時空間ルーティングを用いた 複数自律移動ロボットの協調走行

福島 悠人[1] 浅井 悠佑[1] 浦野 健太[1] 青木 俊介[3] 米澤 拓郎[1] 河口 信夫[1,2]

概要：本研究では, クラウドサーバを用いた経路共有による, 複数の異なる自律移動ロボットのデッドロックを回避する協調経路計画システムを提案する. ここ数十年で自律移動ロボットの実用化が進み, 自律移動ロボットは決められた道を走行するだけでなく, 動的に変化する環境で柔軟に走行できるようになってきている. しかしながら, 自律移動ロボットの経路計画手法は, 狭い空間での将来のロボット同士のデッドロックや, 密集を想定できないという課題がある. 本研究では, 複数の自律移動ロボットが任意の時刻, 場所で目的地が与えられる環境を想定し, クラウドサーバを用いた経路計画により, 停止や迂回をすることで先に計算された経路を妨げないロボットの協調の実現を目指す. 最後に, 提案する経路計画アルゴリズムをクラウドサーバシステム上で実装し, シミュレーションされたロボットをもちいて本手法の判断の最適度, 処理能力の限界値を評価し, 実世界での実現可能性について考察する.

Temporal Space Path Planning for Multiple Autonomous Mobile Robots Collaboration

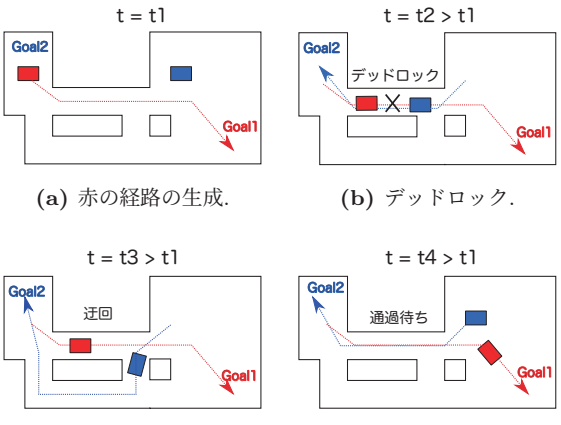
YUTO FUKUSHIMA[1] YUSUKE ASAI[1] KENTA URANO[1] SHUNSUKE AOKI[3]
TAKURO YONEZAWA[1] NOBUO KAWAGUCHI[1,2]

1. はじめに

ここ 10 年の間に, 配送, 配膳, 掃除, 警備など, さまざまな分野での移動ロボットの実用化が急速に進んできている [1]. ロボットは決められたレーンを走るものから, 自律的に移動するものへと変化しており, 人や異種ロボットと共存しつつ, 安全に移動することが求められている.

しかしながら, 複数の移動ロボットが独立して目的地を持って移動するような状況では, そのロボットの経路計画に課題が生じる. 一般的に, 移動ロボットは LiDAR やカメラなどのセンサより, 外界を知覚し自己位置推定を行いながらフィードバック制御を行うため, 現在地から離れた場所にいる別のロボットの存在を知覚することが難しい. そのため, ロボットが 1 台しか通れないような狭い通路では, 2 台のロボットがそれぞれ反対側から侵入し, 自力で回避ができなくなる FRP(Freezing Robot Problem) が発生してしまうという問題がある. 以降, 本論文ではこの FRP が発生することをデッドロック状態とする.

本研究の目的は, 複数の移動ロボットに独立して目的地



(a) 赤の経路の生成. (b) デッドロック. (c) 迂回による回避. (d) 待機による回避.

図 1: 狭い通路におけるデッドロックとその回避の例.

1 名古屋大学大学院 工学研究科
Graduate School of Engineering Nagoya University
2 名古屋大学 未来社会創造機構
Institute of Innovation for Future Society
3 国立情報学研究所
National Instute of Informatics

が与えられる状況において, 狭い通路におけるロボット同士の接近やデッドロック状態を回避し, 最適な経路を生成する経路計画を行うことである. そのために, クラウドサーバを用いて順番に計算されたロボットの経路を共有し, 停止を含めた 3 次元空間上の経路探索を行うことで, 他のロボットの経路を邪魔しない準最適な経路を生成する.

移動する障害物が存在する環境で, 時間を考慮した経路探索を行う手法は自動車や歩行者のナビゲーションの分野においても多く用いられている. しかし, 道路ネットワーク情報やレーン, 交差点などの走行可能な範囲やルールが決められていない屋内の移動ロボットの走行においては, 経路の情報は LiDAR によって得られる点群より生成された地図のみによって得ることが多い. そのため, 屋内移動ロボットの経路探索は広域な走行可能範囲からグリッドベースの探索を行うことが主であり, 同様の手法をそのまま適用することは難しい. 本研究では走行範囲に制約のない複数の移動ロボットの協調走行に注目している.

本手法の概要を図 1 に示す. 図 1a は, ロボットがすれ違えない狭い通路がある空間において, 赤の移動ロボットの経路が計算された状態を表している. ここで, 青のロボットに目的地が与えられたとき, 最短経路を生成するだけの手法では, 狭い通路で赤のロボットと向かい合うことでお互いに回避がとれないデッドロック状態になる (図 1b). 青のロボットがデッドロックを回避する方法は二つある. 一つは図 1c に示した迂回した最短経路を通過する方法である. もう一方は, 図 1d に示した赤のロボットの通過を待機したのち, 最短経路を通過する方法である. 本研究において提案する経路計画手法は, 先に計算された赤色のロボットの経路を用いて, 時間を考慮した経路計算を行うことで, 最短時間で目的地まで到達できる経路を生成するものである. 具体的には, ヒューリスティック関数を用いたグラフ探索アルゴリズムである A*アルゴリズム [2] をベースに, ロボットの停止時間を含めた 3 次元の経路探索を行う. 本論文ではこのアルゴリズムを時間を考慮した順番待ちによってデッドロックを回避する A*アルゴリズムとして DTQ-A*(Deadlock-avoidance Time-oriented Queue A*) と呼ぶことにする.

最後に, DTQ-A*を使ったクラウドサーバのシステムを我々の開発しているサービス基盤プラットフォーム上で実装する. このクラウドサーバとシミュレーションされたロボットを用いて DTQ-A*の処理速度の限界と回避性能の評価実験を行う.

2. 関連研究

複数の移動ロボットによる協調走行や安全に走行するための経路計画手法はここ数十年間で盛んに研究されてきている. 本章では, 関連する研究としてデッドロック回避に関連するものと, 探索アルゴリズムについて, 本研究と比較する.

2.1 自動運転におけるデッドロック回避

複数の移動物によるデッドロックはしばしば自動運転の一つの課題として多く研究されている [3], [4], [5], [6]. 先行研究においても, 物流倉庫における人と自動運転ロボット協調の実験 (図 2 に実際の実験風景を示す) では, 狭い通路でのすれ違いへの対処が一つの課題となった [7]. また, Aoki らは, 交差点のない道路における自動運転車の交差が起こりうる状態を定義し, 車車間通信によって安全にデッドロックや衝突を回避する方法を提案している [3].

自動運転ロボットにおいては, 大きさも比較的小さく, 行動範囲が広いという性質上, 自動運転ロボット同士の衝突やデッドロックはレールや回避方向, 優先度などのプロトコルで比較的簡単に回避が可能である. しかしこれらの手法は限られた空間, 狭い通路等でも汎用的に回避可能ではないという課題があった. 移動ロボットのデッドロック回避手法としては, Jager らは別々に経路が与えられた移動ロボットが, 接近を判定してデッドロックや衝突を回避するように経路を修正する手法を提案している [4].

2.2 グラフ経路計画アルゴリズム

与えられた環境内で現在地から目的地までの経路を探索する手法は古くから研究されている [8]. グラフ探索によるアルゴリズムは最適な経路を探索可能であり, 代表的な手法としてダイクストラ, A*[2], D*[9], [10] などが移動ロボットのグローバルプランニングに使われてきた. 特に近年は, ロボットの行動を有限オートマトンで表現する手法も注目され, なかでも線形時相論理 (Linear Temporal Logic) を用いたロボットの動作計画はグラフ探索による経路計画のコストを減らす効果が示されている [11], [12]. しかし, これらの状態遷移, またはその応用を用いた手法は限られた計算資源で効率的に経路探索するのには適しているが, 狭い通路や入り乱れた道のあるようなモデルを単純化することが難しい複雑な環境においては, 実装のコストが大きいため適していないと考えられる.

また, 本研究と近い時間を考慮した経路探索を行う手法

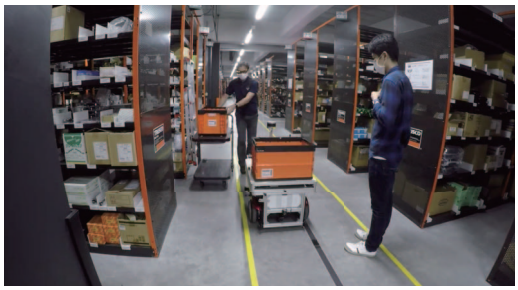


図 2: 移動ロボットを用いた物流倉庫での実証実験.

も多く研究されてきている．本研究はリアルタイムシステムとしてアルゴリズムを利用可能なクラウドシステムを実際に設計, 構築し, 現実の空間での安全な運用を想定して実装と評価を行うところに違いがある．

3. DTQ-A*アルゴリズム

初めに, 本章では経路探索アルゴリズム DTQ-A*を解説する．

3.1 前提条件

DTQ-A*の想定する環境は, 倉庫やオフィス, 図書館などの, あらかじめ走行する範囲が決まっている一定の空間内で, 特に, ロボットが大きく回避行動をとることが難しい狭い通路がある環境である．想定するロボットは自己位置推定が可能な自律移動ロボットで, クラウドサーバと常に安定して通信可能であるとする．このような環境において, 複数の移動ロボットに順番にタスクが与えられる場合, または, それぞれが独立したシステムによって目的地が与えられる状況を想定した．また, 計算処理をシンプルにする都合上, すべての移動ロボットは同じ速度 v で走行し, 経路計画はロボット一台ずつ順番に行う, 複数のロボットに同時に目的地が得られる場合でも先に与えられたロボットから順番に経路を生成することとする．

3.2 前処理

まず初めに, 経路計算を行うため, 環境内の地図を隣接するグリッドの距離が $L[m]$ の正六角形のグリッドに分割し, ロボットの半径 $r[m]$ を用いてロボットの走行不可能なグリッドの地図を取得する．A*アルゴリズムで一般的に使われるグリッドは正方形のグリッドであるが, 本研究では正六角形のグリッドを経路探索に用いる．これは, 隣接するグリッドの数を正方形の場合の 4 つから 6 つに増やせるためであり, 斜交座標への変換が容易であるため用いた．

ロボットの速度を $v[m/s]$, 回転の各速度を $\omega[rad/s]$ としたとき, ロボットがあるグリッドから隣接するグリッドへ移動する時間 $T[s]$ は $T = L/v + 2\pi/3\omega$ と見積れる．($2\pi/3\omega$ は 120 度回転する時間を表している.) この値 T を離散化されたタイムステップ幅 Ts とする．このようにタイムステップを設定することで, 停止時間と隣接するグリッドへの移動が同じ Ts の時間幅で離散化できるようになり, 時間を含む経路を与えられてもロボットは決められたほぼ一定の速度 v での走行を可能にする．

続いて, グリッドの地図を時間軸方向に Ts 秒ごとに N 個だけ拡張する．N は最大探索時間を表し $N * Ts[s]$ 先の時間まで経路探索を続けることを意味する．

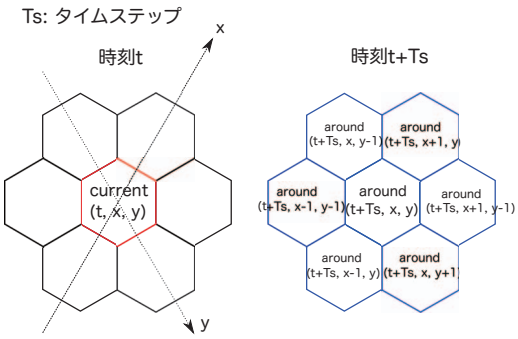


図 3: DTQ-A*における探索範囲.

3.3 ルーティング

DTQ-A*における経路計算アルゴリズムの流れを以下に示す.

(1) 障害物リストと他のロボットの経路リストの取得
ロボットが通過不可能なグリッドを示す二次元のグリッドリストと, 他のロボットの経路により占有されているグリッドを示す三次元のグリッドリストを受け取る.

(2) カレントグリッドの選択
探索リストの中から最もコストが小さいグリッドを選択しカレントグリッドとする.(最初の場合はスタート地点を選択する.) カレントグリッドがゴールの場合, 探索を終了し (5) に移る.

(3) カレントグリッドの更新
カレントグリッドと隣接するグリッドを探索リストに追加する．ここで, 隣接するグリッドとは, 選択されたグリッドからタイムステップが 1 つ進んだ先の 1 つのグリッドとそれに隣接する 6 つのグリッドの合計 7 つのグリッドである.(図 3 に示した.) ただし, 障害物, ロボットの経路リストまたは探索済リストに含まれるグリッドは除外する.

(4) コストの設定
探索リストに追加する 7 つのグリッドのコストを (親のグリッドのコスト)+(タイムステップ)+(ゴールまでの距離)/v とする．ただし, ここでいうゴールまでの距離は斜交座標におけるユークリッド距離 (式 1) である.

(5) 探索済リストの更新
カレントグリッドを探索済リストに追加し, (1) から繰り返す.

(6) 経路の取得
ゴールから一つづつ探索済リスト内で選択されたグリッドの親グリッドをスタートまでたどると経路が得られる.

(7) タイムアウト判定
現在地から目的地までの経路が存在しない場合, 経路探索は時間軸方向に発散し続けるため, 無限ループに陥ってしまう．そのため, 経路計画を始めてから N 個以上タイムステップを進んだら経路計画失敗とみなす.

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

A*アルゴリズムとの違いは, コストをゴールまでの距離ではなく, ゴールまで到達する時間を単位としてコストの低いグリッドから優先的に探索するところである. これは, 移動する障害物がある状況で, 最短時間で目的地に到達できる経路の探索を可能にする.

4. クラウドサーバ実装

4.1 複数の移動ロボットの協調走行

本研究では複数の移動ロボットがそれぞれの目的地をもって移動する場面を想定し, クラウドサーバを用いて, 狭い通路や死角の多い通路でもロボット同士がデッドロックを発生させることなく安全に走行できる手法を検討する.

クラウドサーバを用いたロボットの制御はクラウドロボティクスという分野としても注目を集めている. 複数の移動ロボットをクラウドサーバを用いて協調走行する手法として, 以下の二つの手法が考えられる. (i) ロボットの制御すべてをクラウドサーバが担当する. (ii) 一部の制御のみクラウドサーバが担当する. (i) の手法はすべてのロボットの情報が一つに集約されるため, 理想的には最も効率的で協調を行うことが可能であるが, 大規模化が難しいのと, 大量のセンサ情報によるデータ損失や通信遅延のリスクが存在するため多くは用いられていない.

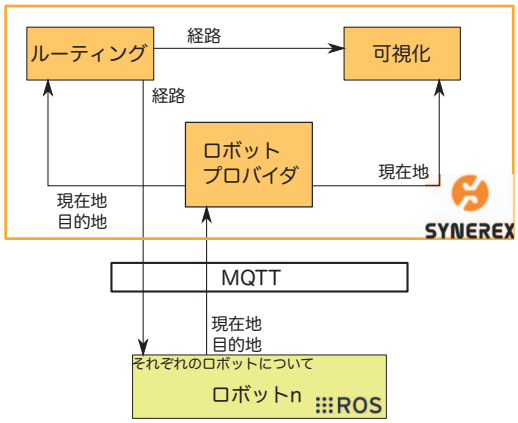


図 4: システム構成.

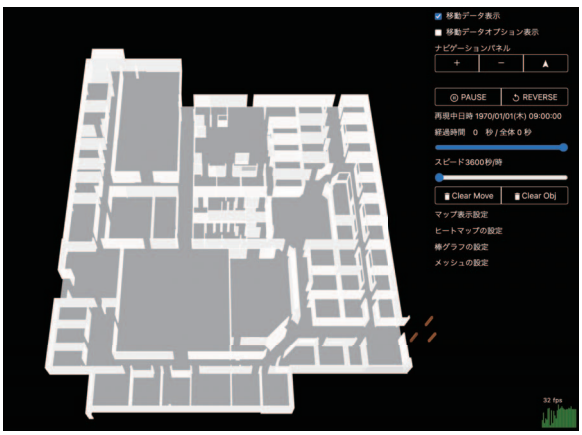


図 5: Harmoware-VIS を用いた可視化.

よって, 本研究では (ii) に属するクラウドサーバが常にロボットの経路計算を代行する手法を用いてロボットの協調走行を行うシステムを設計した. 具体的には, ロボットの主な制御はロボット側が行い, 経路計画のみクラウドサーバが代行する. 一般的に, 移動ロボットの経路計画はスタート地点からゴール地点までの最適な経路を幾何学的に計算するものであり, 走行する環境が複雑で大きくなるほど, 処理に時間がかかるという問題がある. そのため, クラウドサーバが経路計画を行うことで, 他のロボットの経路も知りつつ, 計算コストの大きい処理を分散できるというメリットがある.

4.2 システム構成

図 4 に本研究で実装したシステムの構成図を示す. 本研究ではクラウドサーバ全体を動かすミドルウェアとして, Synerex[13] を用いる. Synerex とは我々が開発している需給交換サービス基盤であり, 先行研究で人流 [14] やロボット [7] のデジタルツインとしても利用してきた. Synerex を用いて機能をマイクロサービス毎に分散することでシステムの拡張性が高くなり, 新たなロボットやサービスを追加することが容易になるため使用する.

また, ロボットと Synerex を通信するため, 軽量で Publish/Subscribe 型の通信モデルのプロトコルである MQTT[15] を使用した. 以下に図 4 に示された Synerex 上の各プロバイダの説明を示す.

4.2.1 ロボットプロバイダ

ロボットプロバイダは接続されるロボット 1 台につき起動され, 各ロボットとクラウドサーバ間を MQTT によって繋げる. ロボットプロバイダをロボットと, クラウドサーバの間に仲介させることによって, 異なる種類のロボットが混在していても, 組み込むことが容易となるというメリットがある.

4.2.2 ルーティングプロバイダ

ルーティングプロバイダはロボットの現在地と目的地を受け取り, 生成された経路を送信する. 手法は 3 章にて示したものである. 平面上のロボットの位置を表す二次元リストを N 個生成し, 経路が要求されるたびに (リストの最小時間からの経過時間)/(タイムステップ) の整数分だけ時間軸方向にリストをずらすことで, $N * Ts$ 秒先までのロボットの位置を表現した. また, DTQ-A*は通常の A*アルゴリズムよりも計算時間が増加してしまう問題があるため, 環境内で初めに計算されるロボットの経路探索時のみ六角形グリッドの A*アルゴリズムで計算するようにした.

4.2.3 可視化プロバイダ

また, 我々が開発しているマップ上に動的なユーザーインターフェイスを生成可能な Harmoware-VIS[16] という

可視化ライブラリを用いて可視化プロバイダも実装した. これはロボットの現在地, 経路情報を受け取り表示することで, リアルタイムでの可視化による安全性の確認が可能にする. 図 5 にその可視化の様子を示した.

5. 評価実験

本章では, DTQ-A*の回避精度を評価するための評価実験を記す. 実験は DTQ-A*を用いて実装したクラウドサーバシステムとシミュレーション環境上に生成した複数台のロボットを用いた. デッドロック, 接触を起こしうる 2 つのシナリオを設定し, 回避性能の最適さと処理速度の評価を行う.

5.1 シミュレーション環境.

移動ロボットは ROS(Robot Operating System)[17] を用いて自律走行システムを構築する. 具体的には, クラウドサーバから受け取った経路を時刻情報に合わせて移動するポイントに PID 制御で追従するようにした. そのため, ロボットは生成された経路の時刻から数百ミリ秒遅れて経路を追従することになる. また, 本研究において障害物を検知する機能は実装していないため, 止まることによるデッドロックは発生しえない. そのため, ロボットが接触した場合はデッドロックが発生したとする. 経路計算時に用いるロボットの速度はすべて速度 $0.3m/s$, 角速度 $0.5rad/s$ とした.

加えて, シミュレーションによる評価を行うため, シミュレータとして Gazebo[18] を用いた. Gazebo とは, ROS のシステムをシミュレーションするために作られたオープンソースの 3D ロボットシミュレータである. 実験を行う環境として, 狭い通路が多く, 複数ロボットによるデッドロックが発生しやすいと考えられるウィローガレージのモデルを使用した. 図 6 に Gazebo 上でのシミュレーション環境を示す. ROS で提供されている gmapping という LiDAR の点群情報から地図を生成するツールを用いて環境内の地図を生成し, 経路探索の障害物リストの生成に用いた. また, クラウドサーバとシミュレーションは別の計算機を用いて実行し, クラウドサーバには Apple M1 チップを搭載した MacBookPro(2020) を用いた.

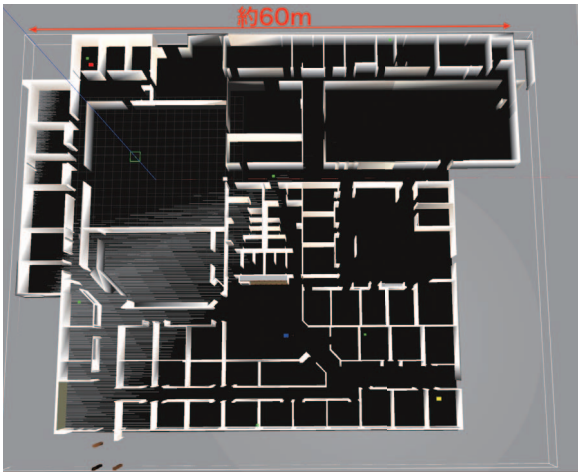


図 6: シミュレーション環境 (ウィローガレージ).

5.2 処理速度の制約

DTQ-A*の抱える課題の一つとして, 平面空間の探索への時間軸追加による処理時間の増加が考えられる. 経路計算にかかる時間はロボットの制御の遅延に直結するため, どのような環境だと応用が難しくなるのか評価する必要があった.

計算速度に大きな影響を与えるのはグリッドの大きさであるが, グリッドの大きさを $0.2m$ より小さくすると計算時間が急増し (タイムアウト), $0.6m$ より大きくすると, 一部の狭い通路が通行不可能グリッドとみなされてしまうため, 本環境では, グリッドの大きさを本環境では最も適切と考えられる $0.4m$ に設定した.

また, シミュレーション環境全体を対象とした経路生成の実験を試みたが, 障害物のない広い部屋を経由することで探索範囲が広がって, 計算時間が急増するという問題も発生したため, 本論文では広い部屋のない狭い空間で実験することにした. 広い部屋における探索範囲の急増は地図上に仮想的に障害物を設置して抑えることが可能であるが, 今後の課題の一つでもある.

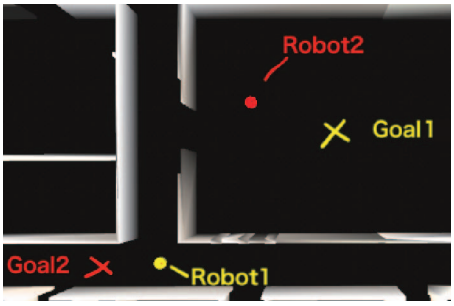


図 7: シナリオ 1: 設定.

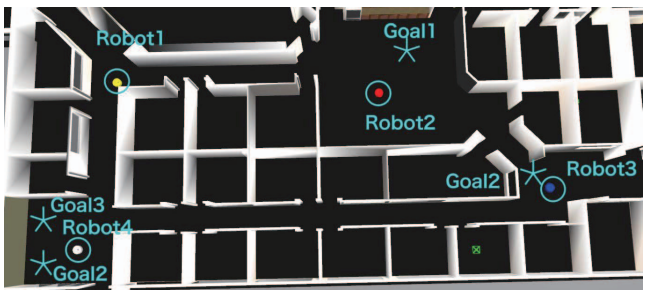


図 8: シナリオ 2: 設定.

表 1: シナリオ 2 における経路探索の処理時間.

	Robot1	Robot2	Robot3	Robot4
探索時間 [s]	0.00480	0.0216	17.617	11.070

5.3 シナリオ設定

複数台の移動ロボットが特定のエリア内の狭い通路で入れ違う状況を想定し, 図 7, 図 8 に示すように二つのシナリオを設定した. ここで, Robot はスタート位置を示し, 番号の少ないロボットから順番に Goal を与え, デッドロック, 衝突を発生させずに走行可能かどうかを試した.

シナリオ 1 では, 迂回ができない 1 本の道で 2 台のロボットがすれ違う状況を想定し, シナリオ 2 では, 移動ロボットを 4 台に増やし, 狭い空間内でばらばらに入れ違う状況で本手法を評価した. また, シナリオ 2 ではロボットの増加による経路探索の時間への影響を評価するため, 経路探索の処理時間も記録した.

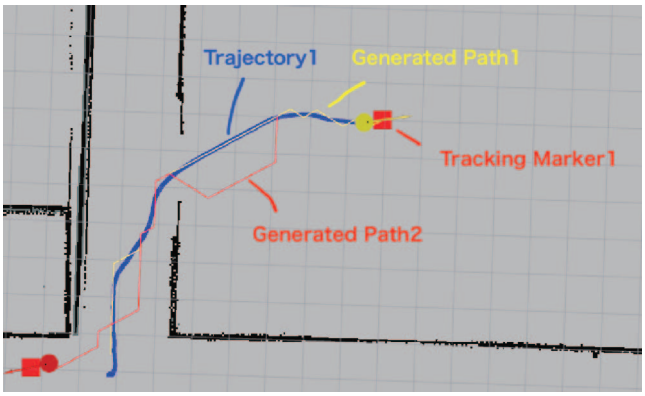


図 9: シナリオ 1: 生成された経路 (移動後).

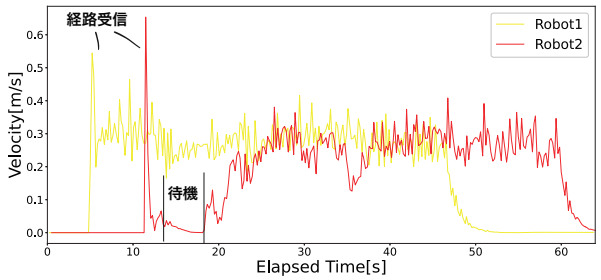


図 10: シナリオ 1: ロボットの移動速度の変化.

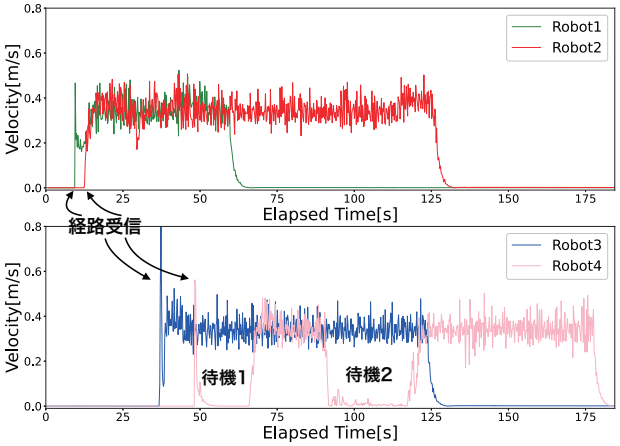


図 11: シナリオ 2: ロボットの移動速度の変化.

5.4 結果

それぞれのシナリオにおいて, 生成された経路を図 9, 12 に示す. 図内の Path は生成された経路を, Trajectory は実際にロボットが通った軌跡, 赤色の正方形は PID 制御で追従するのに用いた経路情報に沿って自動で動くポイントをそれぞれ示す. また, ロボットの移動速度の変化を図 10, 11 に示した. これはどちらも一回だけ実験した記録である. また, 表 1 にはシナリオ 2 における経路探索の処理時間を記した.

ロボットの速度はクラウドサーバに送信された位置情報をもとに計算したものである. 移動速度の変化において, 開始時に短い時間で急に速度が大きく上がっている箇所が見られる. これらはロボットが経路を受信した際に, 現在地からスタート地点のグリッド中心までのずれの分だけ PID 制御の微分項が大きく働いてしまったことを表しているが, 結果的には経路が受信された時刻が示されている.

- シナリオ 1

図 12, 10 に示すように, 後に経路を生成されたロボット 2 はスタート地点において経過時間 12 秒から 14 秒の間に待機によって通過待ちをした後, 一度迂回してロボット 1 の経路を避けながら目的地まで進んだ. これは後に経路を生成した赤のロボットが待機によって接近を回避したことを示している.

- シナリオ 2

図 12 に示すように, 横に伸びる上下 2 つの通路のうち, 初めに計算されたロボット 1 によって上側の通路が占有され, 残りの迂回したロボット 2 と最短経路を通ろうとしたロボット 3, 4 の 3 台が下側の経路を通過する結果となった. このような密集した状況でも, 図 11 に示されるようにロボット 4 が一度狭い部屋に寄って通過待ちをしたことで, 反対側から来ていたロボット 2, 3 とのデッドロックが回避された. ロボット 2, 3 については 3 の経路生成に時間がかかったため, ロボット 2 を追従する形となって接触が回避された.

一方で, 処理時間については, 表 1 に示したように, 最短経路が他のロボットによって塞がれた後に計算されたロボット 3, 4 の処理時間がそれぞれ 17 秒, 11 秒と急激に増加してしまい, ロボットの移動開始が大きく遅れる結果となった. 加えて, ロボット 3 は経路探索時間で遅れた分を追いつくために, 動き始めで一時的に速度が大きくなっている.

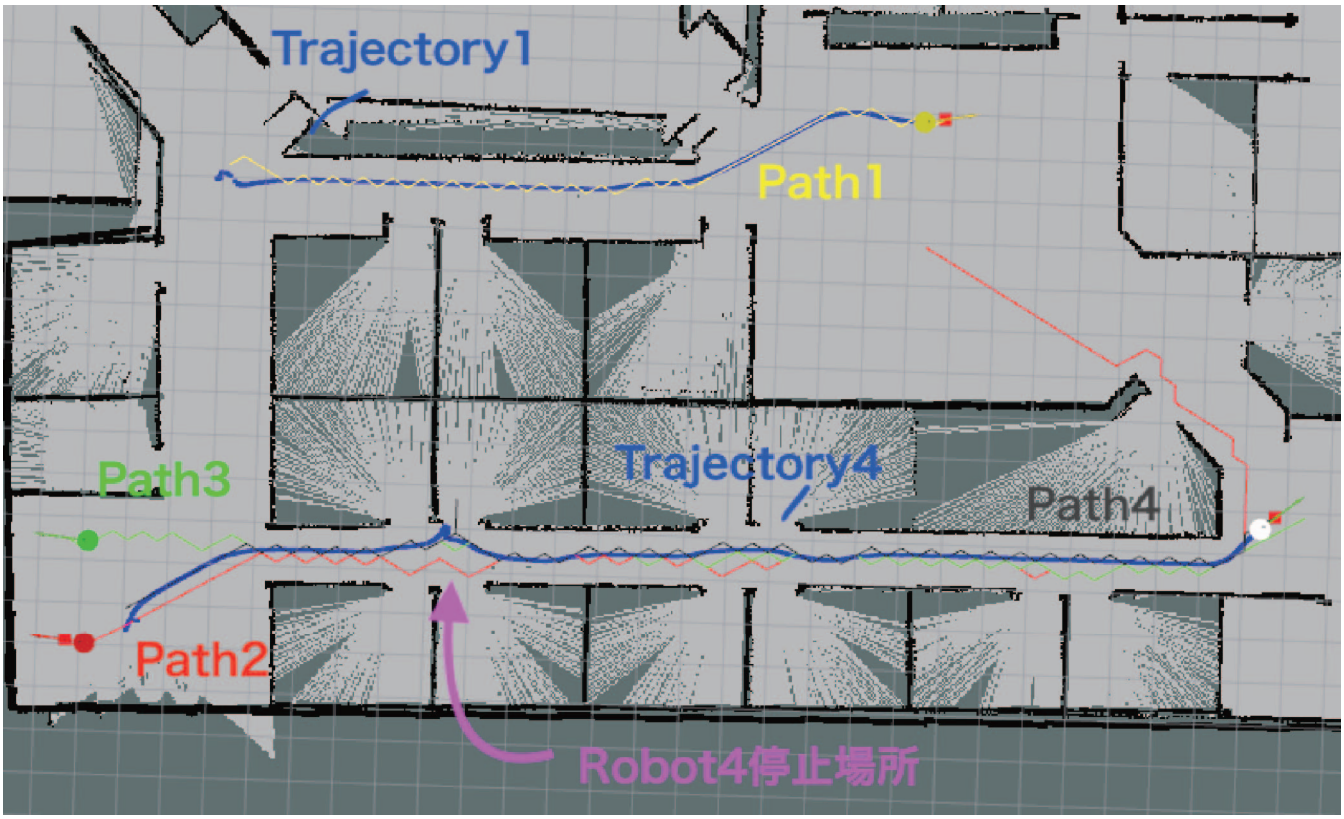


図 12: シナリオ 2: 生成された経路 (移動後).

5.5 議論

シミュレーションされたロボットを用いた実験によって, DTQ-A*を用いて時刻を含む経路を生成し, それにロボットを追従させることで, デッドロックのない複数ロボットの移動が実現可能なことを示した. しかしながら, 経路が 1 台ずつ生成される状況を想定しているため, 本実験のように同時に経路が複数与えられる場合には処理速度の遅さが顕著となって現れてしまった.

生成された経路に注目すると, 停止した後に移動する場合と, 移動した後に停止し, 再度移動する場合の 2 つのパターンが見られた. どちらも別のロボットを待つための停止だと考えると, 相手のロボットが狭い通路を通過するまでの時間は同じである。そのため, どちらのパターンにおいても, 合計した停止時間は同じになるはずである。本論文で提案した DTQ-A*には生成される時間を含む経路に一意性がなく, デッドロックを回避可能な複数の最短経路の中の一つが選択されていると言える. したがって, DTQ-A* に最短経路を一意に決定できる要素を加えることができれば, 経路探索時間の増加が抑えられるはずである. そのために, 停止と移動の二つの行動を 2 値の状態として表現し, 移動と停止のパターンに制約を持たせる方法が考えられる. これは今後の課題である.

6. 結論

本論文では, 移動ロボット同士が狭い通路でデッドロックを発生させてしまう環境を想定し, クラウドサーバを用いた経路計画手法により, デッドロックを回避する最適な 3 次元経路を生成する手法を提案した. 加えて, クラウドサーバとロボットのシミュレーション環境を実装し, 経路探索のグリッドサイズによる処理速度の限界を評価した. また, デッドロックの起こりうる複数のシナリオにおいて本手法を適用し, 回避性能の確認を行い, デッドロックを回避しつつ狭い通路で協調走行が可能であることを示した.

最後に, 本研究の今後の応用と課題について記述する. 一つは, 実世界においてロボットの走行環境を構築し, 評価実験を行う予定である. これは, シミュレーションによる評価はロボットの行動が理想的な条件であると仮定されたものであり, 実機の移動ロボットを用いて実世界における実現可能性を示す必要があるためである. また, 本論文は複数の移動ロボットの経路の共有のみに注目したが, 人のような動きが予測できない障害物が多い環境では, 経路情報の共有のみでは, 安全な走行は難しいため, 今後はコストマップやセンサ情報等の共有も考慮したシステム設計を行う予定である.

謝辞

本研究の一部は、JST COI-NEXT（JPMJPF2006）の支援を受けています.

参考文献

[1] Starship: The self-driving food and groceries delivery robot. https://www.starship.xyz/, 2021. Online; accessed 8 May 2021.

[2] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[3] Shunsuke Aoki and Ragunathan Rajkumar. Dynamic intersections and self-driving vehicles. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 320–330. IEEE, 2018.

[4] Markus Jager and Bernhard Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, Vol. 3, pp. 1213–1219. IEEE, 2001.

[5] Shunsuke Aoki and Ragunathan Rajkumar. V2V-based synchronous intersection protocols for mixed traffic of human-driven and self-driving vehicles. In *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1–11. IEEE, 2019.

[6] Marius Kloetzer, Cristian Mahulea, and José-Manuel Colom. Petri net approach for deadlock prevention in robot planning. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–4. IEEE, 2013.

[7] Yuto Fukushima, Yusuke Asai, Shunsuke Aoki, Takuro Yonezawa, and Nobuo Kawaguchi. Digimobot: Digital twin for human-robot collaboration in indoor environments. In *IEEE International Conference on Intelligent Vehicles (IV)*. IEEE, 2021.

[8] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3260–3267. IEEE, 2011.

[9] Sven Koenig and Maxim Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, Vol. 1, pp. 968–975. IEEE, 2002.

[10] Dave Ferguson and Anthony Stentz. Field d*: An interpolation-based path planner and replanner. In *Robotics research*, pp. 239–253. Springer, 2007.

[11] Danish Khalidi, Dhaval Gujarathi, and Indranil Saha. T*: A heuristic search based path planning algorithm for temporal logic specifications. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8476–8482. IEEE, 2020.

[12] Marius Kloetzer and Cristian Mahulea. Ltl planning in dynamic environments. *IFAC Proceedings Volumes*, Vol. 45, No. 29, pp. 294–300, 2012.

[13] 河口信夫, 米澤拓郎, 廣井慧ほか. Synerex: 超スマート社会を支える需給交換プラットフォームの設計コンセプトと機能. 研究報告ユビキタスコンピューティングシステム (UBI), Vol. 2020, No. 49, pp. 1–6, 2020.

[14] Yoshiteru Nagata, Takuro Yonezawa, and Nobuo Kawaguchi. Person-flow estimation with preserving privacy using multiple 3d people counters. In *Science and Technologies for Smart Cities 2021 (Transaction of 5th EAI International Conference on IoT in Urban Space)*. EAI, 2021.

[15] Banks Andrew and Gupta Rahul. MQTT Version 3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, Oct. 29 2014. Accessed: 2021-04-21.

[16] Harmoware/harmoware-vis: Spatial-temporal visualization library using deck.gl. https://github.com/Harmoware/Harmoware-VIS, 2021. Online; accessed 5 May 2021.

[17] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, Vol. 3, p. 5. Kobe, Japan, 2009.

[18] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3, pp. 2149–2154. IEEE, 2004.