2019 年度 修士論文

移動ロボットのための
状態推定の不確かさを考慮した
障害物回避行動の生成

鈴木勇矢
Chiba Institute of Technology

2020 年 2 月 4 日

謝辞

本論文は, 著者が千葉工業大学大学院工学研究科未来ロボティクス専攻博士前期課程において行った研究をまとめたものです. 本研究を進めるにあたって, ご指導を頂いた指導教員の千葉工業大学大学院工学研究科未来ロボティクス専攻 上田隆一准教授に, 感謝いたします. また, 副査としてご助言を頂いた, 米田完教授, ならびに林原靖男教授に感謝いたします. 最後に, 研究などでお世話になりました同じ研究室の皆様, および未来ロボティクス専攻の先生方に感謝いたします.

目次

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の用途で, 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 [Thrun 05, Seba 07]. 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, ベイズフィルタ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一つの位置ではなく, 空間中における確率分布として表現する. ベイズフィルタによる自己位置推定では, ベイズの定理に基づき, 確率分布を逐次更新していく. この分布は信念分布と呼ばれ, ロボットが得られる情報をもとに, 主観的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 [Dellaert 99, Fox 99]. これは, ベイズフィルタ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. 同じくベイズフィルタ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 [Kalman 60]. 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 [Quigley 09, Open Source Robotics Foundatio]. 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. [夏迫 16] 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

アルゴリズムで不確かさを考慮す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 [Kaelbling 98]. 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 [Roy 99]. 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を足した, 4 つで表される. 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[Littman 95] 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, マルコフ決定過程の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 [Ueda 03]. 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 [Ueda 15]. 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 [Ueda 18].

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　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 法に変更を加えることで, 信念を近似するパーティクル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.

1.4　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, ロボットの状態推定について述べる. 第 3 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 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

MCL による移動ロボットの自己位置推定

本章では, ベイズフィルタによる移動ロボットの自己位置推定手法の一つである, Monte Carlo localization (以下 MCL) について述べる. MCL の概要について述べ, 各処理の処理内容を説明する. 本論文において, 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.

2.1 MCL の概要

MCL は, ベイズフィルタを理論的背景としたロボットの自己位置推定手法の 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. 各パーティクルは変数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

$$\Xi = \{\xi^{(i)} = (\boldsymbol{x}^{(i)}, w^{(i)}) | i = 1, 2, \ldots, N\} \tag{2.1}$$

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

$$\sum_{i=1}^{N} w_t^{(i)} = 1 \tag{2.2}$$

を保つように実装される.

ロボットの状態を表す変数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向二輪型の移動

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

2.2 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 $1/N$ 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, $\boldsymbol{\mu}$ 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 $\boldsymbol{\mu}$ 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.

2.3 動作による更新

動作による更新のステップでは, ベイズフィルタ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

$$\boldsymbol{x}_t^{(i)} \sim p(\boldsymbol{x}|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \tag{2.3}$$

のように更新される.

2.4 計測による更新

計測による更新のステップでは, ベイズフィルタ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

$$w_t^{(i)} = q(\boldsymbol{z}_t|\boldsymbol{x}_t^{(i)})\hat{w}_t^{(i)} \tag{2.4}$$

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.

2.5 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 3 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. 3.1 節では, ロボットの行動決定について述べる. 3.2 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「マルコフ決定過程について述べる. 3.3 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. 3.4 節と 3.5 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.

3.1 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, 経路が算出される.

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.

3.2　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.

3.2.1　状態と終端状態

2 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.

3.2.2　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,

$$\mathcal{A} = \{a_1, a_2, \ldots, a_m\} \tag{3.1}$$

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 $\boldsymbol{x}'$ 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,

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)(t = 1, 2, \ldots, t_f) \tag{3.2}$$

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.

3.2.3 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.

3.2.4 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,

$$J(\boldsymbol{x}_{0:t_f}, a_{1:t_f}) = \sum_{t=1}^{t_f} r^a_{\boldsymbol{x}\boldsymbol{x}'} + V(\boldsymbol{x}_{t_f}) \tag{3.3}$$

で定義される評価値の大きさで行う. $r^a_{\boldsymbol{x}\boldsymbol{x}'}$ は, 3.2.3 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.

3.2.5 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

$$\Pi : \mathcal{X} \to \mathcal{A} \tag{3.4}$$

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,

$$a = \Pi(\boldsymbol{x}) \quad (\forall \boldsymbol{x} \in \mathcal{X}) \tag{3.5}$$

と自動的に決まる.

3.2.6 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

$$V : \mathcal{X} \to \mathbb{R} \tag{3.6}$$

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.

3.3 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

$$a_{t+1} = \Pi_{\mathrm{POMDP}}(a_{1:t}, \boldsymbol{z}_{1:t}, r_{1:t}) \tag{3.7}$$

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

$$a_{t+1} = \Pi_{\mathrm{b}}(b_t) \tag{3.8}$$

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 [Kaelbling 98]. 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近似的に得る, という手法を考えることが有効となる.

3.4 Q-MDP

本節では, Q-MDP について述べる. この手法は, Littman らにより [Littman 95] 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, 観測の不確かさを考慮せずに, MDP 問題として解かれた最適価値関数と, 信念分布から行動を決定する. 各行動に対する期待値計算を行い, 最も期待値が高くなるような行動を選択する. 信念分布がパーティクルにより近似されているとき, 期待値は,

$$Q_{\mathrm{PFC}}(a,b) = \sum_{i=1}^{N} w^{(i)} \left[r^{a}_{s^{(i)}s'} + V(s') \right] \tag{3.9}$$

のように計算される. この期待値を全行動 $a \in \mathcal{A}$ に対して計算し, 最大となる行動を

$$\Pi_{Q_{\mathrm{MDP}}}(b) = \arg\max_{a \in \mathcal{A}} Q_{\mathrm{MDP}}(a,b) \tag{3.10}$$

のように決定する.

Q-MDP 法は, Ueda らによって 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, 有効性が示されている [Ueda 03]. しかし, 本論文で扱うような移動ロボットでは, ローカルミニマムによる行動の停滞が頻繁に起こる Q-MDP 法は, あまり有効な方法とは言えない. また, 1.2 節で述べた沿岸航法の探索動作のような,

3.5 Probabilistic Flow Control

本節では, PFC(Probabilistic Flow Control) 法について述べる. この手法は, Ueda により [Ueda 15] で提案されている.

PFC 法では, 行動評価の式を

$$Q_{\mathrm{PFC}}(a,b) = \sum_{i=1}^{N} \frac{w^{(i)}}{V(\boldsymbol{x}^{(i)})^{m}} \left[r^{a}_{\boldsymbol{x}^{(i)}\boldsymbol{x}'} + V(\boldsymbol{x}') \right] \tag{3.11}$$

のように変更することで, ゴールに近いパーティクルが行動決定に与える影響を大きくし, ロボットのゴール探索動作を生成した. ここで, m は [Ueda 18] で導入されたパラメータで, ゴールに近いパーティクルの行動決定への影響度を決定する. ロボットはパーティク

ル分布を徐々にゴールへと流し込むように動作し, 分布の中にロボットが存在する場合, ロボットの流れに従いゴールへと到達する. この手法は, 知的な探索動作を生成し, かつ Q-MDP 法で問題となっていたローカルミニマムによる行動停滞の頻度の低下も確認されており, 移動ロボットのナビゲーションタスクに対し有効である.

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.

3.5 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 法は, 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, 変数値の更新方法について述べる.

4.4.1 回避用変数の導入

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, 変数を追加で持たせる. パーティクル $\xi^{(i)}$ は, 変数として状態 $\boldsymbol{x}^{(i)}$ 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,

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

のように定義される. $m^{(i)}_{\mathrm{avoid}}$ の値は, $(m_{\mathrm{avoid\_min}} \leq m^{(i)}_{\mathrm{avoid}} \leq m_{\mathrm{avoid\_max}})$ の範囲で変動する. なお, 変数 $m^{(i)}_{\mathrm{avoid}}$ 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

PFC 法において, 行動を評価する式 3.11 は, 離散の状態空間 $\mathcal{S}$ では

$$Q_{\mathrm{PFC}}(a,b) = \sum_{i=1}^{N} \frac{w^{(i)}}{V(s^{(i)})^m} \left[r^a_{s^{(i)}s'} + V(s') \right] \tag{4.3}$$

のように定義される. 本手法では, この式を

$$Q_{\mathrm{PFC}}(a,b) = \sum_{i=1}^{N} \frac{w^{(i)}}{V(s^{(i)})^m} \left[r^a_{s^{(i)}s'} + V(s') \right]^{m^{(i)}_{\mathrm{avoid}}} \tag{4.4}$$

のように変更することで, $m^{(i)\mathrm{avoid}}$ の値により, パーティクルが行動に与える影響の大きさが変わるようにする.

4.4.3　回避用変数値の更新

$m^{(i)}_{\mathrm{avoid}}$ の更新方法について述べる. $m^{(i)}_{\mathrm{avoid}}$ は, タスクの実行中, 状況に応じて

$$m^{(i)}_{\mathrm{avoid}} = \begin{cases} m_{\mathrm{avoid\_max}} & (r^a_{s^{(i)}s'} < -\Delta t) \\ m^{(i)}_{\mathrm{avoid}} & (\mathrm{otherwise}) \end{cases} \tag{4.5}$$

のように更新する. 行動評価のとき, あるパーティクルが行動 a によって障害物に入る場合, 値を大きく増加させる. 次ステップで障害物に入るという判断は, 報酬 $r^a_{s^{(i)}s'}$ の値から判定することができる. **??**節で述べたとおり, 本論文では, 報酬を式 (**??**) の用に設定している. 障害物外への状態遷移では, 報酬は $-\Delta t$ となり, 障害物内への状態遷移では, それ以下となる. したがって, $m^{(i)\mathrm{avoid}}$ は, 報酬が $-\Delta t$ よりも小さくなる障害物内への状態遷移のときに, $m_{\mathrm{avoid\_max}}$ へと更新し, 障害物外への状態遷移では, $m^{(i)}_{\mathrm{avoid}}$ の値は更新しない.

4.4.4　回避用変数値の減衰

更新した $m^{(i)}_{\mathrm{avoid}}$ を, 時間とともに減少させる処理を加える. 値の更新を 4.4.3 項での処理のみで行った場合, ロボットの行動がループする可能性が考えられる. 一度障害物へと入りそうになったパーティクルと, それ以外のパーティクルで, 行動決定に与える影響に差ができる状況を継続させる必要がある.

本論文では, 時間とともに線形に減少していく方法をとる. 一度 $m_{\mathrm{avoid\_max}}$ まで増加した $m^{(i)}_{\mathrm{avoid}}$ は, 10 秒後に $m_{\mathrm{avoid\_min}}$ になるように減衰させていく.

第 5 章

シミュレーションによる評価

本章では, 提案した手法の評価を行う. 5.1 節では, 評価方法について述べる. 5.2 節では, シミュレーション実験の条件について述べる. 5.3 節では, シミュレーション実験の結果を述べる. **??**節では, 考察について述べる.

5.1 評価方法

本節では, 評価方法について述べる. まず, 評価を行うにあたって, 本手法により目指している動作について述べる. そして, その目指している動作を達成しているかを評価するときの基準について述べる.

5.1.1 目指している動作

本論文において, 移動ロボットに行わせたい動作は, 状態推定の不確かさを考慮した障害物回避の動作である. 通常行われるような, 最も確率の高い姿勢を真の姿勢と仮定して行われる行動計画では, 確率的な推定の情報が行動に生かされない. そこで, 信念分布内のどこかにロボットが存在する可能性を考慮し, 信念分布を近似するパーティクル全体が, 障害物内に入らないような行動を行わせることを目指している.

また, PFC 法のゴール探索動作が行われることも求められる. 分布全体を徐々にゴールになぞるように流し込み, 分布内に存在するロボットがいずれゴールへと到達する. これにより, 信念分布がゴール範囲よりも大きい場合でも, ロボットがゴールすることを可能にする.

信念分布が大きいときに, この 2 つの動作をそれぞれ状況にあわせて行うことを目指す. 周囲に障害物があるときには, 分布全体が障害物を避けるような動作を行わせる. そして, ゴール周辺では, 停滞することなくゴールへと到達するするために, 分布をゴールへと流

し込む探索動作を行わせる.

5.1.2　評価方法

1 回のナビゲーションタスクにおいて, 障害物の回避とゴールの探索動作がそれぞれ行えているかを評価する. タスクは, ロボットがゴールへと到達したときに成功とみなす. 逆に, ロボットが障害物内に侵入した時点で, そのタスクは失敗とする. ロボットがゴールへ到達するのに 300 秒以上かかった場合も, 同様に失敗とする.

ロボットは初期姿勢の配置と動作モデルは, それぞれノイズを有している. ロボットの初期姿勢は, 正規分布に従うばらつきを有しており, パーティクルの初期分布も, 同様の正規分布に従い配置されるものとする.

また, ロボットの状態推定の不確かさが大きい状況を維持するために, ロボットが観測により得られる情報が非常に制限されているものとする. 自身の姿勢推定に用いることができる情報は, ゴールしているか否かの情報のみである. これは, PFC 法による行動を行うために必要となる観測情報である. そのほかの距離センサやランドマークによる観測は行えないものとする.

そのために, ロボットがナビゲーションにおいて, 状態推定を考慮した障害物回避とゴールの探索を行えているかを,

- タスクの成功率
- タスク成功時の平均時間
- 障害物内を移動したパーティクル数×時間 の平均

の 3 つを基準に評価する. この基準をもとに, 提案手法を, 通常の PFC 法, Q-MDP 法, ロボットの真の姿勢による最適方策, そしてパーティクルの平均姿勢による最適方策の 4 つと比較する. 各手法それぞれの試行を 100 回行うことで評価する.

5.2　シミュレーション実験の条件

本節では, シミュレーションによる実験の条件について述べる. まず, 評価を行うための環境について述べる. そして, ロボットの初期姿勢や状態遷移確率, 行動の種類について述べ, また, MCL やその他パラメータについて述べる.

5.2.1　環境

二次元のシミュレータを用いて, 評価を行う. 評価は, 図 5.1 に示すように, 一つの障害物を有する, 幅が 10[m] の正方形の空間で行う. 灰色で示された領域は, 障害物を意味す

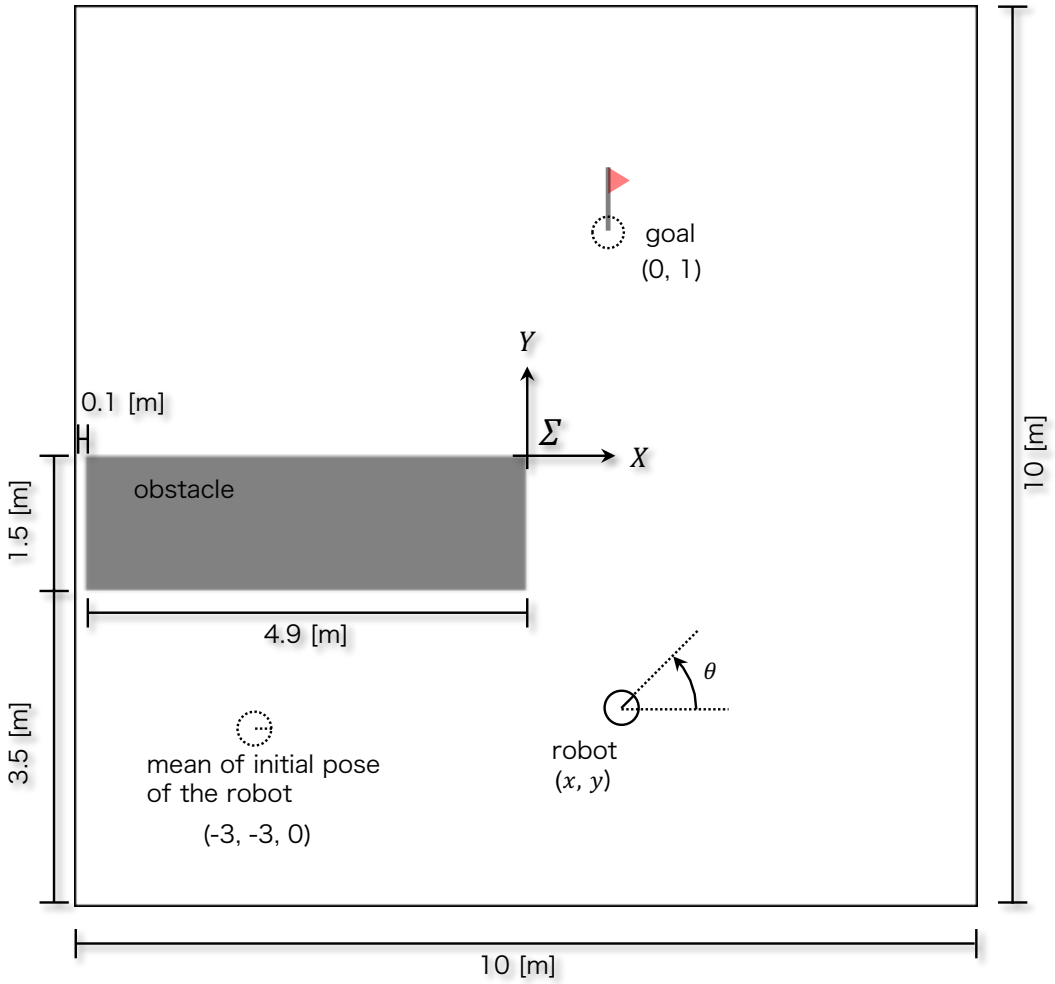

図 5.1　Environment map with one obstacle.

る．また, 環境の中央を原点とする $X-Y$ 座標系 Σ を定義する．X 軸と Y 軸はそれぞれ環境の縁と並行に設定されている．ゴールは $(0,1)$ の位置を中心とした, 半径 0.15[m] の範囲内とする．

5.2.2　タスク内容の成功条件

ロボットに, ゴールまでのナビゲーションタスクを行わせる．図 5.2 に示す実線矢印のように, 障害物を回避しゴールへ入ることができたらタスク成功とする．逆に, 破線矢印のようにロボットが障害物に衝突した場合, タスク失敗とする．また, 300[s] の間ゴールへ入ることができなかった場合も, 同様にタスク失敗とする．

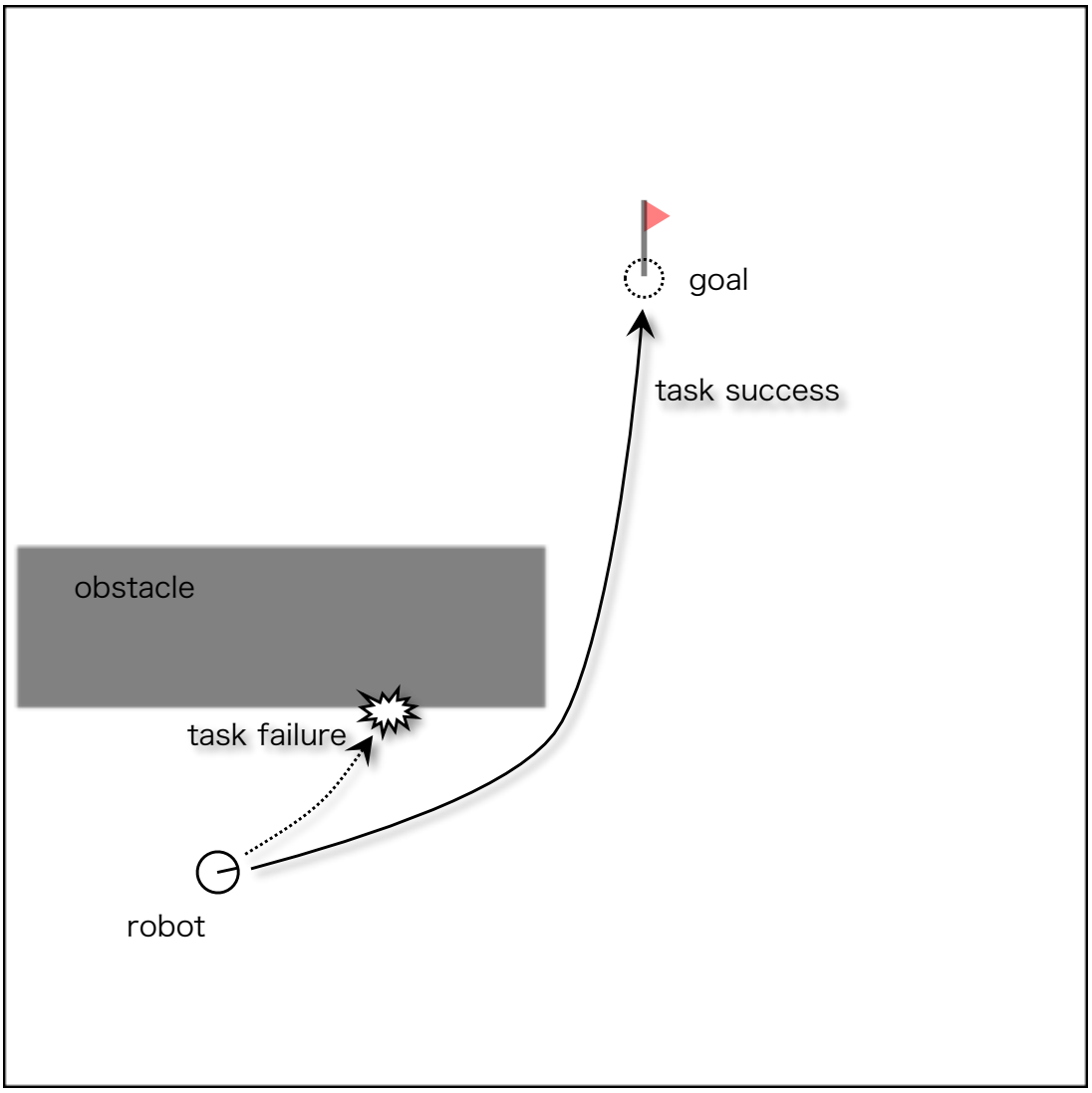

図 5.2　Task success and failure examples.

5.2.3　ロボットの行動と初期姿勢

ロボットの行動は, $\mathcal{A} = fw, ccw, cw$ の 3 つから選択されるものとする. 行動はそれぞれ,

- fw: $(0.2 + 0.01\sigma)$[m/s] の前進
- ccw: $(1 + 0.01\sigma)$[rad/s] のその場旋回
- cw: $(-1 + 0.01\sigma)$[rad/s] のその場旋回

を意味する. ここで, σ は正規分布 $\mathcal{N}(0, 1)$ から生成されるノイズであり, $\sigma \sim \mathcal{N}(0, 1)$ の

ように選択される. ロボットは, この運動モデルを知っていると仮定する.

ロボットの初期姿勢は,

$$(x, y, \theta) = (-3 + 0.3\sigma, -3 + 0.3\sigma, 0.03\sigma) \tag{5.1}$$

のように配置され, 各試行ごとに正規分布に従いばらつく.

5.2.4 価値関数の計算

価値関数を計算する状態空間 $\mathcal{X}$ は, 二次元の位置 (x, y) と方向 θ の 3 状態からなる. 計算のための離散化は, 位置についての離散状態の幅を 0.05[m] とし, 方向については 10[deg] とする. 図 5.1 に示した縦横の幅が 10[m] の空間を離散化すると, 離散状態は $200 \times 200 \times 36$ となる. この, 合計 $1,440,000$ 離散状態の価値関数を事前に計算する. また, ゴール内に含まれている状態 s を終端状態 $\mathcal{S}_f$ とする.

状態遷移の報酬について設定する. このタスクでは, ゴールまで早く到達することと, 障害物を避ける移動が良い移動となるため, 経過時間と障害物への侵入に負の報酬（ペナルティー, コスト）を与えることにする. まず, 時間に関する報酬を,

$$r_{\mathrm{time}}{}^{a}_{\boldsymbol{x}\boldsymbol{x}'} = -\Delta t \tag{5.2}$$

と設定する. ここで, Δt は, 1 ステップの離散時間を意味し, ロボットが一回の行動で移動する時間とする. 今回は, この ΔT は $0.1[s]$ とする. 次に, 障害物内での移動に関する報酬を

$$r_{\mathrm{obstacle}}{}^{a}_{\boldsymbol{x}\boldsymbol{x}'} = -c\Delta t \tag{5.3}$$

と設定する. 状態遷移先が障 t 物だった場合, 負の報酬として移動時間 ΔT の c 倍を与える. MDP では, 報酬は一元化する必要があるため, この 2 つの報酬を合わせて,

$$\begin{aligned} r^{a}_{\boldsymbol{x}\boldsymbol{x}'} &= r_{\mathrm{time}}{}^{a}_{\boldsymbol{x}\boldsymbol{x}'} + c r_{\mathrm{obstacle}}{}^{a}_{\boldsymbol{x}\boldsymbol{x}'} \\ &= -\Delta t - c\Delta t \end{aligned} \tag{5.4}$$

と設定する. c は定数であり, 障害物へ入ることのペナルティー度合いを表す. 今回のタスクでは, 障害物に入ることが許されないため, c の値を大きく 100 と設定する.

価値関数の計算は, `https://github.com/ryuichiueda/simple_value_iteration_ros.git` のコードを利用して行った. 計算には, インテル Core i7 8550U/1.8GHz/4 コアの CPU を搭載したコンピュータを用いて, 32 分 58 秒かかった.

5.2.5 MCL の設定

MCL のパーティクル数 N は, 500 で固定とする. 予測ステップにおけるパーティクルの状態遷移は, ロボットの運動モデルと同様とする. ロボットは, タスクが終了したかどう

かの情報を得ることができる．この情報を計測による更新ステップで, パーティクルの重み $w^{(i)}$ に反映する．尤度関数は

$$q(\text{not finished}|\boldsymbol{x}^{(i)}) = \begin{cases} 10^{-10} & (\boldsymbol{x}^{(i)} \in \mathcal{X}_f) \\ 1 & (\text{otherwise}) \end{cases} \tag{5.5}$$

とし, タスク実行中にゴール内へと侵入したパーティクルの重みを非常に小さくする．

5.2.6　PFC 法の設定

本実験では, PFC 法においてゴールに近いパーティクルが行動決定に与える影響度を表す m は, 2 とする．m の値を変えたときの, タスクの達成率と達成までの時間の関係は, [Ueda 18] において明らかにされている．この研究では, 触覚センサのみを有する簡易マニピュレータによる, 把持物体の探索動作において検証されている．移動ロボットのタスクにおける m とタスクの達成率, および達成までの時間については, まだ検証されていない．

本論文において新たに導入した, 障害物に侵入しそうなパーティクルが, 行動決定与える影響を変更する変数 $m^{(i)}_{\text{avoid}}$ を設定する．最大値と最小値については, それぞれ $m_{\text{avoid\_min}} = 1$ と $m_{\text{avoid\_max}} = 3$ とする．

また, 本評価では, ロボットがデッドロックすることを防ぐための単純な処理を追加する．ロボットは, 自身が左右の旋回行動 ccw, cw を交互に行った場合, 前進する行動 fw を行う．

5.3　結果と考察

本節では, シミュレーションによる評価実験の結果と考察を述べる．まず, それぞれの手法を定量的に比較する．そして, それぞれの手法で生成された挙動について確認する．

5.3.1　他手法との比較

本項では, 提案手法と他手法の比較を行う．5.2 節で述べた条件, タスクのもと, 提案手法を 4 つの手法と比較した．本論文で提案した障害物回避行動を取り入れた PFC 法を, 回避動作を入れていない通常の PFC 法と, Q-MDP 法 ($m = 0$ の PFC 法) と比較する．さらに, 参考として,

- ロボットの真の姿勢 $\boldsymbol{x}^*$ を使用した最適方策
- パーティクルの平均姿勢 $\bar{\boldsymbol{x}}$ を使用した最適方策

の 2 つを加えた, 5 つの手法で比較する．

結果を表 5.3.1 に示す．各列の数値は, 左から順に

- タスクの成功率
- タスク成功時の平均時間
- タスク成功時における障害物内を移動したパーティクル数 × 時間の平均時間

を表している.

当然, ロボットの真の姿勢 $\boldsymbol{x}^*$ を行動決定に使用した場合, タスクの成功率は 100% となっているが, パーティクルの平均姿勢 $\bar{\boldsymbol{x}}$ を利用した行動決定では, タスクの成功率は 20% と低い値になっていることが分かる. 2 つの手法において, タスク成功時の平均時間はほぼ同様の時間である. これは, $\bar{\boldsymbol{x}}$ を使用してタスクが成功したときには, $\boldsymbol{x}^*$ が $\bar{\boldsymbol{x}}$ の付近の少し右下に存在するからである. 大きく右下方向に離れている場合は, 障害物に衝突することはなくなるが, ゴール周辺に到達しても実際にゴールすることがないため, タスク失敗となる.

本論文で提案した手法は, 本評価実験においては通常 PFC 法に比べてタスク達成率が 99% に大幅に向上していることが分かる. パーティクルが障害物の中を移動した時間が 0[s] であるとおり, 分布全体が障害物を確実に回避している. 通常の PFC 法では, 状態価値が低い障害物内のパーティクルが行動決定に大きな影響を与えるため, 多くのパーティクルが長時間に渡って障害物内を移動している. Q-MDP 法では, 状態価値による影響の違いがないため, パーティクルの障害物内移動が PFC 法よりも少ない. 今回の評価では障害物内の報酬 $r_{\mathrm{obstacle}}{}^{a}_{\boldsymbol{x}\boldsymbol{x}'}$ をかなり小さく設定しているため, Q-MDP 法でも簡単にパーティクルが障害物内に入っていくことは少ない. しかし, 全体のパーティクル数に対して僅かな数であれば, 侵入していくことがあるため, 比較的少ないが 0 にはならない.

提案手法において失敗した 1% は, 初期姿勢のズレが平均値から大きく離れていたことや, 運動モデルのばらつきにより, $\boldsymbol{x}^*$ が分布から外れてしまった場合である. 今回の実験条件では, ロボットの観測が制限されていたため, このような自体も起こり得る. また, ロボットに搭載したセンサにより, 自己位置推定に役立つ観測を行える場合であっても, 分布が $\boldsymbol{x}^*$ と乖離する場合がある. このような状態を「誘拐状態」と言い, 再度パーティクルを $\boldsymbol{x}^*$ の周辺に分布させることは困難とされている. これは, 「ロボット誘拐問題」と呼ばれ, 復帰方法についてもいくつか研究が行われている [Lenser 00, **?**]. 本評価においては, $\boldsymbol{x}^*$ が分布ないに存在していることを前提とし, 誘拐状態からの回復アルゴリズムを実装していないため, タスクの失敗へとつながった.

提案手法ではタスク成功率から, 障害物回避後のゴール探索動作についても行えていることが分かる. 今回の実験条件では, ロボットの運動モデルのノイズを小さく設定しているため, 移動による分布の広がりが小さい. そのため, 分布をゴールに流しこむ動作にかかる時間があまり長くならずに済んでいると考えられる. Q-MDP 法では, パーティクルが障害物内を移動している時間が比較的短いにも関わらず, タスク達成率は半分以下と低く

表 5.1　Comparison of PFC with avoidance, nomal PFC, Q-MDP, and some other simulations.

Methods	Successful trials	Avg. of time in successful trials	Avg. of the total time of particles inside obstacle in successful trials
PFC with avoidance	99 %	72.3 [s]	0.0 [s]
PFC	51 %	63.0 [s]	2587.1 [s]
Q-MDP	40 %	65.0 [s]	22.6 [s]
Decision based on $\bar{\boldsymbol{x}}$	20 %	38.9 [s]	898.9 [s]
Decision based on $\boldsymbol{x}^*$	100 %	39.6 [s]	994.5 [s]

なっている. これは, 障害物を回避することができても, ゴール付近で探索動作を行うことができず, 打ち切りの時間まで停滞しタスク失敗となるためである.

タスク成功時の平均時間については, 本論文において提案した手法が, 通常の PFC 法や Q-MDP 法と比べて一番長いことが分かる. パーティクルの分布を行動決定に利用する 3 つの手法の中では, 通常の PFC 法のタスク達成時間が最も短い. これは, 通常の PFC 法は, 障害物内のパーティクルの行動が優先されるため, 3 つの手法の中で最も分布がインコース側を移動するためである. ロボットが分布のアウトコース側に存在し, 障害物を回避することができたとき, 最も早くゴールへと到達することができるのだと考えられる. Q-MDP 法は, 通常の PFC 法よりも分布が障害物内を避けるようにアウトコース側を移動するため, 成功時の時間が少し短いと考えられる. 提案手法では, パーティクルの分布全体が確実に障害物を避けるようにゴールへと向かうため, タスク達成までの時間は一番長くなってる.

5.3.2　生成された挙動の確認

本項では, 各手法により生成された挙動を確認する. 動作の様子を確認し, 挙動の特性について考えていく.

まず, 表 5.3.1 における下 2 つの挙動について確認する. ロボットの真の姿勢 $\boldsymbol{x}^*$ を使用した行動決定は, すべての試行において成功した. ロボットは, 図 5.3 に示すように, 障害物回避し, かつ最短の経路を通ってゴールへと向かう. 軌跡が角張っているのは, 状態空間を離散化したときの粗さが影響していると考えられる. 今回は θ を 10[deg] ごとに離散化したが, もっと細かく離散化することで, 軌跡はなめらかなものになると予想される.

パーティクルの平均姿勢 $\bar{\boldsymbol{x}}$ を行動決定に利用した場合, 図 5.4 に示すような挙動となる.

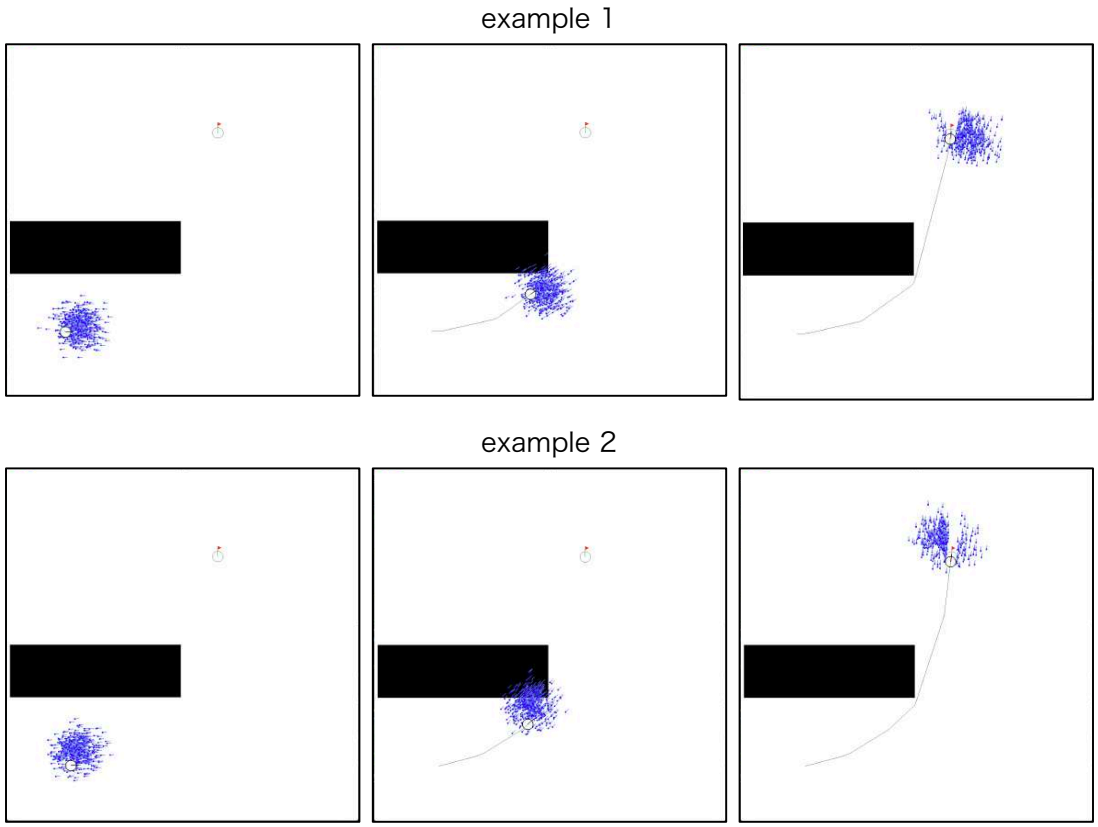

図 5.3 Robot behavior by true pose.

パーティクルの初期姿勢は, 正規分布に従うノイズによるばらつきがあるため, $\bar{\boldsymbol{x}}$ は分布のほぼ中心に位置する. 図の上部の試行では, ロボットの真の姿勢 $\boldsymbol{x}^*$ は分布の右下にあるため, 障害物付近を通る際アウトコース側を移動する. したがって, 障害物に突入することなくゴールへと向かっている. 一方で, 図の下部の試行では, $\boldsymbol{x}^*$ がインコース側を移動しているため, 障害物に衝突してしまっているのが分かる.

続いて, 提案手法による動作について確認する. 図 5.5 に, 提案手法による試行の一例を示す. 提案手法では, パーティクル全体が障害物を避けるように移動している. そして, ゴール周辺へとたどり着いた後, ゴール探索動作を行いタスク成功となっている. $t = 12.7$[s] でひとつのパーティクルが障害物に入りそうになった際, $m_{\text{avoid}}^{(i)}$ の値が最大値となる. ロボットはそのパーティクルの行動を最優先するように右旋回ののち前進し, パーティクルが障害物に侵入するのを防ぐ動作を行った.

しかし, 提案手法の動作には, 無駄な動きの多さが伺える. 図 5.6 に示す試行は, ロボットの無駄な動作が分かりやすく現れている. $t = 15.3$[s] でパーティクルが一つ障害物に侵入しそうになり, $t = 17.7$[s] で回避するように動作する. その後, 再度 $t = 19.8$[s] で別のパーティクルが障害物に親友しそうになり, $t = 12.7$[s] で回避動作をする. このように,

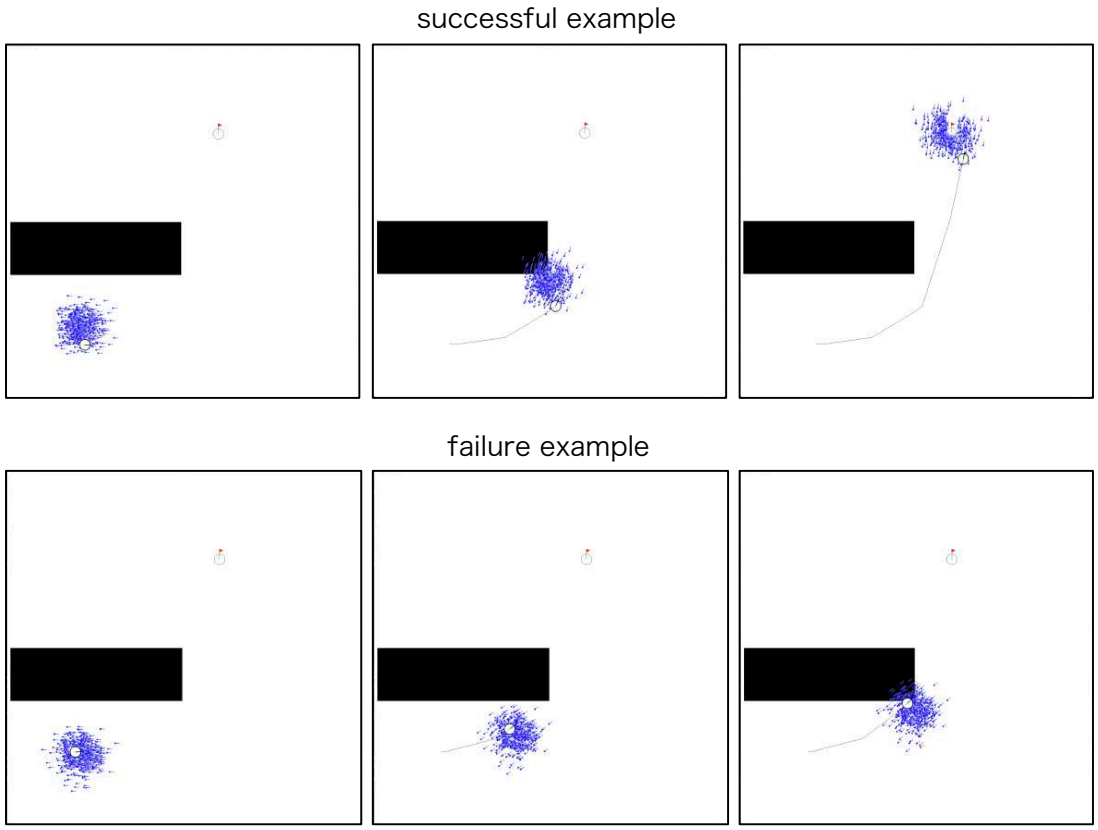

図 5.4　Robot behavior by particles average pose.

パーティクルの分布によっては, ゴールへ向かう動作と回避動作が何度も交互に繰り返され, ギザギザの軌跡を描く.

本手法では, 次の動作によるパーティクルの報酬のみから, $m_{\text{avoid}}^{(i)}$ を決定しているため, このようにその場しのぎのような動作になると考えられる. これを防ぐためには, 数ステップ先の動作までを含んで回避動作を考えるという方法が, 有効ではないかと予想される. しかし, この計算はロボットが動作するリアルタイムで行える必要があるため, 計算量と時間の問題が発生すると考えられる.

最後に, 通常の PFC 法と Q-MDP 法での動作について確認する. まず, 通常の PFC 法による動作の様子を, 図 5.7 に示す. 図の上段がタスク成功時の動作であり, 下段がタスク失敗時の動作である. 通常の PFC 方では, 分布全体がかなりインコース側を走行し, 多くのパーティクルが障害物内を移動しているのが分かる. 成功時と失敗時の違いは, パーティクルの平均姿勢 $\bar{\boldsymbol{x}}$ を利用した行動と同様に, 真の姿勢 $\boldsymbol{x}^*$ がアウトコース側に存在しているか否かである. 成功時の右のコマでは, 障害物を回避することに成功したのち, ゴールを探索する動作を行いタスク成功したことが分かる.

Q-MDP 法の動作については, 図 5.8 に示す. 上段のタスク成功時の動作と下段のタス

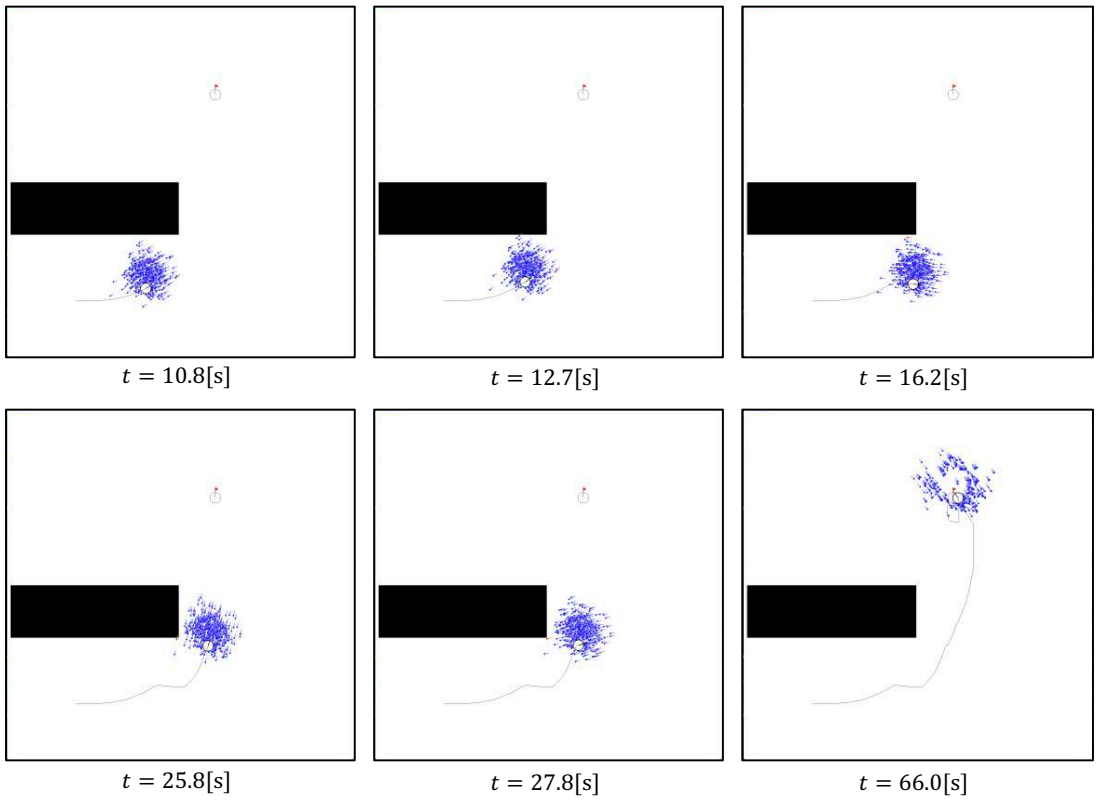

図 5.5 Robot behavior by PFC method with avoidance.

ク失敗時の動作ともに, いくつかのパーティクルは障害物内を移動しているものの, 大部分は障害物を回避するような動作を行っている. 障害物を回避後は, ゴールへと向かう. 成功時の動作では, 分布の中央付近に $\boldsymbol{x}^*$ が存在しているため, ゴールへ入ることができている. しかし, Q-MDP 法はゴールの探索動作を行わないため, 失敗時の動作ではゴール付近で行動が長時間停滞し失敗となる.

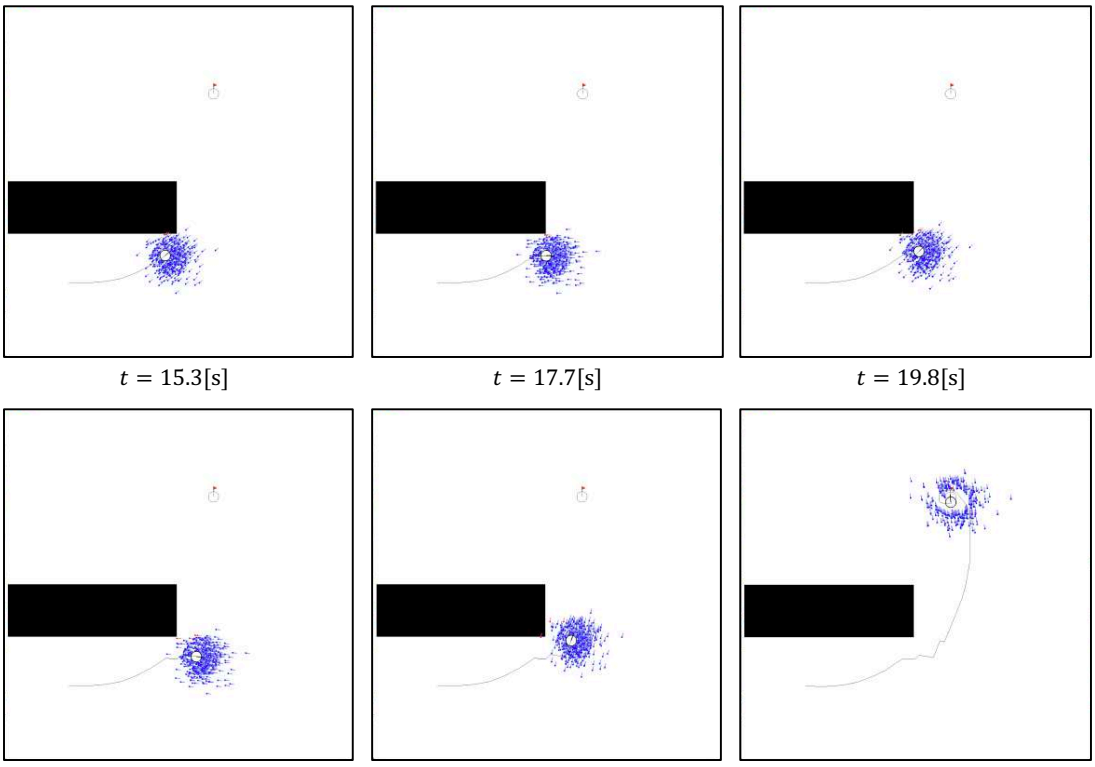

図 5.6　Robot behavior by PFC method with avoidance.

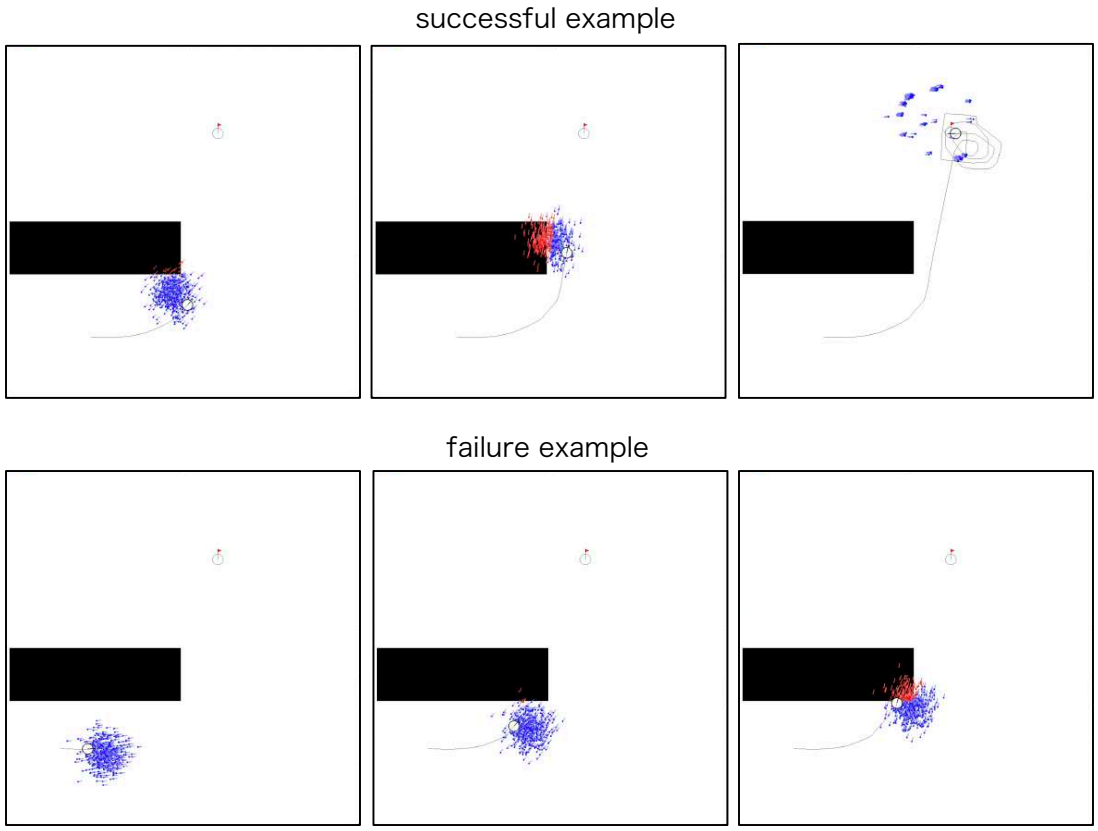

図 5.7 Robot behavior by normal PFC method.

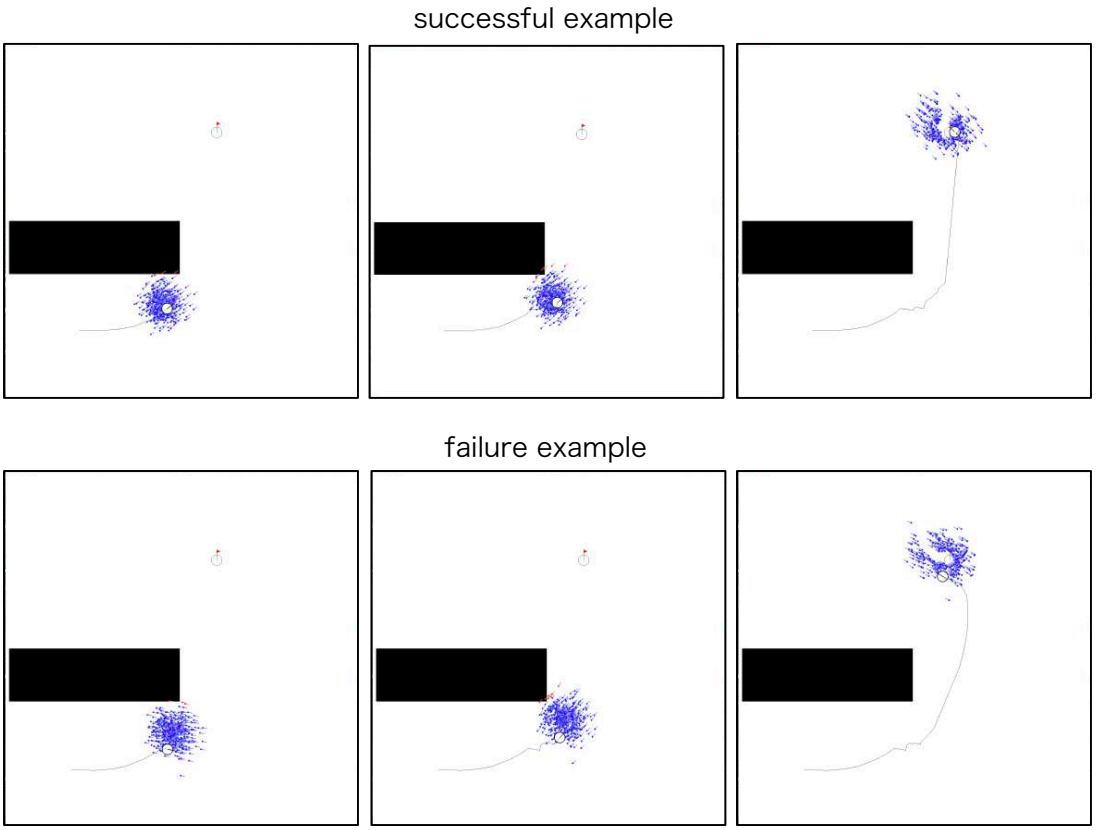

図 5.8　Robot behavior by Q-MDP method.

第 6 章

結論

6.1 本論文のまとめ

本論文では, 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した. MCL によって自己位置推定を行うロボット用に, 信念分布を近似するパーティクルの分布全体が, 障害物を回避するような動作を生成した. この動作は, 行動決定に与える影響度を決定するパラメータを各パーティクルに持たせ, この数値をタスク実行中に動的に変化させることで実現した. 次の行動で障害物に侵入するパーティクルの数値を大きくし, 障害物を避けようとする動作を強めることで, 目的の動作を達成した.

また, 提案手法の有効性について, 先行研究と比較し評価した. 既存の手法では, 障害物内をパーティクルの分布が移動することにより, ロボットの障害物への衝突が起こる可能性があることを示した. 提案手法は, パーティクルの分布が障害物内を移動するのをなくしたことで, この衝突の可能性を大幅に減らすことに成功した.

他にも, 本論文で提案した手法では, ロボットの挙動に無駄があることが問題と述べた. 移動ロボットが障害物を回避するとき, ギザギザの軌跡を描くように動作する. これは, パーティクル一つが障害物に入っただけで, 動作が大きく変わりすぎるためだと考えられる.

6.2 今後の展望

今後, ロボットの挙動の無駄の多さについて改善する必要があると言える. これは, 今回新たに導入した, パーティクルが行動決定に与える影響度の増やし方を工夫することで実現できる可能性がある. 現在の手法では, パーティクルが一つでも障害物に侵入しそうになった場合, 影響度を一気に最大にしている. これを徐々に増加させていくように変更す

る方法が一つ考えられる．また, パーティクルの数から, 次の行動でロボットが障害物に衝突する確率が計算できる．この確率に応じて, ロボットの行動を徐々に回避動作へと移行させるように, パラメータを増加させていくことも, 有効であると考えられる．

今回の評価実験のような, ゴールと障害物が遠くに存在している環境では, 障害物の回避を行うフェーズとゴール探索動作を行うフェーズが分かれている．しかし, ゴール付近に障害物が存在する場合, 障害物回避動作とゴール探索動作はトレードオフとなり, 2つの動作をループするようなデッドロックが発生する可能性があると考えられる．そういった場合, 障害物への衝突確率を判断材料に, 行動を回避動作から探索動作へ徐々に移行させていくような方法が有効ではないかと考えられる．また, パラメータの最大値も各パーティクルに個別で持たせることで, 障害物に入りそうになる度に, 行動へ与える影響度が大きくなりにくくなるような方法も, 有効ではないかと考えられる．

参考文献

[Dellaert 99] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo Localization for Mobile Robots. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA99)*, pp. 1322–1328, 1999.

[Fox 99] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of AAAI-99*, pp. 343–349, 1999.

[Kaelbling 98] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, Vol. 101, No. 1-2, pp. 99–134, 1998.

[Kalman 60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, Vol. 82, pp. 35–45, 1960.

[Lenser 00] S. Lenser, et al. Sensor resetting localization for poorly modelled robots. In *Proc. of IEEE ICRA*, pp. 1225–1232, 2000.

[Littman 95] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 362–370, 1995.

[Open Source Robotics Foundatio] Open Source Robotics Foundation. amcl — ROS Wiki. http://wiki.ros.org/amcl. (visited on 2018-

06-12).

[Quigley 09] 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.

[Roy 99] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal Navigation - Mobile Robot Navigation with Uncertainty in Dynamic Environments. In *Proc. of IEEE ICRA*, pp. 35–40, 1999.

[Seba 07] Sebastian Thrun, Wolfram Burgard, Dieter Fox(著), 上田 隆一 (訳). 確率ロボティクス. 毎日コミュニケーションズ, 2007.

[Thrun 05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic ROBOTICS*. MIT Press, 2005.

[Ueda 03] Ryuichi Ueda, Tamio Arai, Kazunori Asanuma, Shogo Kamiya, and Kazunori Umeda. Mobile Robot Navigation based on Expected State Value under Uncertainty of Self-localization. In *Proc. of IROS*, pp. 473–478, 2003.

[Ueda 15] Ryuichi Ueda. Generation of Compensation Behavior of Autonomous Robot for Uncertainty of Information with Probabilistic Flow Control. *Advanced Robotics*, Vol. 29, No. 11, pp. 721–734, 2015.

[Ueda 18] Ryuichi Ueda. Searching behavior of a simple manipulator only with sense of touch generated by probabilistic flow control. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 594–599. IEEE, 2018.

[夏迫 16] 夏迫, 井上, 寺戸, 天野, 久保田, 後藤, 塩谷, 嶋村, 長島, 上田ほか. つくばチャレンジ 2016 における千葉工業大学ロボット設計制御研究室の取り組み. 第 17 回計測自動制御学会システムインテグレーショ

ン部門講演会, 1B2-3, 2016.