令和 7 年度 卒業論文
自律移動ロボットのための価値反復と A*探索を組み合わせた大域経路計画

中村啓太郎
China Institute of Technology

2026 年 1 月 29 日

謝辞

本研究に取り組み，本稿を執筆するにあたり，ご指導を頂いた上田隆一教授および上田研究室のみなさまに感謝します.

目次

第 1 章

序論

1.1 研究背景

1.1.1 ロボティクスに対する社会的要請

少子高齢化と労働生産性の課題

21 世紀に入り，先進諸国においては少子高齢化に伴う生産年齢人口の減少が深刻な社会問題となっている．特に日本においては，国立社会保障・人口問題研究所の推計によれば，生産年齢人口（15〜64 歳）は 1995 年をピークに減少傾向にあり，2070 年には約 4,500 万人（2020 年比で約 4 割減）まで落ち込むことが予測されている [国立 23]．この人口構造の変化は，経済成長の鈍化のみならず，社会インフラの維持そのものを脅かす要因となっている．

物流業界においては，「物流クライシス」と呼ばれる状況が顕在化している．電子商取引（E-commerce）の爆発的な普及により，小口配送の需要が急増している．国土交通省の調査によれば，宅配便取扱個数は年間 50 億個（2023 年度）を超え，過去 10 年間で約 1.3 倍に増加している [?] トラックドライバーや倉庫内作業員の不足は慢性化しており，労働環境の悪化と配送網の維持困難が懸念されている．また，製造業の現場においても，熟練工の引退に伴う技術継承の問題や，単純搬送作業への人員配置の困難さが指摘されている．

さらに，医療・介護の分野では，高齢者人口の増加に対し，介護従事者の数は圧倒的に不足している．厚生労働省の推計では，2040 年度には約 272 万人の介護職員が必要とされているが，現状のままでは数十万人規模の供給不足に陥る可能性が指摘されている [?]．病院内での検体搬送，リネン類の回収，あるいは介護施設における見守りや配膳など，非接触かつ定型的な業務の負担軽減は，強い需要がある．

業務自動化の変遷と限界

こうした労働力不足を補う手段として，古くから業務の自動化（Automation）が進められてきた．工場内物流においては，無人搬送車（Automated Guided Vehicle: AGV）

が 1950 年代に Barrett Electronics 社によって開発されて以来 [?]，長年にわたり導入が進められてきた．従来の AGV は，床面に敷設された磁気テープや反射テープ，あるいは二次元コードといった物理的なガイドをセンサで読み取りながら走行する．

AGV は，環境が固定されており，かつタスクが定型的である場合には極めて高い効率と信頼性を発揮する．しかしながら，その導入と運用には以下のような大きな制約が存在する．

1. **インフラ敷設コスト**: 走行経路すべてにガイドを設置する必要があり，導入時の工事コストや期間が甚大である．
2. **レイアウト変更の柔軟性欠如**: 製造ラインや倉庫のレイアウトを変更する際，ガイドの敷設し直しが必要となり，多大なコストとダウンタイムが発生する．
3. **動的環境への非適応**: Fragapane ら [?] が指摘するように，想定された経路上に障害物が存在した場合，AGV はその場で停止することしかできず，回避して目的地へ向かうことができない．

これらの制約は，人や有人フォークリフトが頻繁に行き交う物流倉庫や，一般の人々が存在する病院・商業施設といった「非構造化環境（Unstructured Environment）」へのロボット導入を阻む大きな障壁となっていた．

自律移動ロボット（AMR）の台頭

AGV の課題を克服する次世代の搬送ソリューションとして登場したのが，自律移動ロボット（AMR）である．AMR は，LiDAR（Light Detection and Ranging）やカメラといった外部界センサを搭載し，周囲の環境をリアルタイムに認識する．事前に作成した環境地図と現在のセンサ情報を照らし合わせることで自身の位置を推定し，ガイドレスでの走行を可能にする．

AMR の最大の利点は，その「柔軟性」にある．物理的なガイドを必要としないため，導入時の工事が不要であり，ソフトウェア上の設定変更のみで走行エリアや経路を変更できる．また，経路上に障害物を検知した際には，自律的に回避経路を生成し，タスクを継続することが可能である．この特性により，AMR は従来 AGV が導入困難であった動的な環境への適用が進んでおり，Industry 4.0 の中核を担う技術として注目されている．

1.1.2　自律移動ロボットの技術構成

自律移動ロボットが実環境でタスクを遂行するためには，ハードウェアとソフトウェアの両面において高度な統合が必要とされる．本項では，AMR を構成する主要な要素技術について概説する．

センシング技術

ロボットが周囲の環境から情報を得るための技術である．

- **LiDAR**: レーザー光を照射し，反射光が戻ってくるまでの時間（Time of Flight）や位相差から距離を計測する．2 次元平面をスキャンする 2D LiDAR が主流であったが，近年では 3 次元点群を取得可能な 3D LiDAR の低価格化が進んでいる．
- **カメラ**: RGB 画像に加え，深度情報を取得できる RGB-D カメラやステレオカメラが利用される．Visual SLAM や物体認識との親和性が高い．
- **オドメトリ（Odometry）**: 車輪の回転数（エンコーダ値）や IMU（慣性計測装置）のデータから，ロボットの相対的な移動量を推定する．短期的には高精度だが，累積誤差が生じるため単独では長距離移動に適さない．

自己位置推定（Localization）技術の進展

「今どこにいるか」を知る技術は，自律移動の根幹である．オドメトリの累積誤差を補正し，地図座標系上での絶対位置を特定するために，様々な手法が開発されてきた．

■確率的ロボティクスの枠組み 1990 年代以降，Thrun らによって提唱された「確率的ロボティクス（Probabilistic Robotics)」は，センサのノイズや環境の不確実性を確率分布として扱うことで，ロバストな自己位置推定を可能にした．代表的な手法として，モンテカルロ位置推定（Monte Carlo Localization: MCL）あるいはパーティクルフィルタ（Particle Filter）がある．これは，ロボットの存在確率分布を多数の粒子（パーティクル）で表現し，センサ観測と動作モデルに基づいて粒子の重みと位置を更新する手法である．MCL は，「大域的自己位置推定（Global Localization)」や「誘拐ロボット問題（Kidnapped Robot Problem)」への対処能力を有しており，現在の AMR のデファクトスタンダードとなっている．

■SLAM (Simultaneous Localization and Mapping) 地図を持たない未知の環境においては，自己位置推定と同時に環境地図の作成を行う SLAM 技術が用いられる．

- **LiDAR SLAM**: GMapping や Cartographer など，レーザーレンジファインダを用いたグリッドマップ生成手法．幾何的な精度が高い．
- **Visual SLAM**: ORB-SLAM など，カメラ画像の特徴点を用いて疎な地図（Feature Map）を作成する．テクスチャの豊富な環境に強い．

さらに近年では，Graph-based SLAM と呼ばれる，ロボットの姿勢（ノード）と観測制約（エッジ）をグラフ構造として表現し，非線形最小二乗法によって全体最適化を行う手法が主流となっている．

1.1.3 自己位置推定から経路計画へ

自己位置推定技術の成熟により，ロボットは環境内での自身の位置をセンチメートルオーダーの精度で特定できるようになった．また，SLAM 技術により，障害物の配置や通行可能な領域（Free Space）を表す占有格子地図（Occupancy Grid Map）を自動的に

生成することも可能となった.

しかし,「自分がどこにいるか」と「周囲がどうなっているか」を知ることは，自律移動の必要条件ではあるが十分条件ではない．ロボットが実際にタスクを遂行するためには，現在地（Start）から目的地（Goal）まで，障害物を回避しつつ，かつ効率的（最短時間，最小エネルギーなど）に到達するための軌道を決定しなければならない．これが「経路計画（Path Planning)」である.

経路計画は，自己位置推定の結果と環境地図を入力とし，ロボットのアクチュエータ（モータ）への制御指令を出力とする，自律移動システムの「頭脳」に相当する部分である．自己位置推定がいかに高精度であっても，経路計画が不適切であれば，ロボットは遠回りをするか，狭い通路で立ち往生するか，最悪の場合は動的障害物と衝突する危険性がある.

特に，社会実装が進むにつれて，ロボットが稼働する環境はより複雑化している.

- 静的な障害物だけでなく，人や他のロボットが移動する動的環境.
- 路面状況によるスリップや，センサ計測の不確実性.
- 複数の目的地を効率よく巡回する等のタスクの複雑化.

これらの要因を考慮し，安全かつ最適な経路をリアルタイムに導出することは，依然として困難かつ重要な研究課題である．次節では，この経路計画に関する従来研究を概観し，本研究で着目する価値反復法の位置付けを明らかにする.

1.2 従来研究

1.2.1 移動ロボットにおける経路計画の階層構造

移動ロボットのナビゲーションシステムは，計算負荷と応答性のバランスを保つため，一般的に以下の 2 つの階層に分離して設計される.

1. **大域経路計画（Global Path Planning）**：環境全体の地図情報（事前地図）に基づき，スタートからゴールまでの大局的な最適ルートを生成する．更新頻度は比較的低く（数秒〜数分に 1 回，またはトポロジカルな変更時)，静的な障害物の回避を主眼とする.
2. **局所経路計画（Local Path Planning / Obstacle Avoidance）**：大域経路に追従しつつ，搭載センサで検知した未知の障害物や動的障害物をリアルタイムに回避するための制御入力を生成する．更新頻度は高く（10Hz〜100Hz)，ロボットの運動学的制約（Kinematics）や動力学的制約（Dynamics）を考慮する．Dynamic Window Approach (DWA) や Model Predictive Control (MPC) などが代表的である.

本研究は，このうち「大域経路計画」に焦点を当てるものである．局所計画器がいかに

優秀であっても，大域的な指針が不適切（例えば，物理的に通過不可能な狭路へ誘導してしまう，あるいはU字型の袋小路に迷い込む）であれば，ロボットはゴールへ到達できない．したがって，環境の大域的な構造とコストを正しく評価する大域経路計画アルゴリズムが不可欠である．

1.2.2 決定論的アプローチによる経路計画

大域経路計画の歴史は古く，多くの手法は環境をグラフ構造としてモデル化する「グラフ探索問題」に帰着される．

グリッドベースの探索手法

環境地図を格子状（グリッド）に分割し，各セルをノード，隣接セルへの移動をエッジとみなす手法である．

■Dijkstra 法 Edsger W. Dijkstra によって考案された Dijkstra 法は，非負の重み付きグラフにおける単一始点最短経路問題を解くアルゴリズムである．スタートノードから順に，隣接ノードへの移動コストを累積し，確定したノードから探索範囲を広げていく（幅優先探索のコスト付き版と言える）．あるノード n までの最小コストを $g(n)$ とするとき，常に $g(n)$ が最小となるノードを展開することで，数学的に最短経路が保証される．しかし，探索が全方位に均等に広がるため，ゴールの方角情報利用されず，探索範囲が膨大になる欠点がある．

■A*アルゴリズム A*（A-Star）アルゴリズムは，Dijkstra 法にヒューリスティック関数 $h(n)$ を導入することで探索を効率化した手法である．Hart らによって提案された．評価関数 $f(n)$ を以下のように定義する．

$$f(n) = g(n) + h(n) \tag{1.1}$$

ここで，$g(n)$ はスタートからノード n までの実コスト，$h(n)$ はノード n からゴールまでの推定コスト（ヒューリスティック）である．移動ロボットの場合，$h(n)$ として現在のセルからゴールセルまでのユークリッド距離やマンハッタン距離が用いられる．$h(n)$ が実際の最短コストを決して上回らない（許容的，Admissible である）場合，A*アルゴリズムは最適解を保証しつつ，Dijkstra 法よりも少ないノード展開数で解に到達できる．現在でも最も広く使われている標準的なアルゴリズムである．

サンプリングベースの手法

高次元の構成空間（Configuration Space: C-Space）を持つロボット（多関節アームなど）や，広大な環境においては，グリッドベースの手法は計算量が爆発する．これに対し，空間内の点をランダムにサンプリングしてグラフを構築する手法が提案されている．

■RRT (Rapidly-exploring Random Tree) LaValle によって提案された RRT は，ランダムにサンプリングされた点に向かってツリーを拡張していくことで，空間を急速に被覆する手法である．非ホロノミック拘束（自動車のような移動制約）を持つロボットの経路計画にも適用しやすい．しかし，RRT によって生成される経路は最適性が保証されず，ジグザグな無駄の多い経路になりがちである．これを改良した RRT*（RRT-Star）は，漸近的最適性を有するが，収束には時間を要する．

■PRM (Probabilistic RoadMap) PRM は，学習フェーズで空間全体にランダムなノードを配置してロードマップ（グラフ）を構築し，クエリフェーズでスタートとゴールをそのロードマップに接続して経路を探索する手法である．多点間の移動を繰り返すようなタスクに適しているが，狭い通路（Narrow Passage）の通過が困難であるという問題が知られている．

1.2.3 価値反復法（Value Iteration）

前節で述べた A*や RRT は，基本的に「ある特定のスタートからゴールへの一本のパス」を見つける手法である．これに対し，本研究で取り扱う「価値反復法」は，環境内の**すべての状態（位置）**について，ゴールへの最適方策を計算するアプローチである．この手法は，環境の不確実性を確率的に扱うことが可能であり，外乱に対して極めて堅牢（Robust）なナビゲーションを実現する．

マルコフ決定過程（MDP）による定式化

価値反復法を理解するためには，その基礎となるマルコフ決定過程（Markov Decision Process: MDP）の定義が必要である．MDP は，エージェントが環境と相互作用しながら学習・行動決定を行うための数理モデルであり，以下の 4 つの要素の組 $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ で定義される．

1. **状態集合 $\mathcal{S}$ (State Space)**: ロボットが取り得るすべての状態の集合．2 次元グリッドマップ上での経路計画の場合，各グリッドセル (x, y) が一つの状態 $s \in \mathcal{S}$ に対応する．さらに，ロボットの方位 θ を含めて (x, y, θ) を状態とすることもある．
2. **行動集合 $\mathcal{A}$ (Action Space)**: 各状態でロボットが選択可能な行動の集合．グリッドマップ上では，隣接する 8 近傍（上下左右＋斜め）への移動や，その場での停止などが行動 $a \in \mathcal{A}$ となる．
3. **遷移確率 $\mathcal{P}_a(s, s')$ (Transition Probability)**: 状態 s で行動 a を選択したときに，次の時刻に状態 s' へ遷移する確率．

$$\mathcal{P}_a(s, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a) \tag{1.2}$$

決定論的な環境（A*などが想定する世界）では，ある行動を行えば 100% 意図した隣接セルへ移動する．しかし実環境では，タイヤのスリップや制御誤差により，意

図したセルへ移動できない場合がある．MDP ではこの不確実性を確率分布として明示的にモデル化できる．例えば，「前進」を選択しても，10% の確率で「横滑り」する，といった表現が可能である．

4. **報酬関数 $\mathcal{R}_a(s, s')$ (Reward Function)**: 状態遷移に伴って得られる即時報酬（またはコスト）．経路計画問題においては，通常「コストの最小化」または「負の報酬の最大化」として定式化される．例えば，ゴール状態に到達したときに大きな正の報酬を与え，障害物に衝突したときに大きな負の報酬（ペナルティ）を与える．また，移動にかかる時間やエネルギーを表現するため，各ステップごとにわずかな負の報酬（ステップコスト）を与える．

ベルマン方程式と価値関数

MDP の目的は，各状態でどのような行動をとるべきかというルール，すなわち「方策（Policy） $\pi : \mathcal{S} \to \mathcal{A}$」を見つけることである．最適な方策 π^* を見つけるために，「状態価値関数 $V^\pi(s)$」を導入する．これは，ある状態 s からスタートし，方策 π に従って行動し続けたときに得られる将来の報酬の総和（期待値）である．

割引率を γ $(0 \leq \gamma < 1)$ とすると，価値関数は以下のように定義される．

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, \pi\right] \tag{1.3}$$

最適な方策 π^* に従ったときの価値関数を最適状態価値関数 $V^*(s)$ と呼ぶ．$V^*(s)$ は，以下のベルマン最適方程式（Bellman Optimality Equation）を満たす．

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_a(s, s') \left[\mathcal{R}_a(s, s') + \gamma V^*(s')\right] \tag{1.4}$$

この方程式は再帰的な構造をしており，「ある状態の価値は，そこで最適な行動をとった際に期待される即時報酬と，遷移先の状態の価値の割引和によって決まる」ことを意味している．

1.2.4 価値反復アルゴリズム

ベルマン最適方程式 (1.4) を用いて，反復計算により $V^*(s)$ を求める手法が価値反復法（Value Iteration）である．アルゴリズムの手順は以下の通りである．

1. **初期化**: すべての状態 s について，$V(s)$ を任意の値（通常は 0）に初期化する．
2. **反復更新**: 以下の更新式を，すべての状態 s に対して適用する．

$$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_a(s, s') \left[\mathcal{R}_a(s, s') + \gamma V_k(s')\right] \tag{1.5}$$

ここで k は反復回数を表す．

3. **収束判定**: 全状態における価値関数の更新量 $|V_{k+1}(s) - V_k(s)|$ の最大値が，事前に定めた閾値 ϵ 未満になれば停止する．
4. **方策抽出**: 収束した価値関数 $V^*(s)$ を用い，各状態で価値を最大化する行動を選択する貪欲方策（Greedy Policy）を得る．

$$\pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s' \in \mathcal{S}} \mathcal{P}_a(s, s') \left[\mathcal{R}_a(s, s') + \gamma V^*(s') \right] \tag{1.6}$$

この計算により，環境内のあらゆる場所からゴールへ向かうための最適な「勾配」が得られる．これはポテンシャル場に似ているが，ポテンシャル法が抱える「局所解（Local Minima）」の問題（ゴール以外の窪みにハマって出られなくなる現象）が発生しないという強力な数学的保証がある．なぜなら，ベルマン方程式による更新は，大域的な最適性を伝播させる処理だからである．

1.2.5　価値反復法の移動ロボットへの適用事例と課題

古典的な適用とナビゲーション関数

移動ロボットの経路計画において，価値反復法の概念を早期に取り入れた研究として，Konolige による Gradient Method[?] がある．彼は，障害物との距離に基づくコストマップ上で波及法（Wavefront Propagation）に類似した計算を行い，ゴールからの距離（コスト）を表すナビゲーション関数（Navigation Function）を作成した．この関数は，実質的に決定論的 MDP（$\gamma = 1$, 遷移確率 1.0）における価値関数と等価である．ロボットはこの関数の勾配（Gradient）に従って降下することで，滑らかかつ最短の経路でゴールへ到達できる．この手法は，A*などの探索手法が「離散的なセルの列」を経路として出力するのに対し，連続的なベクトル場を生成するため，制御系への入力として扱いやすいという利点がある．

不確実性の考慮と確率的 MDP

Thrun ら [?] は，著書『Probabilistic Robotics』において，センサノイズやアクチュエータ誤差を考慮した MDP ベースのプランニングの重要性を説いている．例えば，狭い通路を通る際，A*アルゴリズムなどの幾何的な最短経路探索では，壁ギリギリを通るルートを選択しがちである．しかし，実ロボットには移動誤差があるため，壁に衝突するリスクが高い．確率的な遷移モデル $\mathcal{P}$ を組み込んだ価値反復法を用いると，壁際での移動は「衝突して大きなペナルティを受ける確率がある行動」として評価される．その結果，多少遠回りであっても，壁から距離を取った安全な広い通路を選択するような，リスク回避的な経路（Risk-Aware Path）が自然に生成される．これは，安全性が最優先されるサービスロボットにおいて極めて重要な特性である．

計算コストの問題と高速化手法

価値反復法の最大の欠点は，計算コストである．状態空間 $\mathcal{S}$ のサイズに対して計算量が線形（1 回の反復あたり $O(|\mathcal{S}||\mathcal{A}|)$）に増加する．広大な環境を高い解像度でグリッド化すると，状態数は数百万から数千万に達し，リアルタイムでの再計算が困難になる．

これに対し，いくつかの高速化手法が提案されている．

- **Prioritized Sweeping**: 価値の変化が大きかった状態の周辺のみを優先的に更新する手法.
- **GPU による並列化**: 各状態の更新計算は独立性が高いため，GPU（Graphics Processing Unit）を用いた並列計算と相性が良い．近年では CUDA を用いた実装により，数百万セルの更新を数ミリ秒で行う研究も報告されている．
- **階層化（Hierarchical MDP）**: 環境を粗いトポロジカルな領域に分割し，上位層で大まかな遷移を決定し，下位層で詳細な価値反復を行う手法.

深層強化学習との融合

近年では，Tamar らが提案した Value Iteration Networks (VIN)[**?**] のように，価値反復の計算プロセス自体を微分可能なニューラルネットワークの層として組み込む研究が盛んである．これにより，生のセンサ画像（入力）から，障害物のコストマップ（中間表現），そして最適な移動方向（出力）までを End-to-End で学習することが可能となる．これは，未知の環境における探索能力や，人混みの中でのナビゲーションといった複雑なタスクにおいて成果を上げている．

1.2.6 本研究の目的と構成

解決すべき課題

以上の従来研究の調査から，価値反復法は移動ロボットの大域経路計画において「大域的最適性の保証」「局所解の回避」「不確実性への対処」という優れた特性を持つことが確認された．しかしながら，実環境での運用を考えた場合，以下の課題が依然として未解決，あるいは改善の余地がある．

1. **動的環境へのリアルタイム追従性**: 環境の変化（ドアの開閉，荷物の移動など）に対し，全状態の価値関数を再計算するには時間がかかる．部分的な更新で整合性を保つ効率的なアルゴリズムが必要である．
2. **コスト関数の設計困難性**: 「安全性」や「社会的受容性（人への配慮）」といった抽象的な指標を，どのような報酬関数として設計するかは自明ではない．逆強化学習（Inverse Reinforcement Learning）などのアプローチもあるが，計算負荷が高い．
3. **3 次元空間への拡張**: ドローンや不整地走行ロボットへの適用を考えた際，状態空

間の次元爆発をどう抑えるかが課題となる.

1.3　研究目的

価値反復 ROS パッケージを用いたナビゲーションにおいて，走り出しまでの時間を短縮することで移動にかかる時間を短縮することを目的とする.

1.4　論文の構成

章では，問題設定と価値反復アルゴリズム，A*アルゴリズムについて述べ，章では，提案手法，章では，実験について述べる．章でまとめる.

第 2 章

移動ロボットの経路計画問題

2.1 扱う問題

本研究では，移動ロボットを平面上で自律移動させる問題を扱う．広く用いられている Nav2[?](ROS 2 の標準ナビゲーションパッケージ) が扱う問題と同様の問題である．ロボットは，行動を始めるタイミングで目的地の座標を与えられ，障害物との衝突を避けながらできる限り短い時間で目的地まで到達しなければならない．ロボットが移動を行う空間を環境と言い，図 2 にその環境の例を示す．環境には，世界座標系が 2 次元の直交座標系で設定されており，ロボットは，位置 (x, y) と，x 軸となす角 θ を向きとして持っており，これらをまとめてロボットの状態（位置と向き）$x = (x,\ y, \theta)$ 3 変数で表現される．目的地地点は図中の destination area のように XY 平面上の領域や，$xy\theta$ 空間内の領域として与えられる．環境中には，その位置は既知である固定障害物と移動障害物が存在するが，これらは VI パッケージの既存の機能で対処可能である．しかし，本研究では移動障害物の回避は陽には扱わない．

2.2 価値反復

VI パッケージの価値反復で計算される式は，ロボットがある位置・向き $\boldsymbol{x}$ にあるとき，そこから目的地までのコストを計算した状態価値関数

$$V : \mathcal{S} \to \mathbb{R} \tag{2.1}$$

である．ここで $\mathcal{S}$ は，$XY\theta$ 空間を格子状に離散化した離散状態 s の集合である．また，コストというのは，時間と，時間換算で与えるペナルティーを足した値である．ペナルティーは悪路に相当する s など，ロボットが入るとなんらかの問題のある状態に与えられる．

V は，価値反復の計算が進むと正解の値に近づいていく．このとき，V の値は図**??**(b) のように目的地の周りから収束していき，目的地から遠いところが最後に収束する．この計算は探索ではないが，「幅優先探索」に相当する処理となる．

V が収束すると，ロボットは V の値が小さくなるような行動を選択し続けることで，最適な経路を選択して移動できるようになる．また，V が完全に収束しなくても，ロボットのいる s から目的地に向かって値が単調減少していると，ロボットは目的地に向かうことができる．このような状態を，本稿では「経路が見つかる」と表現する．

1 章で述べたとおり，価値反復は状態価値関数 V を計算する．VI パッケージの実装では，$xy\theta$ 空間を格子状に離散化して作った離散状態の集合 $\mathcal{S}$ の要素 s に対し，ひとつひとつコスト $V(s)$ が計算される．

$V(s)$ には初期値として，s が目的地の領域に含まれる場合に 0，そうでないときは無限大（実装上は目的地まで 10 万秒など，非常に大きな値）が与えられる．価値反復は終端状態以外の $V(s)$ の値を，繰り返し計算で実際のコストまで値を小さくしていく．

完全に収束した V は最適状態価値関数 V^* と呼ばれ，V^* からは，最適な行動を求めることができる．ただし，ロボットが移動を開始するためには，V^* を厳密に計算する必要はない．現在地の周辺において，目的地に向かうための適切な勾配が V にできると，ロボットは目的地に向かう（V のコストを減らす）ように行動をとることができる．

しかしながら，この勾配の伝播は A*などの探索手法が経路を探すよりも遅い．価値反復は，探索の手法として見ると幅優先探索になっており，V の勾配は目的地の周辺からできはじめ，それがより遠い地点に伝播していく．またこの伝搬の計算は，ロボットが通らないであろう経路でも行われる．そのため，目的地までの経路が存在すれば，それが複雑に入り組んでいても必ず見つけることはできるが，ロボットは現在地に勾配が到達するまで，長く待たされることになる．価値反復アルゴリズムは，ベルマン方程式の形式に乗るように定式化することで，最適な状態価値関数を得られるアルゴリズムである．

2.3 経路探索アルゴリズム

2.3.1 Dijsktra 法

よく知られたグラフ探索アルゴリズムである．

2.3.2 A*アルゴリズム

Dijkstra 法を発展させたものであり，ゴールまでのコストを推定するヒューリスティック関数を追加したものである．

第 3 章

提案手法

3.1 価値反復と A*の組み合わせ

A*と価値反復を並列に計算し，A*の計算結果を価値反復に適用する．既存の価値反復と並行で A*探索を実行し，図**??**(c) のように，A*の見つけた経路沿いに V の値を書き換えるというものである．図のようにロボットが行動を開始する状態から目的地まで，V の値を少しずつ減らしていくように書き換えることで，価値反復が経路を見つける前に，ロボットを目的地に向かわせるようにする．

V の書き換えは，A*の見つけた経路沿いの各 s に対して，

$$V(s) \longleftarrow Kf(s) \tag{3.1}$$

という代入の式を用いて行う．K は定数であり，f は s から目的地までの経路上での道のりである．

この方法では，ロボットと目的地の間の環境が迷路のようになっていなければ，A*で見つかる経路がほぼ最適な経路となり，価値反復のみの場合よりもロボットが速やかに目的地に向かえるようになる．一方，迷路のような環境だと，たとえば A*で見つけた経路が遠回りで，そのあとで価値反復がよりよい経路を見つけると，目的地までの時間が増えてしまう可能性がある．

また，A*がほぼ最適な経路を見つけられる場合，価値反復は大域計画に対しては必須ではなくなる．しかし，A*の見つけた経路の最適化や，未知の障害物が現れた場合の V の修正や迂回先の V の計算に必要となる．

3.2 3D A*の適用

3D のほうがよくなるのではないか

本稿では，$xy\theta$ 空間での A*探索と価値反復を組み合わせる．ヒューリスティック関数 H については，[**?**] での平面上のユークリッド距離を計算するものから，

$$H(s) \triangleq W_1\sqrt{(x_\mathrm{c}-x_\mathrm{g})^2+(y_\mathrm{c}-y_\mathrm{g})^2}+W_2|\theta_\mathrm{c}-\theta_\mathrm{g}| \tag{3.2}$$

に変更する．ここで，(x_c, y_c, θ_c) は各離散状態の中心の座標，(x_g, y_g) は目的地の中心地点の xy 座標，$\theta_c - \theta_g$ は，(x_c, y_c, θ_c) から見た (x_g, y_g) の方角である．W_1, W_2 は定数である．W_1 と W_2 の比は，ロボットが一定時間あたりに移動できる距離と方向転換できる角度の比で決まる．

A*で算出した暫定経路の V への適用は，[**?**] と同じく式（3.1）で行う．ただし，V の値を変更する対象となる状態 s については，A*の算出した $xy\theta$ 空間中の経路をそのまま使うように変更した．また，f の値については，A*の算出した経路上での s とゴールまでの離散状態数とした．A*には，ヒューリスティック関数を「許容的」にすると最適な経路が見つかるという性質がある．ただし，許容的な場合には経路の探索に時間がかかる可能性がある．これについては次章の実験で調査する．

第 4 章

実装

4.1 A*の実装と価値反復への適用

4.1.1 A*の実装

2D A*の実装

A*の探索空間は，xy 平面であり，次章の実験のための実装として，ROS 2 版の VI パッケージ [?] に，A*探索を実行するスレッドを 1 つ追加した．この追加では，ike_nav[?] パッケージ内の ike_nav_planner を A* 探索のプログラムとして流用した．

ike_nav_planner はマップと現在地と目的地に対して A*で計算された経路を出力する．この経路上で，前節のように V の値を書き換えるコードを追加した．ただし出力される経路は XY 平面上のもので，θ 方向の情報がない．そのため，XY 平面上で同じ位置にある離散状態にはすべて同じ値を式 (3.1) で代入し，あとは価値反復で θ 方向の V の値を計算させることとした．この実装には改良の余地があり，$XY\theta$ 空間内での A*探索を用いるほうが，より時間短縮の効果が得られるものと考えられる．

3D A*の実装

A*の具体的な実装については，ike_nav[?] パッケージ内の A*探索ノード（ike_nav_planner）を 3 次元での探索に拡張し，ROS 2 版の VI パッケージ [?] に移植して利用した．

4.1.2 価値反復と A*の併用

4.2 提案手法の実機への適用

4.2.1 コンピュータの選定

実機に載せる都合上，サイズに気を使い，選定する必要がある．また，CPU を限界ギリギリまで長時間使用する都合上，排熱も考慮する必要がある．

第 5 章

実験

5.1 シミュレーター実験

シミュレーターで実験した．千葉工業大学津田沼キャンパスで得られた地図を用いたシミュレーションで，提案手法を評価する．図?に，この地図を示す．紙面横方向が 300m，縦方向が 200m の 6ha の環境の地図で，形式は，ROS のナビゲーションスタックで用いられる占有格子地図である．白色が障害物のない画素，黒色が障害物のある画素, 灰色が不明な画素を表す．この地図のパラメータは表**??**の通りである．

シミュレータ内で走行させるロボットは，図?に見られる差動二輪型のロボットである．実験に用いた計算機は，CPU として Ryzen 9 7945HX3D（16 コア 32 スレッド），DRAM として DDR5-4800 32GB が 2 枚（64GB）搭載されたものである．VI パッケージはマルチスレッド化されており，環境の全域で V を計算する大域計画器と，ロボットの周囲で V を計算する局所計画器に任意のスレッドの数を割り当てることができる．本稿の実験では，大域計画器に 8，局所計画器に 6 のスレッドを割り当てた．A*は別のプロセスで動作する．CPU から見ると，A*には 1 つのスレッドを割くこととなる．また，シミュレーションには Gazebo （ROS の標準的なシミュレータ）を使用するので，これにも計算機のリソースを使うこととなる．

5.2 実機実験

実機で実験した．

表 5.1　Configurations of the Map

map size	294.6[m]×199.95[m]
cell resolution	0.15[m]×0.15[m]
number of cells	2,615,346
number of free cells	165,076
the area of the free cells	3714.98[m^2]

第 6 章

結論

よくなりました.

参考文献

[国立 23] 国立社会保障・人口問題研究所. 日本の将来推計人口（令和 5 年推計）. 2023.