第 1 章

序論

1.1 背景

人間による操縦を必要とせず、自律的に活動するロボットに対する社会の期待が高まっている []。自律ロボットの活躍が期待される場所は多岐に渡り、従来のロボットのように工場のラインだけでなく、住宅や商業施設、あるいは被災地のような極限環境も含まれる。ロボットには様々な種類が存在するが、中でも自律移動ロボットは、病院内の搬送や空港の警備など、すでに社会実装されたものも多く存在する []。

これらの環境で移動ロボットが自律的に活動するためには、自己位置や周辺環境の把握と行動計画を行う必要がある。実世界で活動するロボットは、自身の位置を直接知ることはできない。したがって、搭載したセンサを用いて自身の位置や周辺環境を把握する必要がある。そして、ロボットは把握した自己位置や環境の情報を頼りに、目標地点へと移動するための行動を計画し動作する。

しかしながら、移動ロボットの活躍が求められている環境の多くは複雑で、不確かな要素が多く存在する。人間の生活する空間は、工業の組み立てラインのように不確かさや誤差がなるべく小さくなるように精密に構成されてはいない。また、センサが知覚できる情報には限りがありノイズが含まれているため、ロボットが環境の情報を完全に知覚することはできない。アクチュエータにおいても、制御ノイズや消耗のような要因から誤差が存在し、モデル化やアルゴリズムの近似による誤差も存在する。したがって、ロボットが実世界で自律的に活動するためには、これら多くの不確かさに対処していく必要がある。

不確かさを考慮するためには、確率論に基づく方法が有効である。ロボティクスにおいて、確率論を用いて不確かさに対処する試みは「確率ロボティクス (Probabilistic Robotics)」と呼ばれ、盛んに研究が行われている [**?**, **?**]。確率ロボティクスは、ロボットの知覚と制御に確率・統計を駆使することで、ロボット技術において避けて通ることができない不確かさを陽に表現することを可能としている。移動ロボットにおいて重要な自己

位置推定や行動計画についても、不確かさを考慮した様々なアルゴリズムが提案されている。

この確率論に基づく自己位置推定には、Baysian Filter が有効であると示されている。確率ロボティクスでは、ロボットの自己位置を決定論的な 1 つの位置ではなく、全ての位置からなる空間中における確率密度関数として表現する。Bayesian Filter による自己位置推定では、ベイズの定理に基づき、ロボットの自己位置を事後確率の分布として推定する。これらの分布は信念分布と呼ばれ、ロボットが主観的に得られる情報を基に自身の位置をどのように信じているかを表している。

現在、多くの移動ロボットにおいて、確率論に基づく自己位置推定が取り入れられている。中でもとくに、Monte Carlo Localization (MCL) という手法が多く利用されている [?][?]。これは、Bayesian Filter によるロボットの自己位置推定を実装する方法の 1 つである。MCL では、ロボットの信念分布を位置と姿勢からなる空間中に分布させた重みを持つ粒子（パーティクル) で近似的に表現する。これにより、ロボットの信念を複雑な確率分布として表現することを可能としている。同じく Bayesian Filter を理論的背景としたカルマンフィルタによる自己位置推定では、信念は正規分布でしか表現できない [?]。対して、MCL では一様や多峰性の分布などを表現することが可能であるため、複雑な環境で活動することが求められる移動ロボットに適している。ロボット開発用のミドルウェアである ROS では、LiDAR と専有格子地図を用いた MCL による自己位置推定のプログラムが、標準のナビゲーションパッケージとして用意されており、多くの開発者や研究者に利用されている [?, ?]。また、実環境において移動ロボットに自律移動を行わせる技術チャレンジであるつくばチャレンジでは、多くのチームが LiDAR と専有格子地図用いた MCL を用いている。[?] これらのことからも、MCL は実環境における自律移動ロボットの自己位置推定の手法として、有効であることが分かる。

現在、様々な自律移動ロボットにおいて確率的な自己位置推定手法が取り入れられている一方で、行動決定においては不確かさについて考慮されないことがある。一般的に多くの自律移動ロボットは、MCL やカルマンフィルタにより自身の位置を確率分布として推定する。そして、最も確率の高い位置を自身が存在する真の位置だと仮定し、行動計画を行う。しかし、このような方法では、ロボットの自己位置推定により得られた確率的な情報が、行動決定に反映されない。ロボットは信念分布が大きい場合も、分布が推定誤差が小さい場合も、同様の行動を行うことになる。

ロボット同様に、実世界で活動する我々にとっても不確かさは避けることができない問題だが、人間や一部の動物は、得られる情報が制限されている状況下でも、適切な行動を選択する。たとえば、人間は壁沿いを移動することで暗闇の中でも寝室に向かうことが可能であり、明るく視界に制限がないときとは異なる動作を行うことで、不確かさに対処している。このように状況に合わせた自律的な行動をロボットに行わせるためには、行動決

定に不確かさを考慮させることが必要である。

1.2 先行研究

不確かさに応じた柔軟な行動決定をロボットに行わせることができると、得られる情報が制限された状況下でもタスクを実行させることが可能となる。ロボット工学では、部分観測マルコフ決定過程 (POMDP) という枠組みで研究されているが、計算量の観点から最適な解を導くことができないと知られている [?]。そこで、近似的に POMDP を解く手法が複数提案されている。

暗闇の中、壁伝いに寝室へ向かうような行動については、Roy らによって提案された手法によりロボットへと実装されている [?]。この手法では、ロボットは事前に 4 次元の状態空間において行動計画が行われる。状態は、X-Y 平面上ロボットの位置と方向に、推定の不確かさの大きさを表すスカラー値である。計画された行動には、推定の不確かさを小さく保ちながらゴールへと向かう特性が現れた。この研究では、ロボットの自己位置推定は壁との距離を測定することで行われるため、ロボットは壁から大きく離れないようにゴールへと向かう動作を行った。この手法は、沿岸航法 (coastal navigation) と呼ばれる。

Littman らによって提案された Q_{MDP} value method[?] は、事前の行動計画と観測の不確かさを分けて扱った。Q-MDP 法では、観測の不確かさを考慮せずに、MDP 問題としてロボットの行動を事前に計画し、実際にロボットが行動を決定するときに、観測の不確かさを考慮する。事前に計画した行動と現在の信念の確率分布から、各行動に対する期待値計算を行い、最も良い行動を選択する。

Q_{MDP} value method は、Ueda らによって実際のロボットへと実装された [?]。RoboCup の四足歩行リーグのゴールキーパーのロボットへと適用され、有効性が示された。

Ueda によって提案された PFC 法では、移動ロボットがゴールを探索するような動作を可能にしている [?]。この手法では、Q_{MDP} value method の行動評価の式に変更を加えることで、ゴールに近いパーティクルが行動決定に及ぼす影響を大きくする。これにより、ロボットは自身の位置がほぼ分からない状況から、パーティクルにより近似された信念を徐々にゴールに流し込むような動作を行う。分布が徐々にゴールに吸い込まれるように移動していき、ロボットが分布の中にいる場合、ロボットもその流れに従いゴールへと到達する。また、ゴール付近のパーティクルが行動決定に及ぼす影響の大きさを変更した際に、Q_{MDP} value method で問題となっていた、ローカルミニマムによる行動の停滞が起こる頻度が低下することについても示されている [?]。

PFC 法は、障害物が存在しない空間において、自己位置推定が極めて不確かな移動ロ

ボットのナビゲーションに有効であることが、シミュレーション実験において示されている。しかし、障害物が存在する環境での動作については、有効性は確認されていない。PFC 法では、ゴール付近のパーティクルが行動決定に及ぼす影響を大きくする一方で、ゴールから遠いパーティクルや障害物内に存在するパーティクルは、軽視されることになる。つまり、ロボットが障害物を回避しようとする行動は反映されにくいという性質がある。一般的に、自律移動ロボットが活動することが求められている環境には、動的なものや静的なものを含め、多くの障害物が存在する。移動ロボットのナビゲーションでは、障害物の回避について考えることが必要であると言える。

1.3　問題提起

目的への繋ぎ方検討中。

1.4　目的

そこで本研究では、自己位置推定の不確かさを考慮した障害物の回避行動を自律移動ロボットに行わせるための手法を提案する。PFC 法に少し変更を加えることで、信念を近似するパーティクルクラウド全体が障害物を回避するような動作を生成する。これにより、自律移動ロボットに自己位置推定の不確かさを考慮した障害物回避の行動をとらせつつ、ゴールを探索するような動作を行わせることを可能にする。また、その有効性について通常の PFC 法と比較することで検証する。

1.5　本論文の構成

まず、第 1 章では、本研究の背景と先行研究を述べ、目的を設定した。第 2 章では、不確かさを考慮したロボットの行動決定について述べ、第 3 章では、ロボットの状態推定について述べる。第 4 章では、提案する手法について述べ、第 5 章では、提案手法を PFC 方と比較し評価する。最後に第 6 章で、本論文のまとめを述べる。

第 2 章

不確かさを考慮した行動決定

本章では、不確かさを考慮した行動決定について述べる。2.1 節では、ロボットの行動決定について述べる。2.2 節では、状態が既知のロボットの行動決定を扱う枠組みである「Markov 決定過程について述べる。2.3 節では、ロボットの状態が不確かにしか分からない状況での行動について扱う枠組みである「部分観測マルコフ決定過程」について述べる。2.4 節と 2.5 節では、本研究の先行研究となる、Q_{MDP} value method および PFC 法について述べる。

2.1 ロボットの行動決定

本節では、ロボットの行動決定について考える。本論文で扱うような移動ロボットの行動決定は、現在のロボットの位置から目的地までの最短経路を算出して移動する、「経路計画問題」として扱われる。ダイクストラ法や A*法、人工ポテンシャル法といった多くの手法が提案されており、現在でも研究が行われている。一般的にロボットの経路計画問題では、ロボットの観測や移動は決定論的なもとのして扱われ、経路の算出が行われる。

しかし、これまで述べたとおり、移動ロボットは多くの不確かさを有している。移動ロボットに最短の経路を算出して移動するだけでなく、これらの不確かさを考慮した上でさらに知的な動作を行わせる方法について考える必要がある。たとえば、人間であれば、最短ではあるが危険でゴールに到達できる可能性が低い道よりも、多少遠回りではあるが高確率で安全にゴールできる道を選択するような、「急がば回れ」が有効な場合も存在する。あるいは、自身の位置が正しく把握できていない場合に、安全を優先し全く動かないでいるよりも、多少の危険を冒しとりあえず行動してみてタスク達成を目指す方が有効であるような場合も考えられる。

このような知的な行動を、単純に経路計画問題として考えることは困難である。そこで本章では、ロボットの移動と経路計画について、より一般化した枠組みである（有限）マ

ルコフ決定過程および部分観測マルコフ決定過程について述べる。

2.2　マルコフ決定過程

本節では、移動ロボットの現在の自己位置 $\boldsymbol{x} \in \mathcal{X}$ が既知という前提での行動決定についての定式化を行う。このような問題は、マルコフ決定過程 (Markov decision process, MDP) という枠組みで議論される。時間やロボットの状態等を連続系ではなく離散系で考える場合は、有限マルコフ決定過程と呼ばれる。

2.2.1　状態と終端状態

3 章で述べたものと同様に、状態変数 x, y, θ からなる状態空間 $\mathcal{X}$ を定義する。ロボットは自身の現在の状態ベクトル $\boldsymbol{x} \in \mathcal{X}$ が分かっているものとし、観測の不確かさについては考慮しない。

ロボットのタスクには必ず終わりがあるとし、ロボットの状態が、事前に定めたある状態になったときをタスクの終了とする。このタスクが終了する状態は、終端状態と呼ばれ、終端状態の集合は $\mathcal{X}_f \subset \mathcal{X}$ と表現される。終端状態は、移動ロボットの経路計画問題における、ゴールや目標地点などのロボットが目指すべき望ましい状態だけでなく、陥りたくない状態も含まれる。

2.2.2　行動と状態遷移

ロボットは有限子の制御指令の中から、一つを選択することで動作する。MDP ではこの制御指令のことを行動と呼ぶ。ロボットの行動は m 種類存在し、行動の集合は、

$$\mathcal{A} = \{a_1, a_2, \ldots, a_m\} \tag{2.1}$$

と表現される。

時間は離散的に表現され、タスクの開始からは終わりまでは $0, 1, 2, \ldots, t_f \equiv T$ と定義される。ロボットが最初に行動を選択する時刻を $t = 0$ とし、行動が実行されるたびに $t = 1, 2, \ldots$ と次のステップへと進んでいく。ロボットが終端状態に入りタスクが終了する時刻を t_f とする。t_f は固定ではなく、タスク達成までにかかった時間により異なる。また、「時刻 $t-1$ の状態」、「時刻 t に遷移するために選択された行動」、「時刻 t の状態」をそれぞれ $\boldsymbol{x}_{t-1}, a_t, \boldsymbol{x}_t$ と表現する。しかし、MDP で扱うシステムは時不変であるため、多くの場合 t の具体的な値は重要でははない。したがて、今後はこれらをそれぞれ $\boldsymbol{x}, a, \boldsymbol{x}'$ と表記する。

ロボットの状態 $\boldsymbol{x}$ は、ある行動 a により状態 x' へと遷移する。状態の遷移にはノイズ

が含まれているものとし、状態と行動が同一の $(\boldsymbol{x}, a)$ であっても、事後状態 $\boldsymbol{x}^{'}$ は各試行ごとに異なる。ロボットの状態遷移は、マルコフ性を持ち、

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)(t = 1, 2, \ldots, t_f) \tag{2.2}$$

に従うものとする。また、ロボットが $\boldsymbol{x}$ から行動 a により $\boldsymbol{x}^{'}$ に遷移する確率を $p^a_{\boldsymbol{x}\boldsymbol{x}'}$ と表現する。この確率 $p^a_{\boldsymbol{x}\boldsymbol{x}'}$ が、$\boldsymbol{x}, a, \boldsymbol{x}^{'}$ のあらゆる組に対して既知であり、時不変であると仮定する。

2.2.3 報酬

状態 $\boldsymbol{x}$ で行動 a を行った場合に、状態が $\boldsymbol{x}^{'}$ に遷移した際に、その状態遷移に対して報酬を与える。たとえば、移動ロボットではこの報酬は、行動一回ごとに消費する電力や時間などを設定する。この報酬は、一つの実数値とし、$r^a_{\boldsymbol{x}\boldsymbol{x}'} \in \mathbb{R}$ と表現される。評価の基準が多次元的の場合も、一つの実数となるように一元化し評価を行う。

2.2.4 評価

ロボットがある状態 $\boldsymbol{x}$ からある終端状態 $\boldsymbol{x}_f \in \mathcal{X}_f$ に到達するまでの一連の状態遷移に対して評価を与える。評価は、

$$J(\boldsymbol{x}_{0:t_f}, a_{1:t_f}) = \sum_{t=1}^{t_f} r^a_{\boldsymbol{x}\boldsymbol{x}'} + V(\boldsymbol{x}_{t_f}) \tag{2.3}$$

で定義される評価値の大きさで行う。$r^a_{\boldsymbol{x}\boldsymbol{x}'}$ は、2.2.3 項で述べた報酬である。$V(\boldsymbol{x}_{t_f})$ は終端状態の価値とし、事前に目的に応じて与えられるものとする。ロボットが目指すゴールや目標地点とする場合、0 を設定する。逆に、ロボットがその状態になった場合タスクが失敗として終了するような、絶対に陥ってほしくない状態として設定する場合、非常に小さい値を設定する。

本論文では、この評価値 J を最大化する一連の状態遷移を、最適な制御とする。状態遷移にはノイズが伴うため、同じ姿勢 $\boldsymbol{x}_0$ から常に最適となるような行動をとったとしても、試行ごとに評価値 J は異なる。そのため、ある状態から終端状態までの一連の状態遷移に対する評価 J は、期待値として考える。

2.2.5 最適方策

ある状態における、J の期待値を最大化するための行動を与える関数

$$\Pi : \mathcal{X} \to \mathcal{A} \tag{2.4}$$

を定義する。この関数 Π は最適方策と呼ばれる。最適方策 Π が決まると、任意の状態 $\boldsymbol{x}$ でロボットが取るべき行動は、

$$a = \Pi(\boldsymbol{x}) \quad (\forall \boldsymbol{x} \in \mathcal{X}) \tag{2.5}$$

と自動的に決まる。

2.2.6　最適価値関数

また、ある状態に対して評価値の期待値 J を与える関数

$$V : \mathcal{X} \to \mathbb{R} \tag{2.6}$$

は、最適価値関数と呼ばれる。$V(\boldsymbol{x})$ は、$\boldsymbol{x}$ が初期の状態でも、タスクにおける途中の状態でも変わらない。終端状態の価値 $V(\boldsymbol{x}_{t_f})$ は、この最適価値関数に含まれる。

2.3　部分観測マルコフ決定過程

本節では、移動ロボットの現在の状態が不確かな中での行動決定について述べる。このような問題は、部分観測マルコフ決定過程 (partially observable Markov decision process, POMDP) という枠組みで研究が行われている。

POMDP が MDP と異なる点は、ロボットが真の状態が分かっていないという点にある。ロボットは MDP 同様、評価値 J を最小化するように終端状態にたどり着くことを目的とする。しかし、ロボットの姿勢 $\boldsymbol{x}$ が未知であるため、最適方策 $\Pi(\boldsymbol{x})$ を使用することができない。したがって、$\boldsymbol{x}$ の代わりにその場で得られる情報をもとに行動を決定する必要がある。ロボットがこれまで行ってきた行動の履歴 $a_{1:t}$、それまでに得た観測の情報 $z_{1:t}$、およびそれまでに得た報酬の履歴 $r_{1:t}$ から決定される方策は

$$a_{t+1} = \Pi_{\mathrm{POMDP}}(a_{1:t}, \boldsymbol{z}_{1:t}, r_{1:t}) \tag{2.7}$$

と定義される。

ロボットが行動を決定するとき、その時点での状態推定の不確かさを考慮したい場合、関数

$$a_{t+1} = \Pi_{\mathrm{b}}(b_t) \tag{2.8}$$

を考えればよいことになる。この関数は、ロボットの姿勢を状態とするのではなく、ロボットの信念分布自体をひとつの状態としてみなし、とるべき行動をロボットの現在の信念分布から決定する。状態としてみなされる信念分布は、信念状態と呼ばれる。

このように信念の分布をひとつの状態とみなすことで、POMDP の問題を MDP の問題として考えることができる。信念状態を用いることで POMDP を MDP として考える

方法は、belief MDP と呼ばれる [?]。しかし、信念状態の数は非常に膨大であり、最適方策を得ることは通常不可能とされている。そこで、最適ではなくとも、全く不確かさを考慮しないときよりは優れた行動を得る、という手法を考えることが有効となる。

2.4 Q-MDP

本節では、Q_{MDP} value method について述べる。この手法は、Littman らにより [?] で提案され、Ueda らによって [] でロボットに適用された。

Q-MDP 法は、状態が既知の問題である MDP で得た最適行動価値関数

2.5 Probabilistic Flow Control

第 3 章

移動ロボットの自己位置推定

本章では、Bayesian Filter による移動ロボットの自己位置推定について定式化する。また、Bayesian Filter の実装方法の 1 つである Monte Carlo Localization (以下 MCL) について述べる。本論文においても、移動ロボットの自己位置推定には MCL が利用されるものとする。

3.1 Bayesian Filter による自己位置推定

3.2 Monte Carlo Localization (MCL)

Monte Carlo Localization (MCL) は、Bayesian Filter を理論的背景としたロボットの自己位置推定手法の実装方法の 1 つである。MCL では、表現したい任意の空間上 $\mathcal{X}$ 上に存在する確率密度関数を、空間上に散布した標本により表現することで、ロボットの姿勢を確率的に推定する。この標本はパーティクル（粒子）と呼ばれ、ロボットの信念分布を表す確率密度関数 $b_t(\boldsymbol{x})$ を近似するように配置される。パーティクルはそれぞれパラメータとしてロボットと同次元の姿勢 $\boldsymbol{x}$ と重み w の情報を持っており、N 個のパーティクルの集合は

$$\Xi_t = \xi_t^{(i)} = (\boldsymbol{x}_t^{(i)}, w_t^{(i)} | i = 1, 2, \ldots, N) \tag{3.1}$$

のように定義される。基本的に全パーティクルの重みの合計は、常に

$$\sum_{i=1}^{N} w_t^{(i)} = 1 \tag{3.2}$$

を保つように実装される。

ロボットの状態を表すパラメータについては様々存在するが、自己位置推定ではロボットの姿勢（位置と向き）を推定する。本論文では、二次元平面を低速で移動する対抗二輪

型の移動ロボットを想定し、ロボットの二次元平面上における位置 (x, y) と向き θ をあわせた 3 次元の状態 $\boldsymbol{x} = (x, y, \theta)$ の推定について扱う。したがって、ロボットの姿勢と同様に、パーティクルの姿勢はそれぞれ $\boldsymbol{x}^{(i)} = (x^{(i)}, y^{(i)}, \theta^{(i)})$ となる。

MCL のアルゴリズムは、主に次の 4 ステップからなる。2 から 4 を繰り返し行うことで逐次ロボットの姿勢を推定する。

1. 初期化
2. 動作による更新
3. 計測による更新
4. リサンプリング

3.2.1　初期化

初期化のステップでは、パーティクルの初期化を行う。N 個のパーティクルの初期姿勢 $\boldsymbol{x}^{(i)}$ を決定し、重み $w^{(i)}$ を $\frac{1}{N}$ とする。パーティクルのばらまき方は、ロボットの初期の信念分布に従うように行う。一般的には、以下のようなパーティクル初期姿勢の決定方法が用いられる。

ロボットを人間が自由に配置する場合など、初期姿勢が予め分かっている場合は、ロボットの信念分布をロボットが存在すると考えられる位置を平均とした正規分布と考え、パーティクルの初期姿勢 $\boldsymbol{x}^{(i)}$ を平均 μ、分散 Σ の正規分布に従うように決定する。本論文における移動ロボットの自己位置推定では、状態 $\boldsymbol{x}^{(i)}$ が位置と向きからなる 3 次元のため、μ と Σ はそれぞれ 3 次元ベクトル、3×3 の分散共分散行列として扱う。多くの場合 μ と Σ はヒューリスティックに決定する。

一方、ロボットの初期位置が完全に不明なとき、ロボットの初期信念分布は状態空間 $\mathcal{X}$ 全体の一様分布と考えられる。その場合は、パーティクルの姿勢は一様分布に従うように配置する。

3.2.2　動作による更新

動作による更新のステップでは、Bayesian Filter における予測ステップを、各パーティクルに対して行う。移動ロボットの状態遷移にはノイズが伴うため、同様の制御入力でも試行ごとに異なる状態遷移結果となる。したがって、ロボットが動作すると信念分布が広がることになる。この広がった信念分布を表現するように、パーティクルの分布を更新する。各パーティクルの状態をロボットの動作モデルロボットの状態遷移確率

$p(\boldsymbol{x}|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ とすると、動作後の各パーティクルの姿勢は

$$\boldsymbol{x}_t^{(i)} \sim p(\boldsymbol{x}|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \tag{3.3}$$

のように更新される。

3.2.3 計測による更新

計測による更新のステップでは、Bayesian Filter における計測更新ステップを行う。センサにより得られた情報が、各パーティクルの状態で得られる確率を計算し、重み $w^{(i)}$ を更新する。重みの更新はベイズの定理を用いて

$$w_t^{(i)} = q(\boldsymbol{z}_t|\boldsymbol{x}_t^{(i)})\hat{w}_t^{(i)} \tag{3.4}$$

のように計算される。尤度関数 $q(z_t|\boldsymbol{x}_t^{(i)})$ は、使用するセンサの観測モデルをもとに決定する。

現在、移動ロボットの自己位置推定のために、LiDAR と呼ばれる光センサによる測距センサや、RGB-D カメラが広く利用されている。これらのセンサを自己位置推定位に用いるロボットは、条件が揃えば非常に正確な推定を行うことが可能である。しかし、本論文では移動ロボットの状態推定が不確かなときに有効となるような行動決定の手法を提案することを目的としている。したがって、移動ロボットが曖昧にしか状態推定を行えない状況を想定するために、ロボットはこれらのセンサを搭載していないものとする。

代わりに、ロボットはタスクの終了を検知することができるものとする。この仮定は、4 章で述べる提案手法、およびその先行研究となる PFC 法で必要となる。ロボットは、自身がゴールに到達したか否かを各ステップごとに知ることができ、その情報を計測による更新に用いることができるものとする。

3.2.4 リサンプリング

リサンプリングステップでは、パーティクルを再配置し重みを均一にする。再配置するパーティクルは、重みに従った確率で選択される。重みの小さいパーティクルを間引き、重みの大きいパーティクルの位置に多くのパーティクルを配置する。この操作により、一箇所に重みが偏り続け推定の精度が下がることを防ぐ。

パーティクルの選択（サンプリング）にはいくつかの手法が存在する。本論文では、系統抽出法によるリサンプリングを用いる。

第 4 章

不確かさを考慮した障害物の回避動作

本章では、不確かさを考慮した障害物の回避動作を生成する手法について述べる。4.1 節では、提案手法の概要について簡単に述べる。4.2 節では、本論文において回避を行う障害物の定義を述べる。4.3 節では、観測の不確かさがない前提で価値関数を計算する方法について述べる。4.4 節では、提案手法において PFC 法から変更する部分について述べる。

4.1 提案手法の概要

本論文では、不確かさを考慮した障害物の回避行動を生成する。MCL により自己位置推定を行う移動ロボットの行動決定に、パーティクルの分布をそのまま利用し、推定の不確かさを考慮した行動を行わせる。本手法では、ロボットの信念分布を近似するパーティクルの分布全体が、障害物外を移動するようにロボットを動作させる。これにより、ロボットの自己位置推定が正しく行われており、真の姿勢がパーティクルの分布内に存在する場合、ロボットが障害物に侵入するのを防ぐことができる。

2.5 節で述べた PFC 法に変更を加えることで、この動作を実現する。オフラインフェーズでは、通常の PFC 法同様に、観測に関する不確かさがない前提で MDP 問題を解き、最適価値関数を計算する。変更は、実際にロボットの行動決定を行うオンラインフェーズでの行動評価の式に行う。PFC 方は、全パーティクルが行動決定に与える影響を同じにするのではなく、ゴール付近のパーティクル影響を強めている。この影響度を、障害物付近のパーティクルでも強くする。次ステップの行動で障害物内に侵入する可能性があるパーティクルが、行動決定に与える影響度を一気に大きくすることで、回避動作を行わせる。これにより、パーティクルの分布全体が障害物を避けるような動作を行わせるとも

に、PFC 法の利点である、ゴール付近で分布をゴールに流し込むような探索動作を行うことが可能となる。

4.2　本論文における障害物

本節では、本論文において、ロボットが回避する対象とする障害物について述べる。まず、障害物の種類を述べる。次に、本論文で回避する対象の障害物として扱うものについて述べる。

4.2.1　障害物の種類

自律移動ロボットが活動することが求められる環境には、様々な種類の障害物が存在する。ここで言う障害物は、ロボットがぶつかったり侵入したりすることを避けたい場所全般を指すものとする。壁や段差など、ほとんどの場合動かないような静的な障害物もあれば、人間やロボットなど、常に動き続けている動的な障害物もある。あるいは、ドアや椅子など、多くの場合止まっているが、人間やロボットの行動により変位するような、準動的な障害物も存在する。

障害物は、ロボットが直接観測できるものとできないものにも分けて考えられる。ロボットに搭載したセンサにより検出可能なものは、SLAM により地図を作成する際に地図に反映することができる。現在主流である 2D LiDAR を搭載したロボットについては、壁や棚については検出が可能だが、溝や段差は直接検出することができない。

4.2.2　本論文において回避対象とする障害物

本論文においては、地図に記載されている動かない障害物を回避することを目指す。ロボットに搭載しているセンサにより、動作の際に直接観測できる必要はないものとする。しかし、地図に載っていることが前提となるため、搭載したセンサにより観測ができないものについては、SLAM により地図を生成したあとに、人間の手によって書き加えられている必要がある。

4.3　価値反復による価値関数の計算

本節では、本手法および先行研究となる PFC 法のオフラインフェーズである、価値関数の計算について述べる。ロボットの観測の不確かさを考慮せず、状態が既知であるという前提のもと、MDP を解くことで価値関数を計算する。

4.3.1 状態空間の離散化

計算機により価値関数 V を計算するために、状態空間の離散化が必要となる。状態空間 $\mathcal{X}$ を有限個の離散状態の集合 $\mathcal{S}$ で表現し、価値関数 V は、この各離散状態に対する関数 $V(\boldsymbol{s})$ とする。このように、離散化された状態空間における制御問題は、有限マルコフ決定過程 (finite Markov decision process, finite MDP) と呼ばれる。S に対する最適価値関数 V は、

$$V(s) = \max_a \sum_{s' \in \mathcal{S}} p^a_{ss'} \left[r^a_{ss'} + V(s') \right] \tag{4.1}$$

で計算することができ、この式はベルマン方程式と呼ばれる。

4.3.2 価値反復

計算をすべての離散状態 $\mathcal{S}$ に対して、値が収束するまで繰り返し行っていくことで、最適価値関数 $V(s)$ を求める。この方法は、動的計画法のひとつである価値反復と呼ばれる。この価値反復により、全離散状態 $\mathcal{S}$ に対する価値関数 $V(s)$ を事前に計算しておく。

4.4 提案する手法

本節では、パーティクル全体が障害物を回避するための方法について述べる。まず、行動決定の際に行動を評価する式の変更についてと、追加するパラメータについて述べる。次に、パラメータの値の決定方法について述べる。

4.4.1 回避用パラメータの導入

本手法では、各パーティクルが行動決定に与える影響を動的に変更させるために、パラメータを追加で持たせる。パーティクル $\xi^{(i)}$ は、 パラメータとして状態 $\boldsymbol{x}^{(i)}$ と重み $w^{(i)}$ を持っている。これに、行動決定に与える影響を変更するためのパラメータ $m^{(i)}_{\mathrm{avoid}}$ を追加で持たせる。変更後のパーティクルは、

$$\Xi_t = \xi^{(i)}_t = (\boldsymbol{x}^{(i)}_t, w^{(i)}_t, m^{(i)}_{\mathrm{avoid}} | i = 1, 2, \ldots, N) \tag{4.2}$$

のように定義される。なお、パラメータ $m^{(i)}_{\mathrm{avoid}}$ は、行動決定の際に用いるものであり、自己位置推定においては使用されない。

4.4.2 行動評価式の変更

4.4.3 回避用パラメータの増加

4.4.4 回避用パラメータの減衰

第 5 章

評価

5.1 **評価方法**

5.2 **シミュレータ**

5.3 **実験条件**

5.4 **結果**

5.5 **考察**

第 6 章

結論