第8回 相磯秀夫杯 FPGAデザインコンテストに係る自動運転車の設計と実装

荒川 麻衣子, 新明 洋平, 弥栄 俊介,
斉藤 大晃, 中村美由香, 加山 愛里子,
小堺 美咲, 蓬田 大貴, 奥山 祐市
チーム 自動四輪部
会津大学

***Abstract*—TrenzElectronics 社による SoC である ZynqBerry を用い、自動運転車の設計と実装を行った。恒常的な処理が必要なモータ制御部を PL でまかなうことにより、計算時間の高速化をはかった。**

I. はじめに

本稿では、演算ユニットとして、TrenzElectronics 社の SoC である ZynqBerry[1] を採用して自動運転システムを開発した。この ZynqBerry には Xilinx ZYNQ-7010 が搭載されている。ZynqBerry はは Raspberry Pi 互換の基板形となっており、Raspberry Pi 向けシールド等の資産の再利用が容易である。
車体は対向二輪型とすることで、ステアリング機構を採用したものと比べて旋回半径を小さくした。実装は、最低限の実装をすることで自動運転システムの機能の検討が出来るようにすることを目的として行われた。また、特に恒常的な処理が必要なモータ制御部 (左右モータのスロットル制御のために用いられる) をプログラマブルロジックで実装した。これによりプロセッシングシステムの計算資源をより複雑な処理、特に PL に実装するには複雑な処理に集中させることが期待できる。

II. 実装

今回は TrenzElectronics 社による SoC である ZynqBerry を用いて処理の中枢部を実装した。実装された自動運転システムのブロック図を図 1 に示す。ZynqBerry はカメラを

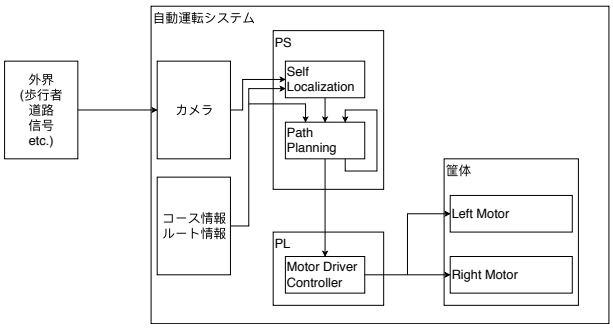

Fig. 1. 実装された自動運転システムのブロック図

通して外界の情報を取得する。その情報を用いて、適切な行動を決定・実行する。PS では、Self Localization と Path Planning が施行される。これらは PS 上で動作する Linux を介す。Self Localization では、カメラの情報とコース・ルート情報を照らし合わせて自己状態 (x 座標 x, y 座標 y, 回転角 ψ) を推定し、Path Planning に渡す。Path Planning では、推定された自己状態とルート情報、それと Path Planning から得られる制御のフィードバックにより車体の特性による運動上の制約を考慮した経路とその経路を追跡するための制御 (速度 v, 操舵角 ψ) を計算する。PL では、モータ制御のための処理が施行される。PS で計算された速度と操舵角を入力し、同速度と操舵角を実現するための電圧を出力するようモータドライバへの命令を発行する。操舵は左右モータの回転速度の差によって制御される。

A. 構成要素

- TrenzElectronics ZynqBerry
- FaBo #605 Motor Driver for Raspberry Pi
  - モータドライバ DRV8830
  - ADC MCP3008
- モバイルバッテリ
- タミヤ ユニバーサルプレート

III. 自己位置推定

取得したカメラ画像を鳥俯瞰図に変換し、テンプレートマッチング [2] を用いて自己位置を推定する。今回は自己位置として (x,y,ψ) を出力する。x,y はコース上の座標、ψ は車体の角度である。

取得したカメラ画像にはコース全体の一部分の情報が含まれている。それをテンプレートとしてコース全体図とマッチングをする際、コース全体図は俯瞰図であるため、カメラ画像も俯瞰図に変換する必要がある。次に変換した画像からコース部分を切り抜き、それをテンプレートとしてテンプレートマッチングをする。テンプレートを 1 画素ずつ走査し、各類似度を計算する。類似度の計算には正規化相互相関を用いた。正規化相互相関は画像をベクトルとみなして内積を計算するため、値がベクトルの長さに対して影響を受けず、照明の影響を受けにくくなる。正規化相互相関を用いた類似度は次の式で求められる。

$$NCC(d_x, d_y) = \frac{\sum_{x,y}[I(d_x+x, d_y+y)T(x,y)]}{\sqrt{\sum_{x,y}[I(d_x+x, d_y+y)]^2}\sqrt{\sum_{x,y}[T(x,y)]^2}} \quad (1)$$

このとき $I(x,y)$ は座標 (x,y) におけるコース全体図の輝度値、$T(x,y)$ は座標 (x,y) におけるテンプレートの輝度値である。

テンプレートマッチングではテンプレートと全体画像の一致する領域が同じサイズである必要があるが、実際に入力されるコースの部分画像は必ずしも同じ大きさ、角度ではないため、調節する必要がある。サイズ、角度を任意の範囲で与え、そのサイズと角度ごとにテンプレートマッチングを行い、最も類似度が高くなるサイズと角度を採用する。NCC の値は-1.0 から 1.0 の範囲に収まる。最大値の 1.0 に最も近くなった走査位置が、テンプレート画像に最も類似する部分の左上座標となる。
しかしコース全体図の中にテンプレートと類似する箇所が複数あり、本来の箇所ではない箇所の類似度が高くなることがある。それを防ぐために、以前の自己位置から一定の範囲内にある類似度の高い点を自己位置を推定する。誤差を少なくするために、テンプレートマッチングによる推定と移動量による自己位置推定の結果を使う。2 つの推定位置の中点を自己位置と決定する。

$$(x, y, \psi) = max(TemplateMatching(x\prime, y\prime, \psi\prime, scale\prime)) \quad (2)$$

ただし $(x\prime, y\prime) \in R$ and $R : (x\prime - cx)^2 + (y\prime - cy)^2 < r$

IV. Path Planning について

Path Planning のパートは、PS（で動作する Linux) 上で動作する。まず、前段から自己状態 $\vec{x}$、大域地図、視点から終点までの経路を入力し、ゴールに到達するために現時点で最適であると決定されたスロットル v、ステアリング角 ψ を出力する。

$$f : \{\vec{x}, M, R\} \longmapsto \{v, \psi\} \quad (3)$$

最適な制御を算出するための処理 (この f) は二段階に分かれる。

$$f = g \circ h \quad (4)$$

$$g \quad : \quad \{\vec{x}, M, R\} \longmapsto \{\vec{x}, M, R, t_{tile}\} \quad (5)$$

$$h \quad : \quad \{\vec{x}, M, R, t_{tile}\} \longmapsto \{v, \psi\} \quad (6)$$

このアルゴリズムでは、コース情報を道路の構成要素（タイルと呼ぶことにする）で区切って考える。まず、g では、現在位置から、「現在位置を含むタイルの出口 t_{tile}」を計算する。次に h で、現在位置から「現在位置を含むタイルの出口 t_{tile}」に到達するための経路を計算する。
g はまず $\vec{x}$ から「現在位置を含むタイルの種類」を、$\vec{x}$ と R から「現在位置を含むタイルの出口を含むタイルの辺」を算出する。そのあと、「現在位置を含むタイルの種類」と「現在位置を含むタイルの出口を含むタイルの辺」から t_{tile} を計算する。t_{tile} は「出口を含む辺の左（走行）車線中央」と定義した。
例を図 2 に示す。三角が現在の車の状態である。経路がこの丁字路を右折することになっているとき、タイルの出口は「出口を含む辺の左車線中央」である丸印が表す座標になる。ここが t_{tile} となる。
h は g の出力を用いて v, ψ を計算する。これのために、Dynamic Window Approach (DWA)[3] を採用した。DWA は dynamic window と呼ばれる採用可能な $v - \psi$ 空間を定義し、また、目的地に到達するための評価関数を定義する。この二つをもちいて、制御の候補を算出し、その中から評価関数を最大にするようなものを選び、その制御を出力する。
今回、DWA の仕様は以下のようにした。事前に道路の種類と進行方向ごとに「不可侵域」を定義しておき、それを DWA における「障害物」とした。評価関数は車体の向き、速度、障害物からの距離できまるものとした。

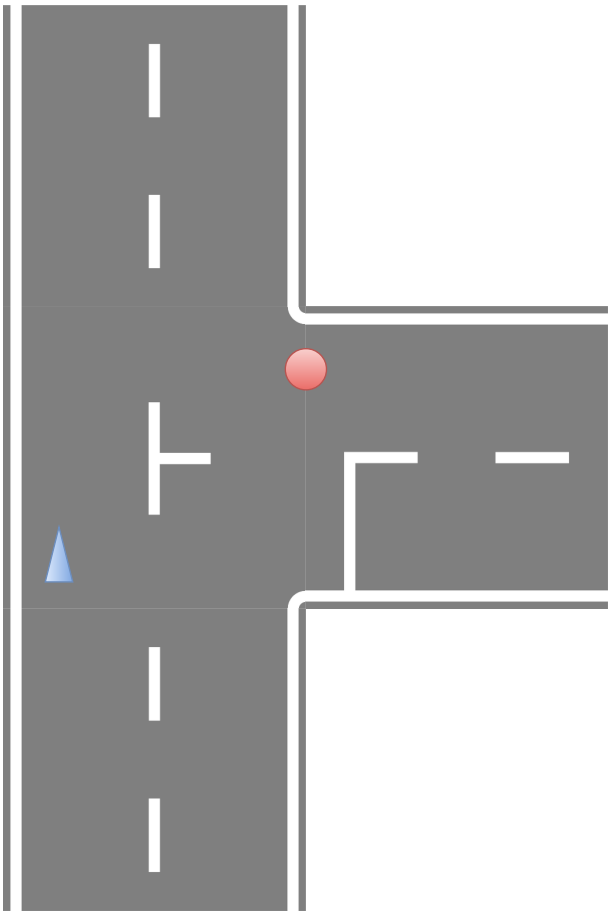

Fig. 2. タイルの出口

V. モータ制御について

モータ制御は車が走る限り常に動作していなければならないので、PL と外部回路による実装を行った。今回は車が対向二輪型であるので、左右のモータを別に制御する必要がある。
コミュニケーションは、図 3 のように、linux から AXI バスを用いてリファレンス速度とステアリング角を入力し、出力電圧の値を外部回路であるモータドライバに I2C で出力する。モータドライバは左用と右用の 2 つある。

VI. 今後の課題

今後は歩行者と信号の認識に対応していく必要がある。その際、計算量の増加とそれに伴う最大車速の低下が予想されるため、アルゴリズムの幾許かを PL へ移植する。

References

[1] TrenzElectronic. Zynqberry について — trenz 社製品販売サイト. [Accessed: 03- Sep- 2018]. [Online]. Available: http://www.trenz.jp/aboutzynqberry.html

[2] opencv dev team. Template matching — opencv 2.4.13.7 documentation. [Accessed: 06- Sep- 2018]. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[3] S. T. D. Fox, W. Burgard, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, pp. 23 – 33, 1997.

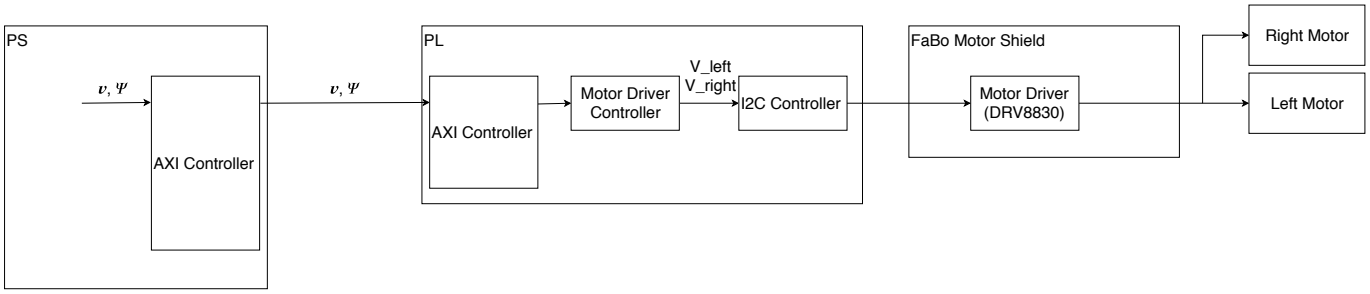

Fig. 3. モータ制御のブロック図