターフェースしかないため,USB 端子に無線 LAN アダプタを接続し, Wi-Fi 通信を行えるようにした.その接続手順は次の通りである.

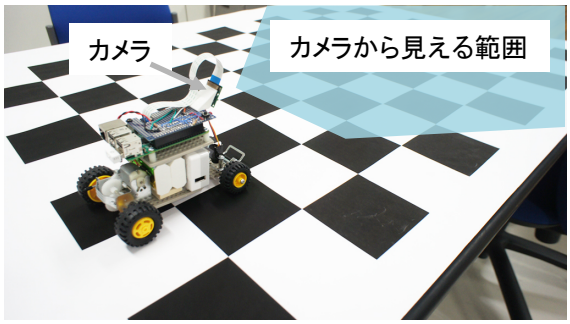
(1) MacBookAir のインターネット共有機能を使い MacBookAir を Wi-FiLan のアクセスポイントとして設定し Raspberry Pi を接続する.
(2) Raspberry Pi 側に xrdp というリモートデスクトップのためのソフトウエアを MacBookAir 側に xrdp クライアント(Microsoft Remote Desktop)をインストールする.
(3) Microsoft Remote Desktop を起動し Raspberry Pi の IP アドレスを入力することで MacBookAir のディスプレイ・キーボード・マウスを使って Raspberry Pi の操作を行うことができる.

3 章　カメラ画像の鳥瞰変換とステアリングの制御

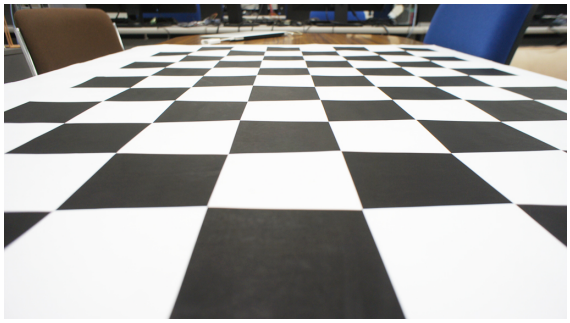
3.1　カメラ画像の鳥瞰変換

本研究では，カメラモジュールで車両の前方を撮影し,路面の情報を得ている．撮影した画像は手前ほど狭く，奥にいくほど広い範囲をとらえている．このため，画像上で見た路面と自動車の位置関係は単純ではない．この問題を解決するために，斜め上方向から路面を撮影した画像を鳥瞰変換する．

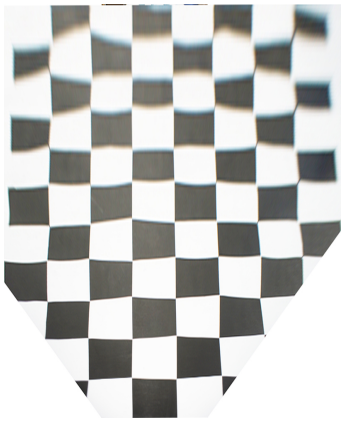
図 6 を用いて鳥瞰変換について説明する．図 6(a)は模型自動車が路面を観察している様子である．説明のために，路面には規則的な市松模様を描いた．図 6(b)は，模型自動車のカメラから路面を撮影した画像の例である．路面は同じ正方形が並んだ市松模様であるが，模型自動車に近い側は大きく見え，遠い側は小さく見えている．図 6(c)は，カメラから見えた路面を鳥瞰変換した例である．鳥瞰変換することで，遠近の違いによる見え方が補正されていることがわかる．ただし，近い側の画像は鮮明に変換されているが，遠い側の画像は不鮮明である．このような２次元画像から２次元画像への変換を鳥瞰変換とよぶ．この幾何変換は平面射影変換で表すことができる．平面射影変換は，２次元同次座標を用いると，3×3 の行列を乗算することで表現できる．



(a) 模型自動車が路面を観察している様子



(b) 模型自動車のカメラから見た路面



(c) 路面を鳥瞰変換した結果

図 6 鳥瞰変換の説明図

(X, Y) を模型自動車の先端を原点とした鳥瞰画像の 2 次元座標, (x, y) を撮影した画像の 2 次元座標とする. (x, y) を (X, Y) に変換する式を, 同次座標を用いて完成させる. 2 つの座標平面の関係を明らかにするために, 図 7 のように, カメラ画像の適当な 9 箇所の画素に印をつけ路面を撮影する. 印をつけた画素に対応する実際の路面の位置を定規で調べる. その結果を表 2 に示す.

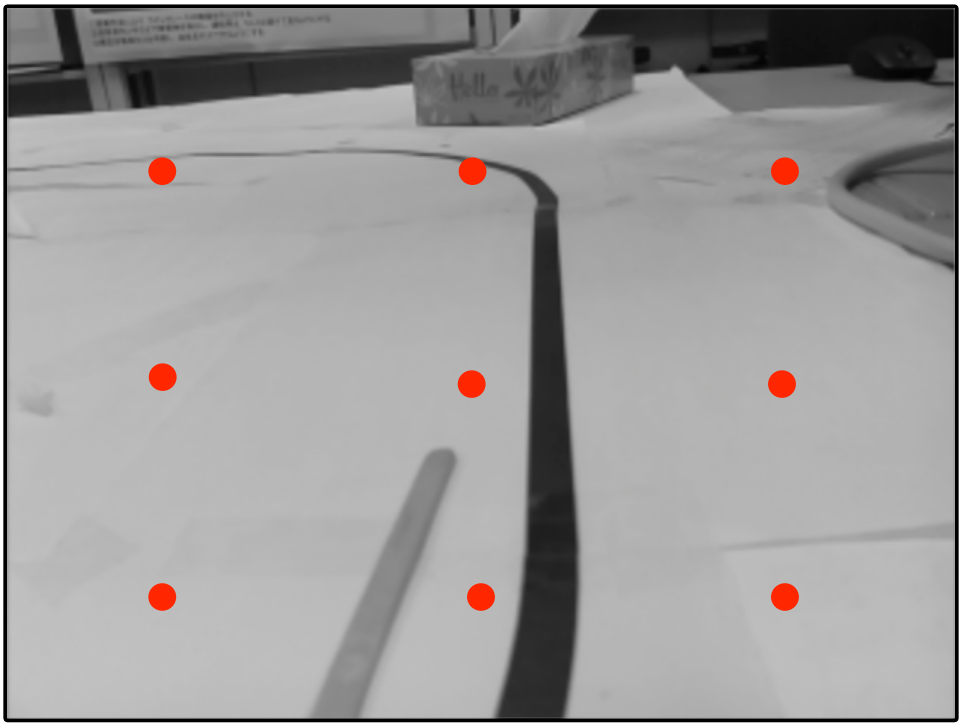


図 7 画像の座標と路面の座標を関係づける手順を説明する図

表 2　カメラ画像と実際の路面の対応

カメラ画像上の座標(x, y) 単位は画素	自動車を原点とした座標平面(X, Y) 単位は mm
20, 230	−71, 97
160, 230	−3, 92
300, 230	60, 90
20, 140	−114, 206
160, 140	4, 192
300, 140	99, 185
20, 70	−215, 468
160, 70	0, 437
300, 70	200, 409

(x, y)と(X, Y)の対応関係は，同次座標を用いると式(1)のように記述することができる．ここで，a〜h はある定数，等号は同次座標における等号である．式(1)に表 2 の関係を代入し，最小二乗法で解くと式(2)を得た．

$$\begin{pmatrix} \mathrm{X} \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \cdots (1)$$

$$\begin{pmatrix} \mathrm{X} \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} 4.79 & -0.142 & -781 \\ -0.536 & -4.19 & 1909 \\ -0.0002 & 0.0369 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \cdots (2)$$

3.2　タイヤのステアリング角度と自動車の動き

ここではタイヤのステアリング角度と自動車の運動について説明する. 自動車は，前輪のタイヤを左右に曲げることで，方向が決まる. タイヤの方向角度が０度以外のとき, 自動車はある一点を中心とした円を描くように進む[1]. ある一点とは，図 8 に示すように，後輪の車軸を水平に延長した線と外側前輪の中心から直角に伸ばした線の交点である. 円の中心から外側前輪までの距離を回転半径とよぶ. 回転半径を求める式を次に示す.

$$\mathrm{R} = \mathrm{W} / \sin\theta \quad \cdots (3)$$

ここで，R:回転半径, W:ホイールベース, θ：外側前輪の切れ角，である.

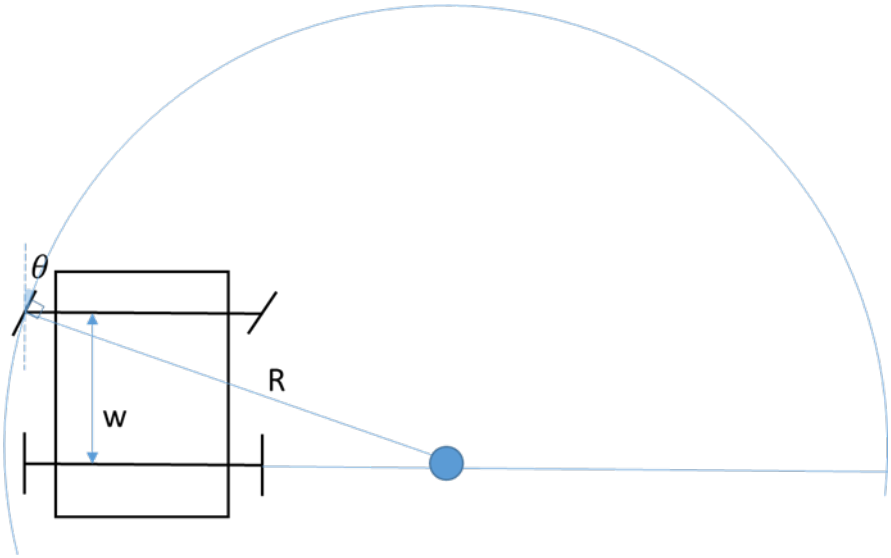


図 8　前輪の方向と自動車の軌跡

3.3　ステアリング制御のめの提案手法

ステアリングを制御する提案手法のフローチャートを図 9 に示す. まず，路面の画像を撮影する. その画像を処理し車線を抽出する．抽出した車線領域の中で, 最も車両側の中央を画像上での目標点とする．目標点を鳥瞰変換し，それを実目標点とする. 次に, 現在の車両位置から可能なすべての走行経路を計算する. それらの走行経路の中で，実目標点に最も近い経路を求める．その経路に対する前輪の角度を得る．最後に，サーボモータを制御して，前輪を求めた角度に設定し，車両を進行させる.

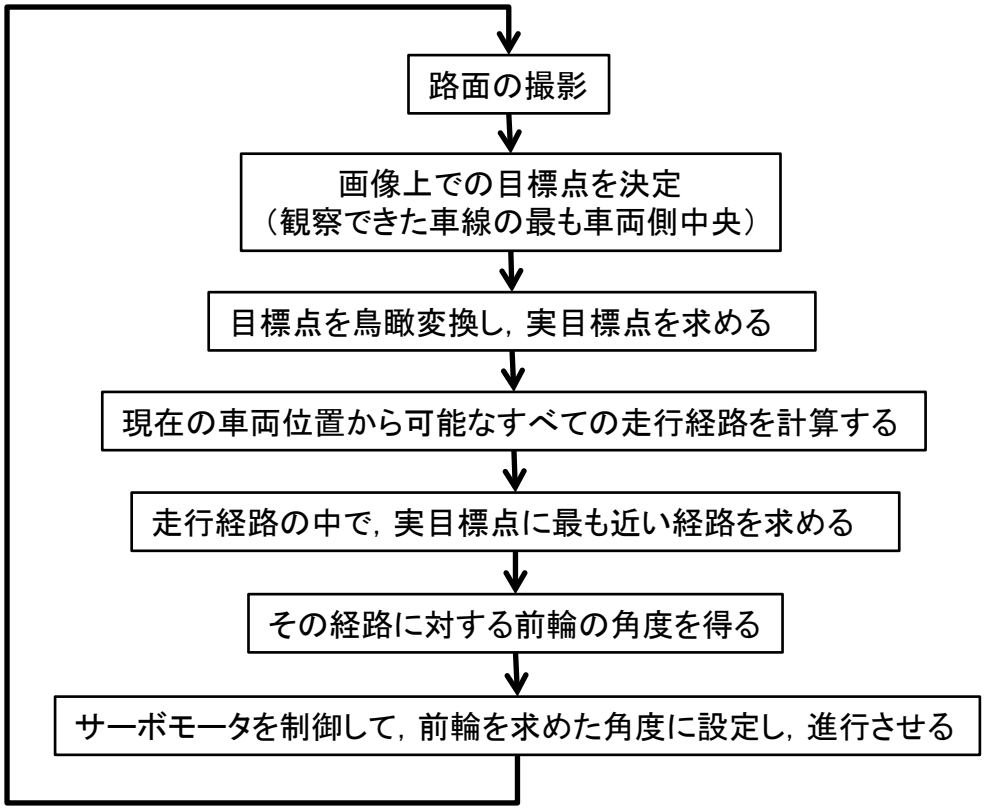


図 9　ステアリング制御のフローチャート

本研究では，路面として大きな白い紙面を使い，その上に描いた黒い実線を車線とした．実際の車線は，レーンとレーンの境界を白い破線で描いたものである．今回の実験では，これを簡単化して，レーン中央を黒い実線で描いた. このような車線を撮影した画像を, 閾値で 2 値化することで, 車線領域とした. 車線領域の中で，最下部が車両に近い側である．従って，画像最下部の車両領域の中央を，画像上での目標点とした.

図 9 のフローチャートの中で，現在の車両位置から可能なすべての経路を計算する部分について詳細に説明する．現在の車両位置で，前輪の角度を θ に設

定した状態を図 10 に示す. 3.2 節で説明したように, この車両は円周上を動く. まず, 円の中心座標を求める. 車体の先端を原点としたときの外側前輪の座標を A, 求める回転中心の座標を B とする. 外側前輪の角度を θ としたとき, A と B を結んでできる直線 AB の傾きは $\tan(\theta)$ (車体より右側の中心座標をもとめる場合は $-\tan(\theta)$) となる. 直線の傾きと通る一点 A がわかったため直線 AB の式を求めることができる. 回転中心は後輪の車軸を水平に伸ばした線上にあるので B の y 座標は既知である. よって直線 AB の式に B の y の値を代入することで回転中心の座標 B を求めることができる.

3.2 節の式(3)から求めた回転半径は回転中心から外側前輪までの距離である. 従って, 上で求めた円の中心と式(3)の回転半径から得る円周は外側前輪が通る軌跡になる. ライントレースを行う場合, 車体の先端中央がラインの上を通過するように走ることが望ましい. そのため車体先端中央が通る軌跡を求める必要がある. これには, 回転半径を AB の長さではなく BO の長さとすればよい. BO の長さは三平方の定理から求めることができる.

本研究で用いる模型自動車の場合, 前輪の角度に対応して, 左右 7 個ずつと直線 1 個の合計 15 個の走行パターンがある. このすべてのパターンに対する回転中心と回転半径を求めることで, 模型自動車がとりうる運動の軌跡をすべて求めることができる.

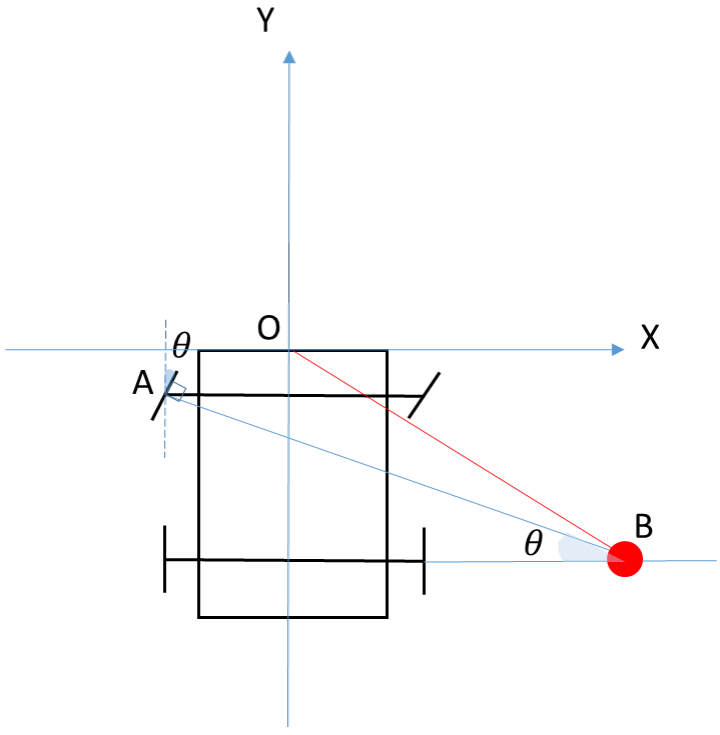


図 10　車体先端を原点とした 2 次元座標平面 XY における
回転中心点 B と回転半径 BO の関係

次に適切なタイヤ角度の選択方法について説明する. 目標点と 15 個全ての走行パターンとの距離を比較する. 目標点を P とすると$|BO - BP|$で目標点と走行パターンとの距離を比較でき, 最も距離が短いものを適切なタイヤの角度とする. ただし, 目標点の位置がタイヤの角度を最大にしたときに得られる円の内側に来てしまうと，曲がりきることができない.

4 章 実験と考察

提案手法を検証するためにライントレースプログラムを作成し，実験した.このプログラムは道路面をカメラで撮影し,カメラから得た情報に従って模型自動車を制御し走行させるものである.道路は白い紙の上に 2cm ほどの幅の黒い線を描いたものである.この車線の上を模型自動車は走行する.プログラムは図 11 の 6 個の処理を繰り返す.

(1) カメラで路面を撮影し画像を取得する.
(2) 画像から車線を検出する.
(3) 車線の画素のうち最も模型自動車に近い一点を「目標点」として設定する.
(4) 「カメラ画像」の目標点に鳥瞰変換を適用し，模型自動車を原点とした 2 次元座標平面の目標点に変換する.
(5) 目標点に向かって進むようにタイヤ角を設定する.
(6) DC モータを動かす.

ここまでの処理を繰り返すことでライントレースを行う.

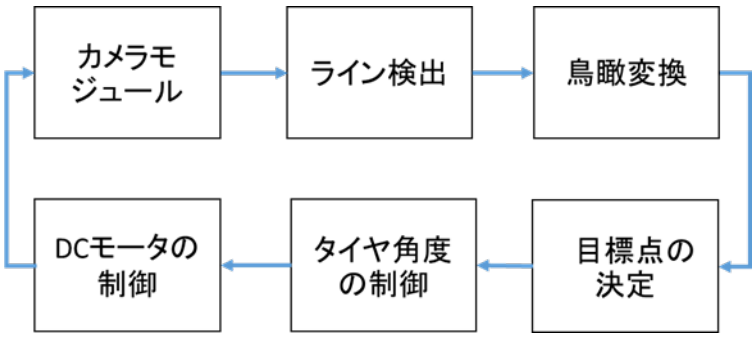


図 11　ライントレースの流れ

4.1　カメラ画像の鳥瞰変換に関する実験

撮影した画像はグレースケール画像である.この画像から車線を抽出するために，画像を 2 値化した.2 値化した結果，車線の画素値は 0,それ以外の画素値は 255 になる.2 値化した画像の水平方向 1 行の画素値を取得し，連続した 0 の画素の中点を求める.これを画像全体に行うことで模型自動車が辿るべき一本のラインを求める.このラインのうち，最も車体に近い一点を目標点とする.この目標点を鳥瞰変換して，車体先端を原点とし路面の実寸に対応した 2 次元座標での目標点の位置を得た.

図 12 での目標点を白いひし形の点で表した.その画像座標の位置は(150,235)である． これを鳥瞰変換すると(-3.071,95.725)になる.実際に，車体先端を原点として定規で計測してみると(-4,91)となった． x 軸方向に約 1mm,y 軸方向に

約 4mm のずれが生じたが，おおむね正しい位置を計算することができた.

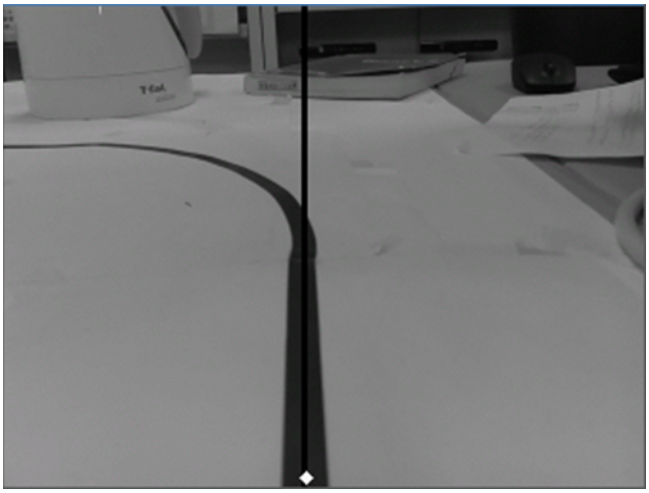


図 12　撮影された路面

4.2　ステアリング角度の制御に関する実験

3.3 節の方法を用いて目標点の位置からタイヤの角度を決定し，DC モータを動かし模型自動車を前進させることで，目標点に向けて動かすことができた.目標点の位置は，画像が読み込まれるたびに変更する可能性がある．変更するたびに適切なタイヤの角度を決定できるので，スムーズにライントレースを行うことができた.4.1 節で確認したように,目標点に数ミリのずれが生じているが,見たところ問題なくライントレースが行えた.

目標点が，タイヤ角度を最大にしたときの円(最小回転円)よりも内側に来てしまった場合，曲がりきることができずに車線からそれてしまう.そこで目標点が最小回転円よりも内側に移動してしまう前に，目標点を最小回転円よりも十分に外側になるまでバック走行を続けるように制御した.そこから再度，前進するように走らせることで，ある程度解決することができた.また,最小回転円より外側に目標点があっても L 字カーブのような切れ角の大きいカーブの場合はバック走行を入れても曲がりきることはできなかった.

4.3　考察

目標点を設定し，その点に向かって走らせることでライントレースを行った.道路面に応じた適切な目標点を決めてしまえば，後はその点に向かって走るだけである．従って，最も重要なのは目標点をどのようにして決めるかであると考えられる.本実験では目標点を常に辿るべきラインのうち，最も車体に近い点と固定していた．しかしそれでは L 字のような切れ角の大きいコースでは曲が

ることができなかった. コースの特徴によって適切な目標点を決めることができれば，ライントレースの精度を上げることができると考えられる.

5 章 結論

撮影した画像上の目標点と自動車の位置関係の問題を鳥瞰変換により解決し,単眼カメラのみで走路の位置や距離などを把握することができた.自動車がカーブするときの走行経路を計算で求め,ステアリングを制御することで任意の場所へ模型自動車を走らせることができた.車線を検出しその上をスムーズに走ることに関しては,車線検出が上手く行えた場合に限りできた.撮影した路面のノイズ除去や車線検出などの処理の精度が低く,路面にゴミや光などのノイズが多いと適切な目標点を求めることができなかった.よって課題としてはノイズ除去や車線検出の処理の精度を高めることがあげられる.今回の実験でステアリング制御を高い精度で行うことができたため, 車線を正確に検出できればライントレースの精度を上げることが可能である.

Raspberry Pi という小型で安価なワンボードコンピュータを搭載した模型自動車で自動運転を行うことで,実際の自動車で実験するよりも模擬的ではあるが安価で容易に実験を行うことができた.よって Raspberry Pi は簡単な実験や製品の試作での利用価値が大きい.

参考文献

[1] 車の数学　http://k-ichikawa.blog.enjoy.jp/etc/HP/js/Car/car.html
[2] 清水宏昭，柳川博彦，「鳥瞰表示による駐車支援システム」，デンソーテクニカルレビュー，Vol.11，No. 1，2006.
[3] Wiring Pi の Web サイト，http://wiringpi.com

謝辞

本論文を作成にあたり,丁寧なご指導を賜りました蚊野浩教授に感謝いたします.

付録

[プログラム名]

raspicam_cv1.c

[内容]

このプログラムはRaspberry Piにつなげたカメラモジュールにより路面を撮影し車線(黒色の線)を検出する.検出した車線から一点,目標点を決定し,この目標点に向かって走るように前輪に取り付けたサーボモータ,後輪に取り付けたDCモータを制御することで車線の上を走行する.