ちびチャレ2019成果報告

B 班 (深津蓮，島田航太，吉内航，土屋一朗)

1　緒言

近年，自動運転車を中心とした自律移動ロボットの研究開発が盛んに行われている．これらのロボットが自律移動を行うためには，自己位置推定，大域経路計画，局所経路計画，環境認識などが非常に重要である．これらの技術は ROS の navigation stack[1] などのオープンソースソフトウェアを用いて容易に実装可能である．しかし，今回はオープンソースソフトウェアを使用せずにそれらの技術を実装することで，コーティング技術と共に基礎的なロボット技術を学ぶことを目的とする．ハードウェアとしては，iRobot 社の Roomba[2] と北陽電機社の 2D-LiDAR の UTM-30LX[3]，Raspberry PI 財団の Raspberry PI Model B+[4]，logicool 社の C270 HD WEBCAM[5]，ノート PC を用いる．今回は，障害物を避け，白線を検知したら一回転し，最終的に明治大学生田キャンパス第二校舎 D 館 1 階を 1 周することを課題とする．

2　提案手法

本章では，提案システムにおける自己位置推定，大域経路計画，局所経路計画，白線検知について述べる．自己位置推定及び大域経路計画で用いる地図については，オープンソースソフトウェアの gmapping[?] を用いて作成した．システム図を Fig. 1 に示す．

2.1　自己位置推定

自己位置推定には Augmented Monte Carlo Localization(AMCL)[6] を用いた．観測モデルのノイズとして，正しい計測時の局所的な計測ノイズ，事前地図の中に存在しない物体による計測ノイズ，レーザがガラスなどを透過したことによる計測ノイズ，ランダムに発生する計測ノイズの 4 種類の観測ノイズを想定した．これらを導入することで，本試験中に想定されるノイズに対して強い尤度計算ができるようにした．また，Eq.1 に示す全パーティクルの平均に比例する確率 p でランダムパーティクルを追加し，位置推定誤差からの復帰ができるようにした．ここで，ω_{fast}，ω_{slow} はそれぞれ尤度の短期および長期の平均値である．これと併用してパーティクルの xy 座標および yaw 角の分散が閾値以下になると，最新の推定位置の周りにパーティクルを初期化することにより，システムのロバスト性を高めた．

$$p = max(0.0, 1 - \frac{\omega_{fast}}{\omega_{slow}}) \tag{1}$$

2.2　大域経路計画

大域経路計画には，A*アルゴリズムを用いた．この経路計画において，地図はセルの集合として扱う．開始地点のセルから経路探索を始め，そのセルに到達するための最小コスト g と目標地点からの距離のヒューリスティック値 h の和が最小のセルを選択して探索を進めていく．これにより，全経路を探索することなく最短経路を導き出すことができる．しかし，このままでは経路が壁に沿って壁と接してしまう．これを改善するために，h 値と g 値に壁からの距離に基づく w 値を足した値を f とすることで壁から少し離れた経路を選択するようにした．最終的な f の値を Eq.2 に示す．

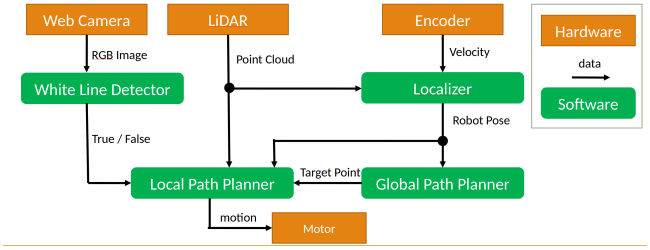

Fig. 1: システム図

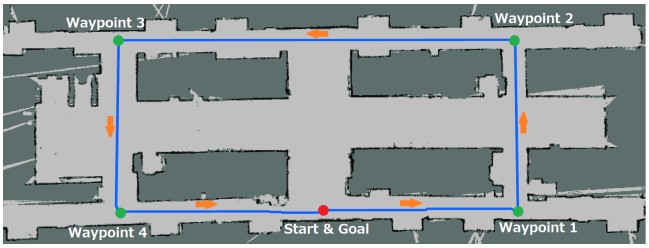

Fig. 2: 大域経路

$$f = g + h + w \tag{2}$$

Fig.2 に導き出した経路を示した．この経路は，機体の位置から次の目標地点を定めるときに用いた．自己位置から一番近い大域経路の点を調べ，その点から経路上で 1.5m 先の点を目標地点と定めるという方法で行った．

2.3　局所経路計画

局所経路計画には Dynamic Window Approach を用いる．Dynamic Window 内の速度 v, 角速度 ω をサンプリングし経路を予測する．そして，コストを計算し，コストが最少となる速度 v , 角速度 ω を出力値とする．ここで，コスト計算には予測経路と障害物までの最短距離である $obs(v,\omega)$，最終予測位置から目標地点までの距離である $dis(v,\omega)$，目標位置に対する機体の向きである $heading(v,\omega)$，最高速度との差である $vel(v,\omega)$ を用いる．特に，ゴールへの評価は距離と向きの二つを用いる．Fig.3 に局所経路計画の評価のイメージを示す．Eq.3 になんとかを示す．

$$\begin{aligned} G(v,\omega) = \alpha obs(v,\omega) + \beta dis(v,\omega) \\ + \gamma heading(v,\omega) + \delta vel(v,\omega) \end{aligned} \tag{3}$$

2.4　白線検知

白線検知にはオープンソースソフトウェアの OpenCV[7] を用いて作成した．WEBCAM から取得し

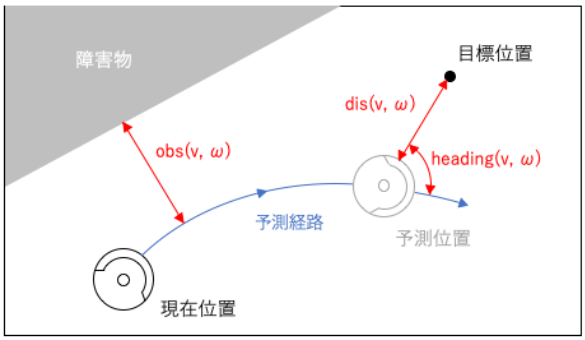

Fig. 3: 局所経路計画における評価

た RBG 画像を，グレー化，ぼかし，二値化，境界線の取得，面積によるフィルタリング，形状判断を行い，白線を検出した．以下に詳しく記述する．

まず，WEBCAM から RBG 画像を取得し，Eq.4 を用いてグレースケール画像を作成した．

$$gray = Red*0.299+Green*0.587+Blue*0.114 \quad (4)$$

そのグレースケール画像に対してガウシアンフィルタをかけ，大津の二値化により白と黒の二値化画像を作成した．大津の二値化とは，グレースケール画像の画素値のヒストグラムが双峰性を持つと仮定して，その２つのピークの間の値を閾値とする手法である．その二値化した画像に対して，白と黒の境界線に沿った連続する点をつなげることで，輪郭を検出した．今回の手法では黒い背景に白い物体がある想定で物体の輪郭を検出した．その後，その物体の輪郭から画像上の面積を取得して，大きすぎるものと小さすぎるものは除く処理を行った．最後に，取得した輪郭を矩形で囲み，輪郭の面積 $S_{contours}$ と矩形の面積 $S_{rectangle}$ が Eq.5 を満たすものだけに絞ることで四角形の判断を行い，白線を検出した．

$$S_{contours} > S_{rectangle} * 0.8 \quad (5)$$

参考文献

1) http://wiki.ros.org/navigation

2) https://www.irobot-jp.com

3) https://www.hokuyo-aut.co.jp/search/single.php?serial=21

4) https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

5) https://www.logicool.co.jp/ja-jp/product/hd-webcam-c270

6) Sebastian Thrun , Wolfram Burgard , and Dieter Fox 著, 上田隆 一 訳, 確率ロボティクス, 2007

7) https://opencv.org