C-022

速度制約および閉塞区間の制約条件を付加した自律移動ロボットの経路計画法の改良
Improved path planning method for autonomous mobile robots with speed constraints and blocked section constraints

佐藤 隆世[1)]　林 誠治[1)]
Ryuse Sato　Seiji Hayashi

1　はじめに

近年，人手不足や新型コロナウイルスの影響で工場，倉庫，ファミリーレストランをはじめとする様々な場所で自律移動ロボットが活躍している．これらで活躍するロボットには高い安全性と効率よく目的地まで移動できることが求められているが，実際に活動する環境下では多くの危険が潜んでいる．ロボットは，あらかじめ生成された経路に従って走行するが，人などの障害物があった場合にはロボット自身が走行可能か，速度を落とすかどうかなどを直前に判断している．また，生成される経路はそのロボットが最大速度で到達できる最短のものとなっており，障害物を避ける場合などにその経路から逸脱してしまうことがある．そこで，あらかじめそのような領域を考慮したうえで経路計画を行えば，移動距離が環境地図上の最短距離でなくても，より効率よく安全に目的地まで到達できる可能性がある．

例えば，ファミリーレストランのような場所で稼働している配膳ロボットは，店内の人ごみなどによる影響を受けやすい．人ごみを避けるために一時停止したり，迂回することで，効率よく目的地までたどり着くことが難しい．あらかじめそのような場所やその付近では，速度制約区間や一時停止位置，進入禁止領域として事前に環境地図上で定義することで，ロボット自身で障害物回避を行う頻度を減らし，また，誰も着席していない客席周辺を走行する場合は最大速度で走行できるように定義することにより，効率よく目的地までたどり着けるのではないかと考える．

本研究では既存のナビゲーションシステム環境 [1, 2] において，速度制約区間の定義とともに，周囲の状況やロボットの走行経路によって閉塞区間などの制約条件を付加することで，より安全に効率よくロボットの自律走行を補助可能な経路計画法の改良を目的とする．

2　経路計画アルゴリズムの改良

本研究を進めるにあたり，生成される最短経路を確認するためのシミュレーション環境を Python 3.9.0 と Python のグラフ描画ライブラリである matplotlib v3.7.3 で開発した．経路生成に使用するのは既存のナビゲーションシステムでも用いられている A*アルゴリズムを使用する．

2.1　A*アルゴリズム

一般的なナビゲーションシステムでは A*アルゴリズムがよく用いられている．広大な環境では経路探索に時間がかかってしまうことが予想されるため，ダイクストラ法ではなく，少ない計算量で最短経路を求めやすい A*アルゴリズムを使用している．A*アルゴリズムで最短経路を求める際には，式 (1) に示すコスト関数 $f(n)$ を用いて行う．ここで，$g(n)$ は式 (2) のマンハッタン距離を用いてスタート地点から現在地までの移動距離を表し，$h(n)$ は式 (3) のユークリッド距離を用いて現在地からゴールまでの最短距離の予測値をそれぞれ表す．また，n は自然数とし，n 時点での座標をそれぞれ (x_n, y_n) とする．(x_{start}, y_{start}) はスタート地点，(x_{goal}, y_{goal}) はゴール地点の座標である．

$$f(n) = g(n) + h(n) \tag{1}$$

$$g(n) = |x_n - x_{start}| + |y_n - y_{start}| \tag{2}$$

$$h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2} \tag{3}$$

スタート地点から出発し，ロボットの現在地に隣り合う上下左右 4 地点のコストを計算してその中でコストが最小となる地点にロボットを移動させる操作を，ゴール地点にたどり着くまで繰り返すことで，最短経路を求めることが可能である．

2.2　シミュレーション環境

実際のナビゲーションシステム上で使用する経路計画アルゴリズムが適切であるかどうかを確認するためのものであり，本研究では以下の機能を実装している．

- 環境地図データの読み込み
- 環境地図データの表示
- 環境地図データ上でスタート地点とゴール地点の表示
- 速度制約区間や立ち入り禁止区域，一時停止位置を考慮した最短経路の表示
- ロボットの動作の可視化

2.3　確認に使用した環境地図

検証に使用した環境地図を図 1 に示す．この環境地図は小さなレストランを模擬したものである．薄い灰色の区域が速度制約区間，濃い灰色の区域が椅子などが設置されたロボットの侵入禁止区域，黒い区域がテーブルや壁などの障害物を示している．レジやドリンクバー周辺など人の混雑が予想されるような場所や，狭い通路には予め速度制約区間として指定している．

この環境地図データは Portable Gray Map と呼ばれるモノクロ画像を格納する形式のファイルで作成され，8bit のグレースケールで表される．このデータはペイントソフトなどのツールでの作成が可能であり，ROS(Robot Operating System) で動作する map_server(生成された環境地図の管理を行うサーバ) で保存されたデータも使用することが可能である．本研究では，環境地図データの 1

1)　拓殖大学大学院 工学研究科
Graduate School of Engineering, Takushoku University

画素は実寸大で 70cm 四方の領域に相当するものとしている．

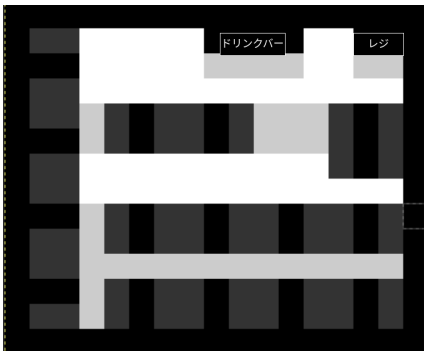

図 1　検証に使用した環境地図データ

3　シミュレーション環境上での最短経路の生成

作成したシミュレータ上に図 1 の環境地図データを読み込み，A*アルゴリズムで最短経路を求めた場合と制約条件を付加した場合で，生成される経路にどのような違いがあるのかを試した．その結果を以下の 3.1 節，3.2 節に示す．

3.1　ナビゲーションシステムにおける既存の経路計画法

式 (1)~(3) に基づいて A*アルゴリズムにより生成された最短経路を図 2 に示す．

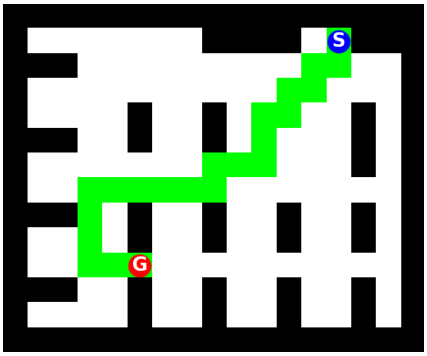

図 2　A*アルゴリズムを用いて求めた最短経路の例

制約条件を考慮していないため，最も最短な経路が求められていることがわかる．

3.2　本研究で改良した経路計画法

3.1 節で実装している A*アルゴリズムの式 (2) を改良することにより，速度制約区間や一時停止位置などを考慮した最短経路を求める．本研究で新たに実装するコスト関数を式 (4) に示す．

$$g(n) = \frac{|x_n - x_{start}| + |y_n - y_{start}| + s(n)}{v(n)} \qquad (4)$$

ここで，$s(n)$ は一時停止位置のコストを表しており，予めフィールド上に設定された 0~9 までの整数である．数字が高ければ高いほど長時間停車する必要があることを示す．したがって，一時停止位置として設定されていない区間では $s(n) = 0$ として設定する．一方，$v(n)$ は速度制約区間のコストを表しており，速度制約区間を設定する際に 0.1~1.0 まで 0.1 刻みで予め設定する．図 2 と同じ環境で $s(n)$ および $v(n)$ を適切な値として経験的に設定した後，式 (4) を用いて環境地図上に生成された最短経路を図 3 に示す．

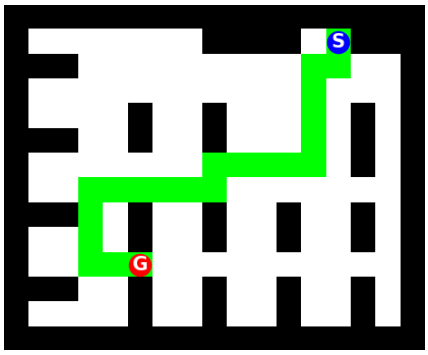

図 3　本研究で改良した A*アルゴリズムを用いて求めた最短経路の例

式 (2) および式 (4) を使用した場合のコストの見積比較を図 4 に示す．式 (4) を使用した場合，速度制約区間をなるべく垂直方向あるいは水平方向に通過することで，速度制約区間でかかるコストを最小限に抑えていることが確認できた．

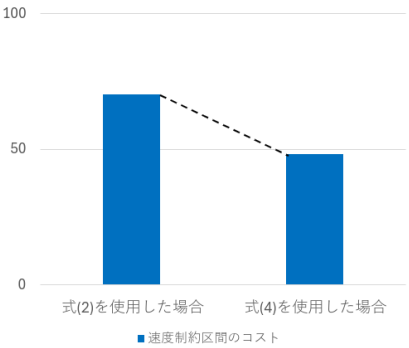

図 4　コストの見積比較

4　まとめと今後の予定

本研究では既存のナビゲーションシステムで用いられている A*アルゴリズムについて，速度制約区間と一時停止位置を考慮してコスト関数を改良し，簡易的なシミュレータ上でその有効性を確認した．予め混雑が予想される箇所を考慮して障害物の回避頻度をなるべく低減させることで，安全かつ効率よく自律移動ロボットを運用できると考えられる．今後はマルチエージェントシステムをシミュレーション環境上に導入する．例えば，人などの動的な障害物を考慮した場合に，本研究で改良したアルゴリズムがどの程度効率よくロボットを運用可能かどうかを検証する予定である．

参考文献

[1] Navigation - TurtleBot3
https://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/
[2] NavigationStack - global_planner
https://robo-marc.github.io/navigation_documents/global_planner.html
[3] Matplotlib 3.7.3 documentation
https://matplotlib.org/3.7.3/index.html