ちびチャレ 2018 成果報告

2 班 (高橋一貴，尾崎亮太，永井良昂，鈴木パスカル賢太)

1　緒言

近年，人間の仕事を代替したり，人間の身近な環境で用いられたりする自律移動ロボットの研究が盛んに行われている．これらのロボットには，地図上の指定された地点間において，地図上での自身の姿勢や位置を推定すること (自己位置推定)，障害物を避けつつ最短距離になるような経路を計画すること (大域経路計画)，自身の運動モデルを考慮しつつ障害物を回避しながら走行すること (局所経路計画) が求められる．このような機能を実現するためには，ROS の navigation stack[1] 等のオープンソースソフトウェア (OSS) を利用することもできるが，ただそれらの使い方を学ぶだけでは，その理論などを深く理解することは難しい．しかし，それらの基本的な理論の理解は，今後の卒業研究などにおいて不可欠なものである．したがって，本年度のちびチャレ (以下，ちびチャレ 2018) では，これらの実装を通して自律移動ロボットに関する技術の基礎を学ぶことを目的とし，大域経路計画，局所経路計画，自己位置推定を主な構成要素としたナビゲーションシステムを作成する．ちびチャレ 2018 においては，D 館 1 階を，障害物を避けながら 1 周できるロボットを制作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大域経路計画，局所経路計画，自己位置推定について述べる．大域経路計画及び自己位置推定で用いる地図については，OSS の gmapping[2] を用いて作成した．システム図を Fig. 1 に示す．

2.1　Localization

自己位置推定には Augmented Monte Carlo Localization (AMCL)[3] を用いる．パーティクルの初期配置は指定したスタート地点を中心とした 2 次元のガウス分布に従う．i 番目のパーティクルの尤度 w_i は，実スキャンおよび，各パーティクルの位置と地図データから求めた参照スキャンの残差平方和 R_i を用いて以下の式により求める．

$$w_i = exp\left(-\frac{R_i}{2\sigma^2}\right) \tag{1}$$

σ^2 は正規分布の分散を表す定数であり，この値を大きくするほどパーティクルの分布が広がることになる．値は動作試験を行って決定し，3.0 とした．推定自己位置としては，最も尤度の高いパーティクルを採用する．また，正規化前の全パーティクルの尤度の平均値に応じた割合でランダムパーティクルを挿入する．ランダムパーティクルの割合は，尤度が短時間に大きく減少したとき，多くなる．リサンプリングと推定自己位置のアップデートは，オドメトリをもとに，前回のリサンプリングから一定距離以上並進移動するか，一定角度以上旋回した時に行う．

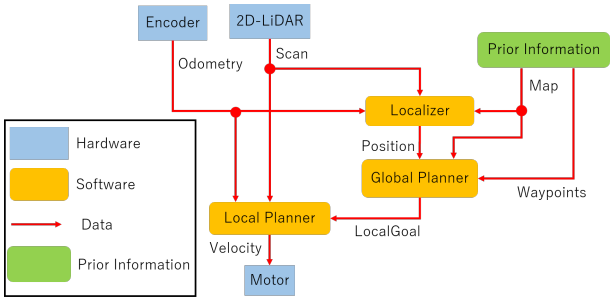

Fig. 1: System architecture

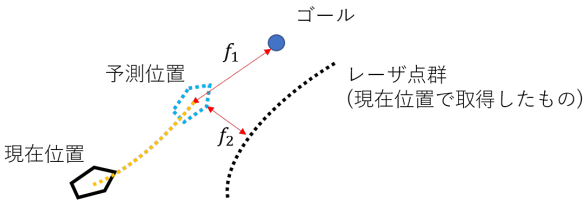

Fig. 2: Evaluation of local path

2.2　Global Path Planning

大域経路計画には A*アルゴリズム [4] を用いる．経由点を複数与えると，それらを順に辿るような経路を生成する．壁からある程度の距離を保った経路を生成するため，地図上にコストを設定する．地図上の各点におけるコストは，壁からの距離 d を用いて

$$cost = maxcost\left(1 - \frac{d}{D}\right) \tag{2}$$

とする．ここで，D はコストを設定する距離を表し，$D \geq d$ である．未探索領域は壁として扱い，すべて壁と同じコスト (最大値) となる．

A*の計算において，ある位置からの移動方向は地図上で縦，横，斜めの 8 方向であり，移動コストはどの方向についても同じとする．また，ヒューリスティック値は，その座標とゴール座標の x 方向と y 方向の距離 dx，dy を用いて

$$heuristic = \sqrt{dx^2 + dy^2} \tag{3}$$

とする．

提案システムにおいては，Fig. 1 に示すように，局所経路計画には大域経路そのものではなく 1 つの目標地点を用いる．この目標地点は，大域経路上の点の内，推定自己位置から l(実験においては 3.0[m] とした) だけ離れた点としており，自己位置が更新されるたびに目標地点も更新される．目標地点と推定自己位置の距離の計算は次の経由点から徐々に近づける形で行う．また，目標地点が経由点と一致した場合は，その次の経由点が存在する場合，経由点と推定自己位置の距離が $l/2$ を下回った時点でその経由点を通過済みとし，次の経由点を基準に目標地点を計算する．

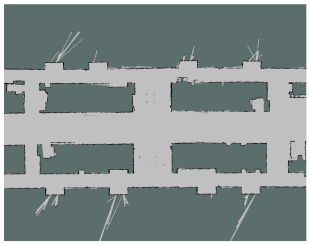

Fig. 3: Map

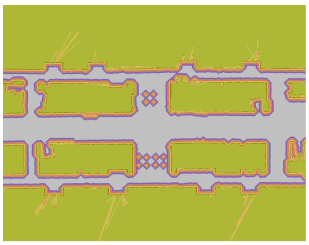

Fig. 4: Cost map

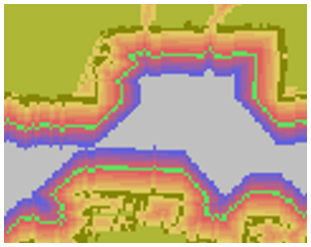

Fig. 5: Expanded cost map

2.3 Local Path Planning

局所経路計画には Dynamic Window Approach (DWA)[5] を用いる．Dynamic Window 内の速度 v，角速度 ω について，以下の評価関数を用いて評価値を算出する．

$$value(v,\omega) = \alpha \times f_1(v,\omega) + \beta \times f_2(v,\omega) \quad (4)$$

ここで，シミュレーション時間 t(実験においては 2.0[s] とした) について，f_1 は t 秒後の予測位置と目標地点との距離を評価し，f_2 は t 秒後の予測位置と障害物の距離の最小値を評価する．α と β は係数である．$f_2 \leq 0$ であるときは障害物との衝突が予測されるため，その動作は選択しない．Fig. 2 に局所経路の評価のイメージを示す．

現在位置と目標地点の距離が L(実験においては 0.1[m] とした) 以内になると，並進移動を停止する．更に，現在の角度と目標角度の誤差が θ(実験においては 0.1[rad] とした) 以内になると旋回を停止する．これにより，最終的なゴール地点で停止することができる．

3 実験

ちびチャレ 2018 の課題である D 館 1 階を 1 周させる実験を行った．

ハードウェアとしては，車体として iRobot 社の Roomba[6]，センサとして北陽電機社製 2D-LiDAR の UTM-30LX[7] を用いる．また，ノート PC を 1 台搭載し，この PC 上でナビゲーションシステムを動作させる．

Fig. 3 に実験に使用した D 館 1 階の地図を示し，Fig. 4 に式 (2) において $D = 0.6$[m] としたときのコストマップを示し，Fig. 5 にその一部を拡大したものを示す．ここで，コストは黄，赤，青の順で大きい．また，Fig. 6 に 1 周分の経路と経由点を示す．スタート地点から 1 → 2 → 3 → 4 →ゴール地点の順に走行させた．

経路付近に障害物 (底面が 1 辺 210mm の正方形からなる角柱のダンボール箱) を配置した．ロボットの半径は 225mm であると設定した．自己位置推定のパーティクル数は 1000 個とした．

実験走行は 10 回行い，ロボットの初期配置や障害物の個数及び位置は毎回変えた．

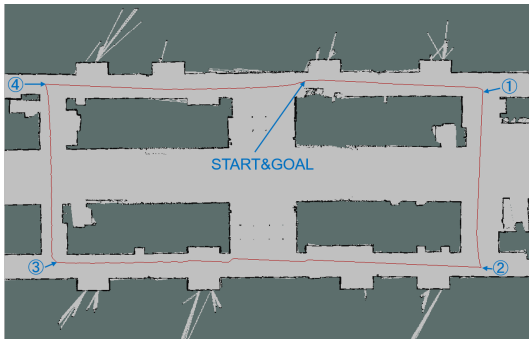

Fig. 6: Path and waypoints

Table 1: Result of experiment

回数	完走	条件	備考
1	○	障害物なし	-
2	○	障害物 1 つ	-
3	○	経路上に 3 つ 1m 間隔で障害物を配置	-
4	○	700mm 間隔で経路に垂直に配置	-
5	○	600mm 間隔で経路に垂直に配置	-
6	○	550mm 間隔で経路に垂直に配置	-
7	○	500mm 間隔で経路に垂直に配置	-
8	×	450mm 間隔で経路に垂直に配置	障害物の前で回転し続けた
9	△	1 辺 500mm の正三角形状に配置	ケーブルが接触するが完走
10	○	1 辺 550mm の正三角形状に配置	-

4 結果

実験結果を Table1 に示す．実験より, 提案システムについて次のような性質を確認できた

- 障害物を回避するための十分なスペースが有れば，障害物を回避して走行できる．しかし，ケーブルがロボットの半径として設定した 225mm の範囲から 20mm ほど出ていたため，条件 9 や 10 のように狭所で左右に旋回する場合，ケーブルが障害物に接触してしまうことがあった．

- 障害物付近でランダムパーティクルの増加が見られたが，条件 8 以外では障害物を通り過ぎると収束し，走行を続けることができていた．条件 8 においては，至近距離にある障害物のためスキャンマッチングがうまくいかないことと，その場旋回中の動作ノイズによってパーティクルが徐々に移動することにより，自己位置推定が破綻していた．

5 結言

簡単な構成のロボットを用いて，局所経路計画，大域経路計画，自己位置推定などからなるナビゲーションシステムを実装した．また，課題を通して提案システムの有用性を確認した．

参考文献

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

2) http://wiki.ros.org/gmapping

3) Sebastian Thrun, Wolfram Burgard, and Dieter Fox 著，上田隆一 訳，確率ロボティクス，2007

4) http://myenigma.hatenablog.com/entry/20140503/1399080847

5) Dieter Fox, Wolfman Burgard, and Sebastian Thrun, The Dynamic Window Approach to Collision Avoidance, IEEE Robotics & Automation Magazine, pp.23-33, 1997

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

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