博士学位論文

移動ロボットを滑らかに制御するための ローカル経路計画手法とその高速化

(Local Path Planning Algorithm for Smooth Control of Mobile Robots and Their Acceleration)

2024年

LIN Ziang

目次

第１章　序論

現代社会の発展に伴い，自律移動ロボットに対するニーズはますます広がってきており，宇宙探査から日常生活における掃除や配達などの様々なシーンで，自律移動ロボットは幅広い応用が期待されている[1]-[3]．自律移動ロボットの分野では，経路計画と動作計画が重要な役割を果たしている．経路計画は，与えられた環境において初期位置から目標までの最適な経路を計画するグローバル経路計画と，グローバル経路に従いつつ障害物と衝突しない経路をリアルタイムに生成するローカル経路計画からなる[4],[5]．様々な環境において，ロボットは障害物を回避し，環境の変化に迅速に対応し，安全を確保しながら効率よく目標地点に到達することが望ましい．

図 1.1に示すように，経路計画（Path planning）の関連研究は，2000年以降毎年増加している．一方，動作計画（Motion planning）とは，経路計画から導き出された経路を実行するためにロボットを制御する具体的な指令を算出することである[6]．実際のロボットの動作計画では，安定して効率的な運動を実現するために，ロボット自身の物理的特性や外部環境の影響などを考慮する必要がある．図 1.2に示すように，動作計画の関連研究は経路計画よりは数は少ないが同様の推移傾向が見られる．

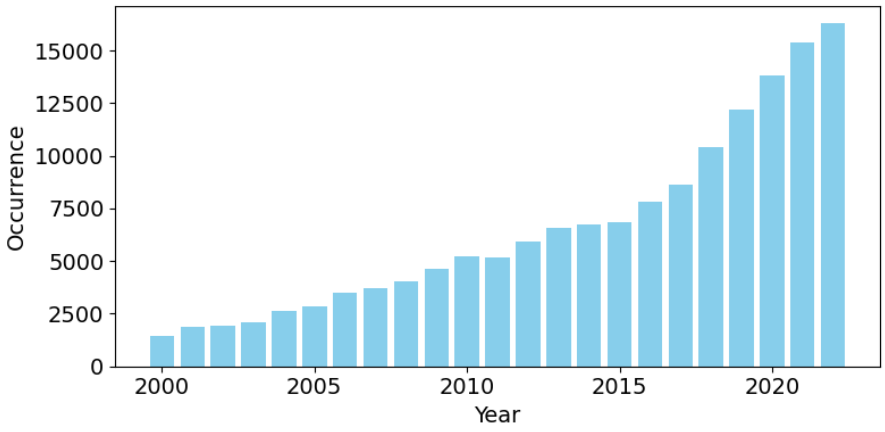


図 1.1　Path planningの関連研究の推移（筆者調べ）
※Google scholarによる英語論文の検索結果の統計

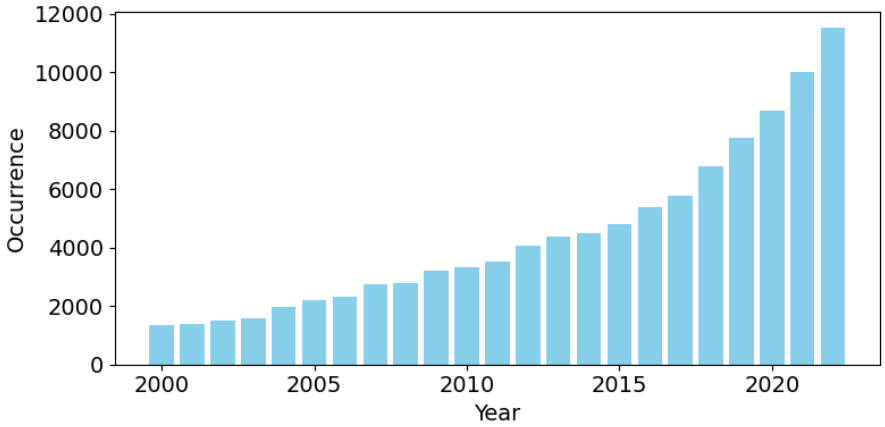


図 1.2　Motion planningの関連研究の推移（筆者調べ）
※Google scholarによる英語論文の検索結果の統計

ローカル経路計画の手法の一つであるDWA(Dynamic Window Approach)[7]は，ローカル経路計画と動作計画を統合した手法であり，ROS(Robot Operating System)[8]のデフォルトのローカル経路計画手法に採用されるなど，広く一般的に用いられている．DWAの関連研究の推移を図 1.3に示す．この図から，その重要性は示された．DWAは，ロボットが実行可能な速度指令空間と，タスクに応じた評価関数を定義することで，障害物に衝突しない速度指令を出力することができる．

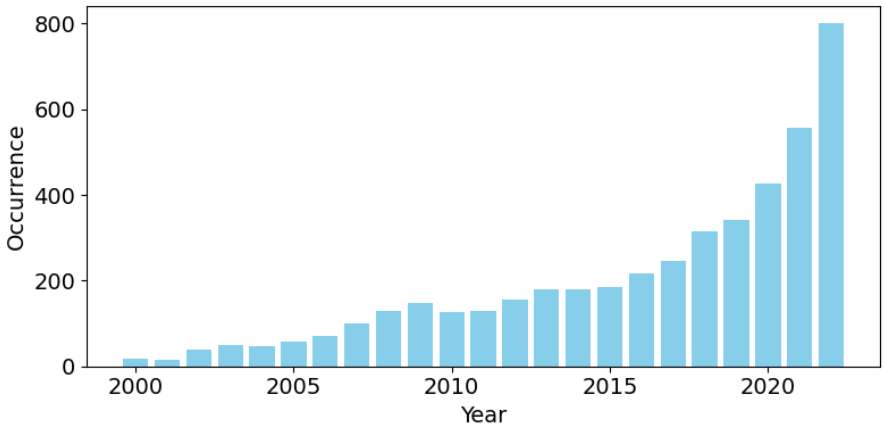


図 1.3　Dynamic window approachの関連研究の推移（筆者調べ）
※Google scholarによる英語論文の検索結果の統計

しかし，従来のDWAには二つの問題がある．一つ目は，実現可能な経路の一部しか予測できないという問題である．DWAでは，経路を予測する際に，速度が一定であることを仮定している．そのため，加減速を伴う経路は計画することができず，障害物の回避に失敗する場合がある．二つ目の問題は，躍度が制御できないという点である．躍度(Jerk)とは加速度の時間微分である．躍度が

大きくなると，力積の増大を招くため，ロボットが転倒したり，内部の部品にダメージを与えたりする[9]-[13]．また，乗り物の場合には，躍度が大きくなると乗客に不快感を与えることも知られている[14]．したがって，躍度を制御できるようになれば，ロボットの安全性や頑健性を高めるだけでなく，感性品質の向上にも寄与する．

そこで本論文の第4章では，躍度で制御可能なDWAを提案する．従来のDWAでは並進速度と角速度を選択するが，提案手法では，並進躍度と角躍度を選択する．また，従来は，予測期間中の速度が一定であることを仮定するが，提案手法では，予測期間中の躍度が一定であることを仮定し，速度と加速度には一定を仮定しない. これにより, 従来よりも現実に即した経路計画が可能となる．さらに，躍度を抑えた滑らかな加減速を計画することもできる．

また, 従来のDWAでは離散的な経路点を用いて予測経路を表しており, 経路点を生成するための繰り返し計算が必要である．衝突検知を行う際には，膨大な経路点と障害物点の点群間距離の最小値を求める必要があり時間がかかる．

そこで本論文の第5章では，予測経路を離散的な経路点ではなく円弧を用いてモデリングし，衝突検知を行う手法を提案する．経路予測に速度制御モデルを利用する場合, 円弧で経路を表現することで, 衝突検出の計算を容易にする．さらに，経路予測に変速モデルを利用する場合，その経路は円弧にはならないが，本論文では複数の円弧を用いて経路を近似し，衝突検出を行う手法を提案する. 提案手法では, 経路点を生成するための繰り返し計算が不要であるため, 経路計画の計算速度が向上する．

本論文の構成は次の通りである．第2章では，自律移動ロボット技術に関連する先行研究について述べる. 第3章では, DWAについて述べる． 第4章では，躍度制御モデルに基づいたDWAの改良とその実験について述べる．第5章では，経路の円弧近似によりDWAの高速化とその実験について述べる．最後に，第6章では結論を述べる．

第 2 章 関連研究

本章では，ロボットの自律移動を実現するための 4 つの技術，環境認識，位置推定，経路計画，動作計画を紹介した．環境認識はロボットが周囲の環境を認識する能力, 位置推定はロボットが自分の位置を正確に決定する能力を与える. 経路計画と動作計画は複雑な環境での安全で効率的な移動経路と運動設計をロボットに提供する.

2.1 環境認識

ロボットが環境から情報を取得・理解する過程を環境認識と呼ぶ. 周囲の環境を感知するために，カメラ，LIDAR，超音波センサーなど，様々なセンサーが使用される．まず，情報を取得する過程では，センサーの併用とノイズの除去が主な課題となる．その後，センサーの種類に対して，異なるアルゴリズムを利用し，物体の検出と認識，静的な障害物までの距離や角度の計測，動的な障害物の速度ベクトルの計算を行う. ここで得られた情報は位置推定や計画に使用される.

2013 年以前の物体検出と認識手法の多くは伝統的な手法に基づいており、特徴抽出と分類器の組み合わせがよく使われていた. 特徴抽出の段階でよく使われる手法は Haar[15]，SIFT[16]，SURF [17]などであり，分類器には SVM[18],[19]，Random Forest[20]などがよく使われていた．伝統的な手法の演算子は人工的に設計され，階層が浅く，特徴量の表現能力が限られている．2013 年から，深層学習に基づいた物体検出が急速に発展した．主に Densebox[21]，YOLO[22]，RetinaNet[23]などに代表される 1 ステージ型の検出手法と，RCNN シリーズ[24]-[27]，FPN[28]などに代表される 2 ステージ型の検出手法に分けられる.

2.2 位置推定

位置推定とは，ロボットが自分の位置を推定する過程である．センサー情報を地図や先験的情報と比較することで, 地図上のロボットの正確な位置を推定する．Monte Carlo Localization (MCL)[29]はよく使われる手法である．MCL は環境におけるロボットの可能な位置の分布を表す粒子の集合を使用する. 粒子の

ランダムにサンプリングと信頼度の更新することにより, ロボットの位置と姿勢を推定する.

一方，ロボットの位置と地図を同時に推定する過程は Simultaneous Localization and Mapping (SLAM)[30]と呼ばれる．入力によって，SLAM は 3 つに分類できる．それぞれは，ライダー(レーザースキャナ含む)に基づいた LiDAR SLAM，単眼またはステレオカメラに基づいた Visual SLAM と深度カメラに基づいた Depth SLAM である.

2.3 経路計画

会話ロボットなどを除けば, 多くのサービスロボットはタスクを達成するために「移動」を伴うため，与えられた地図を元に，ロボットの初期位置から目標までの経路を計画するステップは不可欠である．1959 年，Dijkstra はグラフ上の 2 頂点間の最短経路を求めるアルゴリズム[31]を提案した. Dijkstra 法では, 現在の距離が最小のノードを選択し, そのノードを経由して隣接ノードまでの距離を更新することを繰り返すことで最短経路を求める．1968 年 Hart らは，Dijkstra 法にヒューリスティック関数を追加し，計算時間を効果的に短縮した A-star[32]法を提案した．1994 年，Stentz は動的環境に対応する D-star[33]法を提案した．D-star 法には 2 つの重要な理念がある．1 つ目は後方探索と呼ばれる終点からの経路探索である．2 つ目はインクリメンタルサーチと呼ばれ，前回の計画から変更する必要がある部分のみを注目する.

1986 年，Khatib は物理学におけるポテンシャルエネルギーの概念を利用して，Artificial Potential Field (APF) [34]法を提案した．APF 法では，ロボットを 2 次元または 3 次元のポテンシャル場内を移動する粒子として扱い, 勾配の方向に力を加えることで，ロボットはポテンシャル場の力によって移動する．APF 法では経路が不安定になるという問題があったため，1991 年，Borenstein らは Vector Field Histogram (VFH) [35]法を提案した．VFH では，ロボット周囲の環境は極座標のヒストグラムを用いて記述され，通過可能な区域を記する．評価関数を用いて最適移動方向を計算する． 制御指令の出力方法が示されているが, それは単純な直進と回転しか扱うことができないため, 本論文では動作計画には分類しないこととする． Borenstein らの研究は，経路計画への貢献と認められる．しかし，これらのポテンシャルフィールドに基づいた手法は局部最小に陥る問題が存在する.

Rapid exploring Random Tree (RRT)アルゴリズムシリーズ[36]-[38]はランダム探索に基づいて，ランダムな位置から連続的にサンプリングし，最も近いノードを接続してツリーを拡張することにより，未知の空間を探索する．RRT は，最短経路を見つけるのではなく，経路を素早く探索する傾向があるため，最適経路を見つけるのを保証できない．

一方，曲線あてはめ法は，すでに計画された経路を最適化するものであり，単独で使用することはできない．

ここまで，ほとんどすべてのアルゴリズムは，計画された経路を制御指令に変換する能力を持っておらず，追加の動作計画アルゴリズムを必要とする．また，経路計画にロボットの物理的制約(ハードウェア，運動学)が考慮されていない．この問題に対して，Simmons は経路計画と動作計画を統合する Curvature Velocity Method (CVM)[39]を提案し，障害物回避の問題を制約付きの速度空間最適化問題として考えることで，ロボットの速度と加速度，物理的な制約や環境の制約などを考慮した経路計画を実現とした．すべての制約条件を満たす場合，速度，障害物との距離，ロボットの向き 3 つの要因を含む評価関数が設定された．CVM を基に，Fox らは DWA を提案した．CVM と DWA は非常に似た手法と考えられる．

2.4　動作計画

図 2.6の経路計画アルゴリズムを使用するロボットの場合，経路を追跡するための制御指令を出力するには，動作計画アルゴリズムが必要である．1932年，Nyquistが提案したProportional-Integral-Differential (PID)制御はよく使われる動作計画アルゴリズムである[40]．PID制御は制御工学におけるフィードバック制御の一種であり，入力値の制御を出力値と目標値との偏差，その積分，および微分の3つの要素によって行う方法のことである．しかし，モデルの複雑さが増すに伴い，制御システムは線形から非線形へ変化することが多くなり，非線形性の程度が増すに伴い，PID制御の効果が悪くなる．

非線形システムに対して，1958年，Schwartzが提案したModel Predictive Control (MPC)はよく使われる動作計画アルゴリズムである[41]．MPCの予測モデルには，入力と出力の遷移モデルが必要である．図 2.4(a)にMPCの基本原理を示す．横軸は時間，原点は現在時刻を表す．負の半軸では，すでに発生した入力（青色の線）と出力（薄い茶色の線）を表す．正の半軸では，これから計

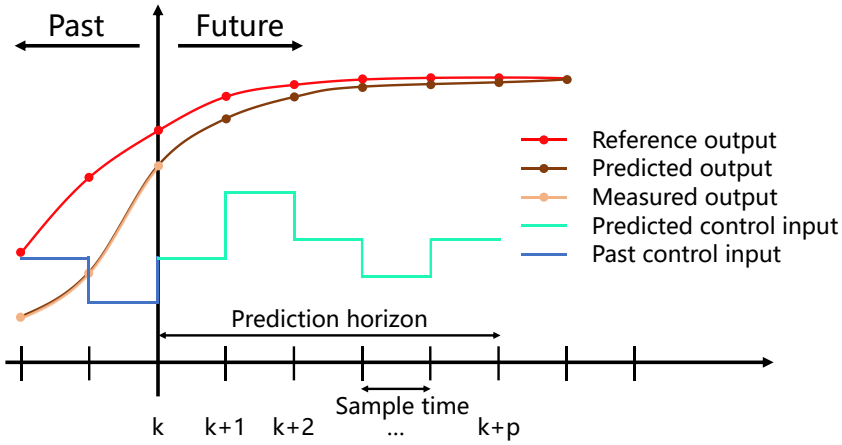
画する入力と出力を表す．縦軸は入力と出力を表す．赤線は計画された毎時刻の出力を表す．制御の目標は，実際の出力（茶色の線）を計画された出力にできるだけ近づけることである．そのために，将来の入力（緑の線）を計画する．現時点からの一定期間は予測区間(Prediction Horizon)と呼ばれる．予測区間はサンプリング間隔(Sample Time)によってP分割される．サンプリング間隔は，2つの制御の間隔として考えられる．サンプリング間隔ごとに異なる制御指令を選択できるため，多様な制御指令の系列が生成できる．評価関数を通じて最適な制御指令の系列を選択し，系列の最初の指令を出力とする．しかし，制御サイクルの短縮と選択可能な制御指令の増加とともに，MPCの計算量は膨大となるため，ソルバーが必要となり，リアルタイム制御が困難になる．

ここで，図 2.4(b)に示すように，予測区間における制御指令の系列を最初のいくつかのサンプリング間隔と残りのサンプリング間隔に分けて，最初のいくつかのサンプリング間隔の制御指令の系列のみを計算し，残りのサンプリング間隔では制御指令が不変とする．最初のいくつかのサンプリング間隔は制御区間(Control Horizon)と呼ばれる．MPCを用いたロボットの自律移動に関する先行研究では，制御区間内におけるサンプリングの数は3～5個に設定されることが多い[42]-[46]．

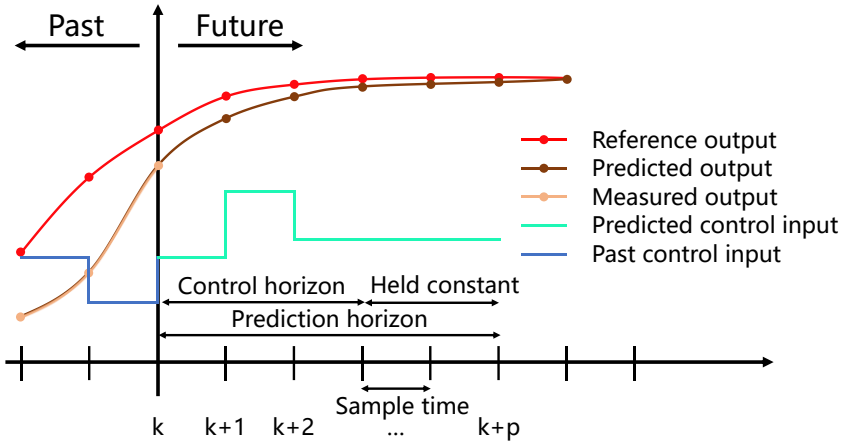
DWAはMPCの簡略版とみられる．DWAでは，ロボットの運動学モデルを予測モデルとして使用するため，ローカル経路計画と動作計画を同時に実行できる．また，DWAの制御区間内におけるサンプリングの数は1のみ，，残りの予測区間では制御指令が不変とする．これにより，MPCに比べDWAの計算量が大きく減少できるが，制御指令の系列の多様性が低すぎるため，障害物と衝突のない経路を生成できない可能性がある．

これらのアルゴリズムは，いくつかの視点によって異なる分類をすることができる．例えば，経路の性質に着目すると，図 2.5に示すように，グローバル経路計画とローカル経路計画に分類できる．この場合，グローバル経路計画は地図上に起点から終点まで，既知の障害物との衝突のない経路を計画する．グラフ探索によるアルゴリズムでは，起点から終点までの最短経路が得られる．サンプリングによるアルゴリズムは，探索速度を重視しているが，得られた経路は最短ではない可能性が高い．しかし，それらのグローバル経路計画で計画した経路は，多くの場合，ロボットの移動にそのまま使用できるものではなく，実際の移動状況に応じて経路を最適化する必要がある．この手法として，多項式曲線やベジェ曲線を使った曲線あてはめ手法や，ロボットの運動学モデルに基づいた手法がある．これらの手法はローカル経路計画と呼ばれる．ローカル

経路計画を適用した結果の軌跡は, 計画されたグローバル経路と一致しないことがある.



(a) Held constantなし



(b) Held constantあり

図 2.4　MPC の基本原理

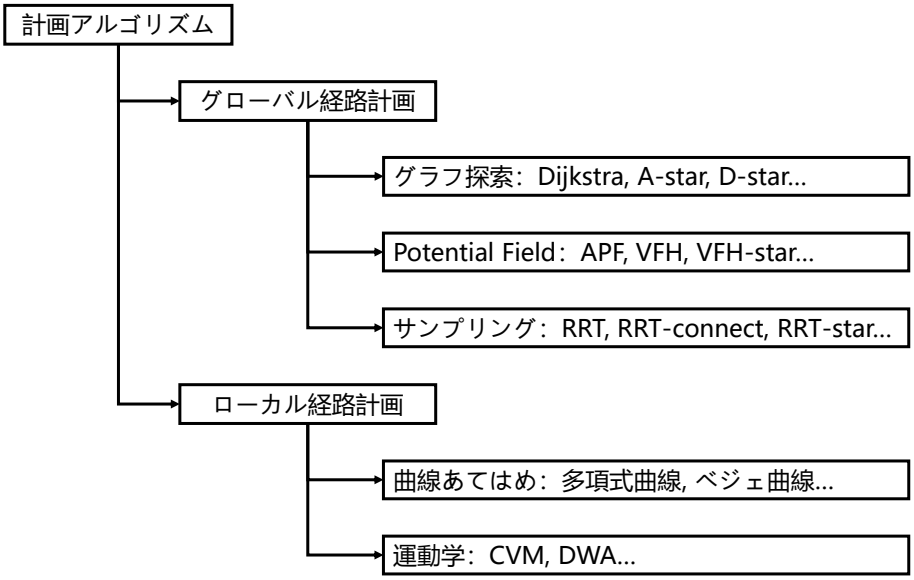


図 2.5　経路の性質による分類

アルゴリズムの出力に着目すると，図 2.6に示すように，経路計画と動作計画に分類される．この場合，経路点群を出力とするアルゴリズムは，経路計画と呼ばれ，制御指令を出力とするアルゴリズムは，動作計画と呼ばれる．

これで，DWAはローカル経路計画と動作計画の両方の能力を持っていることがわかる．

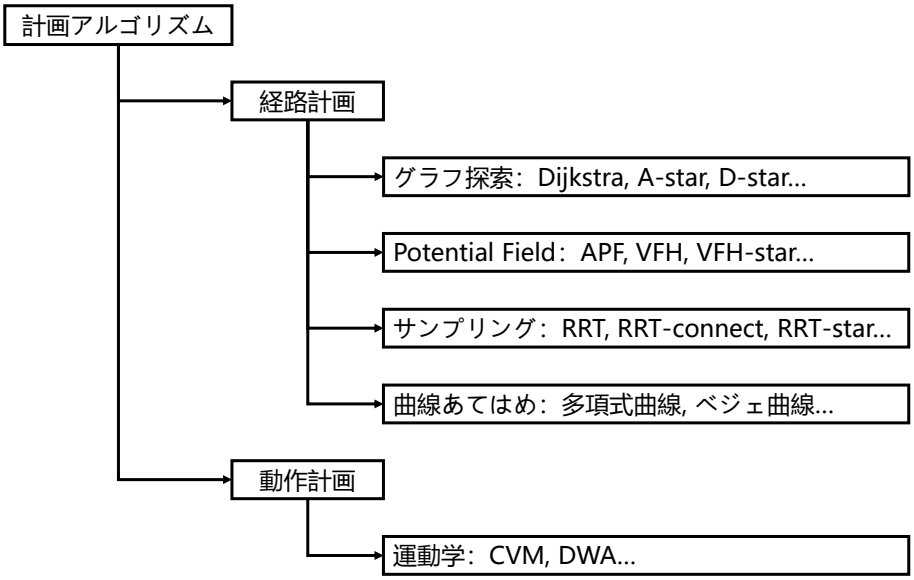


図 2.6　出力による分類

第 3 章　Dynamic Window Approach

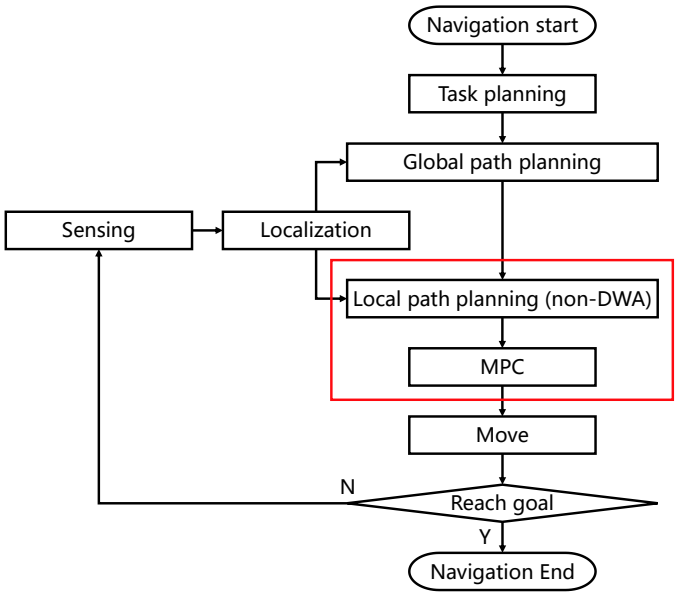
3.1　DWA と MPC の比較

第 2 章では，経路計画アルゴリズムとしての DWA と動作計画アルゴリズムとしての MPC について述べたが，両アルゴリズムはロボットに最適な制御指令を出力できる．DWA について詳しく説明する前に，DWA と MPC の関連性と違いを説明する．

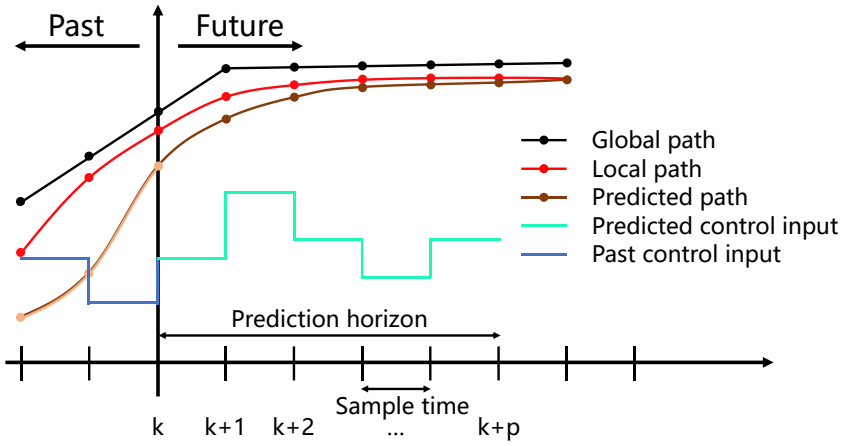
MPC は様々な制御問題を解決するための一般的なフレームワークを提供する方法論である[41]．非線形システム，多変数システム，制約条件付き制御を含む制御理論研究や高度な制御タスクに一般的に適用されている[47]-[51]．DWA は MPC のロボット自律移動への応用とみなすことができる．DWA はロボット自律移動と障害物回避のタスクに集中し，評価関数に基づいて最適なロボット動作を選択する．

具体的に，MPC を用いロボットの制御指令を出力する場合，そのワークフロー図 3.7(a)のようになる．グローバル経路計画では，目標までの最短経路を計画する．図 3.7(b)に，MPC を用い動作を計画するイメージ図を示す．グローバル経路を黒線で表す．ローカル経路計画では，最短経路の最適化と動的な障害物を回避できる経路を計画する．図 3.7(b)ではローカル経路を赤線で表す．次に，MPC では，ロボットの運動学モデルを予測モデルとして使用し，予測経路を生成する．すべての予測経路から，ローカル経路との偏差が最も小さいものを選択し，それに対応する制御指令を出力する．図 3.7(b)では予測経路を茶色の線，制御指令を緑の線で表す．

MPC は予測経路とローカル経路の偏差を主な評価項として使用するため，高い経路追跡能力を持っている．



(a)　ワークフロー



(b)　イメージ

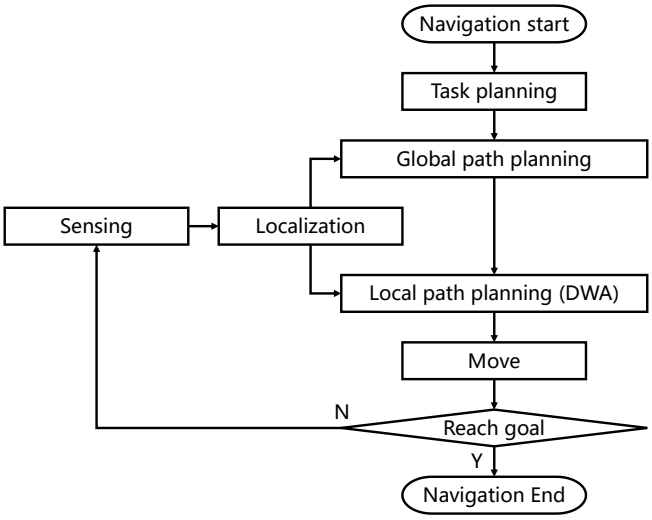
図 3.7　MPC を用いる動作を計画する場合のワークフローとイメージ

DWA を用いロボットの制御指令を出力する場合，図 3.8(a)に示すワークフローのように，ローカル経路計画が不要となる．図 3.8(b)に示すように，予測経路はローカル経路として使用される．予測経路と障害物の距離により，衝突の有無を判断する．すべての予測経路から，障害物と衝突せず，かつ予測経路の到着点とグローバル経路との偏差が最も小さいものを選択し，それに対応する制御指令を出力する．

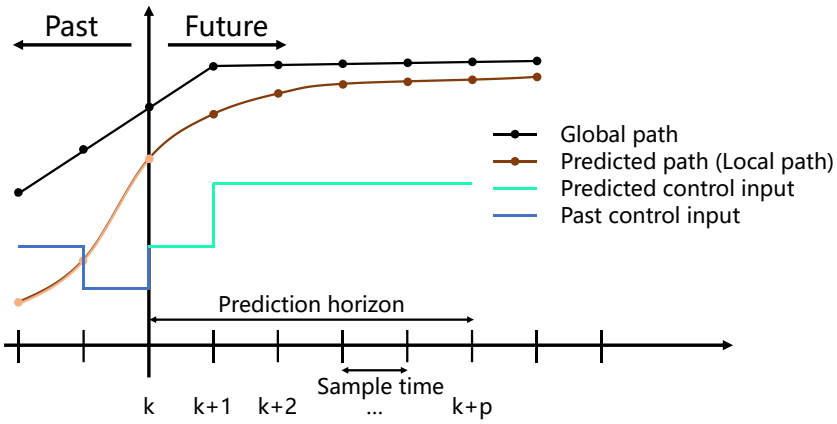
DWA は予測経路の到着点とグローバル経路の偏差を評価項として使用するため，最適化されていないグローバル経路を追跡することはなく，一定の自由度を持っている．また，DWA の制御区間には 1 個のサンプリング時間のみを含むため，予測経路の数は MPC より少なくなり，計算時間は大幅に減少でき

る．一方で，制御指令の系列の多様性が低いため，障害物と衝突しない経路を生成できない可能性がある．

DWA と MPC の比較を表 3.1 に示す．両手法ともに，入力と出力の遷移モデルが必要だが，DWA はロボットの動作計画に使用されるため，より具体的なロボット運動学モデルが使用されることがわかる．両手法ともに，出力は最適な制御系列の最初の指令である．また，DWA は MPC と違い，その制御区間は 1 つのサンプル区間だけで構成される．これにより，計算時間が大幅に短縮されるが，生成された制御系列の多様性も低くなる．



(a)　ワークフロー



(b)　イメージ

図 3.8　DWA を用いて動作を計画する場合のワークフローとイメージ

表 3.1　DWA と MPC の比較

	MPC	MPC in robot	DWA
予測モデル	制限なし	ロボットの運動学モデル	ロボットの運動学モデル
制御区間	3～5 個のサンプリング間隔		1 個のサンプリング間隔
予測区間	約 20 個のサンプリング間隔		
出力	制御指令の系列における最初の指令		
評価対象	予測出力全体		最後の予測出力

3.2　DWA のワークフロー

DWAはローカル経路計画アルゴリズムであり，ロボットの速度パラメータを考慮しながら経路を計画する．図 3.9は，図 3.8に示したワークフローにDWAの具体的な処理を追加したものである．まず，ステップ(A)では，実行可能な速度のサンプリングを行う. 図 3.10に探索空間を示す. 図 3.10における，縦軸と横軸はそれぞれロボットの並進速度と角速度である．灰色の矩形は，ロボットのハードウェアの制約の下，実行可能な速度の範囲を表す．現在時刻t_0の速度(v_0, ω_0)を基準とし，微小なサンプリング間隔Δtで到達可能な速度の範囲を小さい矩形で表す．この矩形の大きさは，選択可能な加速度の上限$a^{\mathrm{V}}_{\max}$と下限$-a^{\mathrm{V}}_{\max}$，角加速度の上限$a^{\Omega}_{\max}$と下限$-a^{\Omega}_{\max}$により決まる．小さい矩形と灰色の矩形の共通部分(図中の斜線部)が，DWAの探索空間となるDynamic Window(DW)である．図中のA～Dは4.2節で説明する．

速度空間は本来連続であるが, DWAでは予測経路を生成するため, 速度空間を格子状に離散化して次の時刻の制御指令を選択する．また，予測期間t_pも時刻$t_0, \ldots, t_n (n = 1, \ldots, n_{\mathrm{p}})$と離散化する．予測期間$t_{\mathrm{p}}$，予測ステップ数$n_{\mathrm{p}}$，ステップ間のサンプリング時間$\Delta t$の関係を式(3.1)に示す. 図 3.11に$t_0$からの可能な速度変化を表す． DWAでは，$t_0$~$t_1$の間のみ加速し，$t_1$以降は速度不変を仮定する．予測期間における並進速度と角速度を式(3.2)，(3.3)に示す．a^{V}は並進加速度$(a^{\mathrm{V}} \in [-a^{\mathrm{V}}_{\max}, a^{\mathrm{V}}_{\max}])$, a^{Ω}は角加速度$(a^{\Omega} \in [-a^{\Omega}_{\max}, a^{\Omega}_{\max}])$, v_nとω_nは時刻t_nでの並進速度と角速度を表す.

$$t_{\mathrm{p}} = n_{\mathrm{p}} \cdot \Delta t \tag{3.1}$$

$$v_n = v_0 + a^{\mathrm{V}} \cdot \Delta t \tag{3.2}$$

$$\omega_n = \omega_0 + a^{\Omega} \cdot \Delta t \tag{3.3}$$

次のステップ(B)では, DWから並進速度と角速度のペア(v, ω)を選択すると，運動学モデルを用いて，時刻t_nにおけるロボットの位置座標(x_n, y_n)と向きθ_n

を$n = 1, \ldots, n_{\mathrm{p}}$の範囲で逐次的に予測する. 並進速度と角速度のペアに対する座標と向きの系列は予測経路となる. 経路点の数と予測ステップの数が等しいことがわかる．図 3.12は並進速度が同じで角速度が異なる5個の速度ペアに対し予測された経路の例を示している．このステップで消費した計算時間を$t_{\mathrm{B1}}(n_{\mathrm{p}})$で表す.

ステップ(C)，(D)，(E)で，式(3.4)で定義される評価関数に使用された評価項目の値を算出する. この評価関数は予測経路の到着点でロボットの向きと目標の方位角の差の評価関数$\mathrm{heading}(v, \omega)$，障害物との距離の評価関数$\mathrm{dist}(v, \omega)$，速度の評価関数$\mathrm{vel}(v, \omega)$から構成されている．なお，$\alpha$，$\beta$，$\gamma$は各評価項目の重みである．$\mathrm{heading}(v, \omega)$と$\mathrm{vel}(v, \omega)$の計算には，経路の到着点での向きと速度のみが使用される．$\mathrm{dist}(v, \omega)$の計算では，経路点群と障害物点群の最小距離を算出する．経路の評価に消費した計算時間を$t_{\mathrm{B2}}(n_{\mathrm{p}}, n_{\mathrm{o}})$で表す．ただし，$n_o$は検出された障害物点の数を表す.

$$\mathrm{G}(v, \omega) = \alpha \cdot \mathrm{heading}(v, \omega) + \beta \cdot \mathrm{dist}(v, \omega) + \gamma \cdot \mathrm{vel}(v, \omega) \tag{3.4}$$

すべての並進速度と角速度のペアに対する経路を予測・評価した後，ステップ(F)で最適な並進速度と角速度のペアを出力する．一回の計画で要する計算時間は式(3.5)で定義される.

$$t_{\mathrm{B}}(n_{\mathrm{p}}, n_{\mathrm{o}}) = t_{\mathrm{B1}}(n_{\mathrm{p}}) + t_{\mathrm{B2}}(n_{\mathrm{p}}, n_{\mathrm{o}}) \tag{3.5}$$

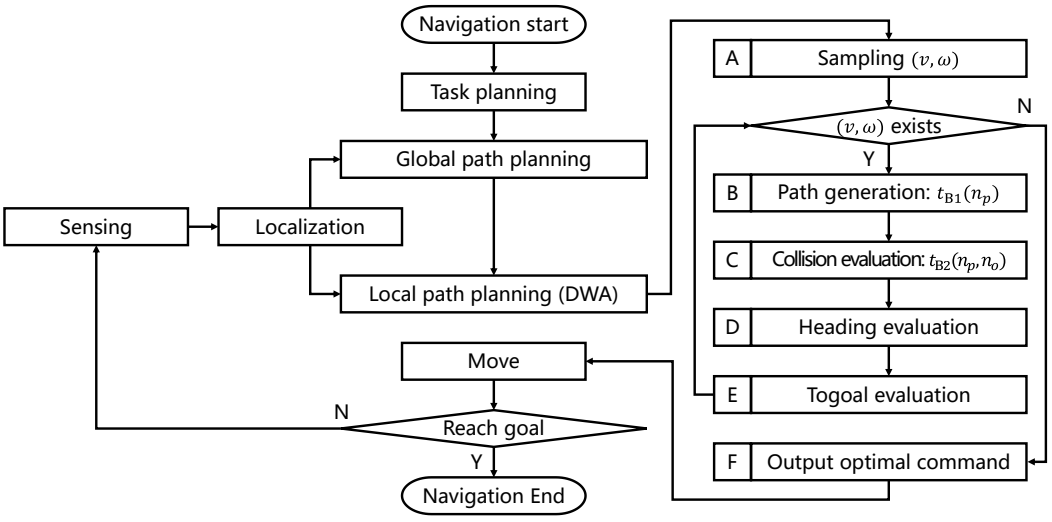


図 3.9　DWA を含んだ自律移動ロボットのワークフロー

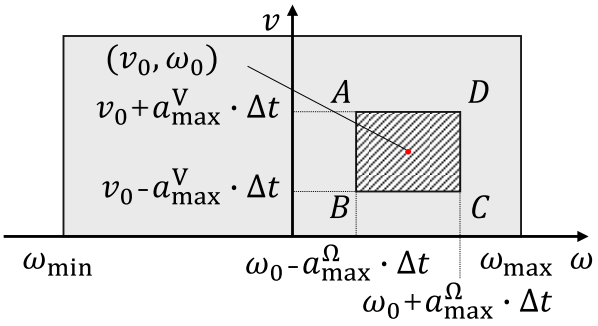


図 3.10　Dynamic Window の概念図

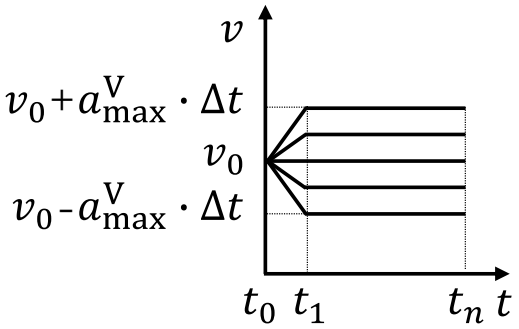


図 3.11　Dynamic Window と予測出力の関係

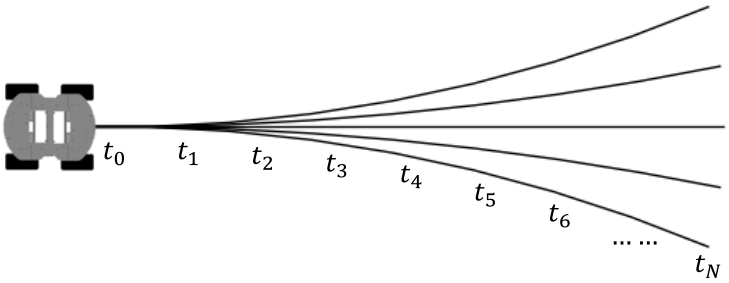


図 3.12　予測経路の概念図

3.3　運動学モデル

ロボットの運動学モデルとは，ロボットが予測入力に対して，どのような動きをするかを表すモデルである．これらの問題は，ロボットの自律移動，経路計画，運動制御，シミュレーションにおいて重要な役割を果たす．ロボットの運動学モデルは研究者が数学的モデリングと解析を通じてロボットの動きを理解し，制御する方法を提供する．

差動車輪付きロボット(Differential wheeled robot)は，最も一般的な移動ロボットである．差動車輪付きロボットは，最低一対の駆動輪を持ち，構造によっ

て非駆動輪を持つ場合と持たない場合がある. 駆動輪を異なる速度で回転させることで，ロボットの移動と回転が可能になる.

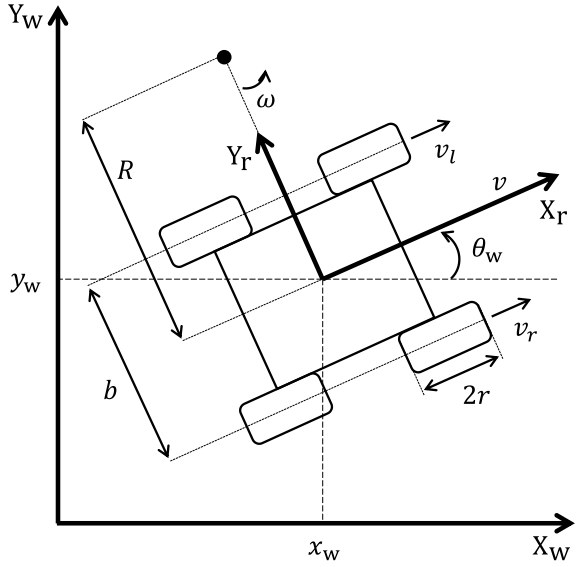


図 3.13　差動車輪付きロボットの運動学モデル

図 3.13 には差動車輪付きロボットの運動学モデルを示す. X_wとY_wは世界座標系の軸を表す．また，X_rとY_rはロボットの中心を原点としたロボット座標系の軸を表す. x_wとy_wは世界座標系におけるロボットの座標, θ_wはロボットの向きを表す. 車輪が常に地面と接触しスリップしないと仮定すると，ロボットは平面上のある点を中心とした円運動をする．v_lとv_rは左右車輪の地面に対する速度を表す．ωは円運動の角速度を表す．Rはロボットの中心から円運動の中心までの距離を表す．角速度の定義から，式(3.6)と(3.7)が得られる.

$$v_r = \omega \cdot \left(R + \frac{b}{2}\right) \tag{3.6}$$

$$v_l = \omega \cdot \left(R - \frac{b}{2}\right) \tag{3.7}$$

式(3.6)と(3.7)を整理すると，式(3.8)と(3.9)が得られる.

$$\omega = \frac{v_r - v_l}{b} \tag{3.8}$$

$$R = \frac{b \cdot (v_r + v_l)}{2(v_r - v_l)} \tag{3.9}$$

式(3.8)と(3.9)を角速度の定義式に代入すると，ロボットの中心の瞬間速度を表す式(3.10)が得られる.

$$v = \omega \cdot R = \frac{v_{\mathrm{r}} + v_{\mathrm{l}}}{2} \tag{3.10}$$

r_{w}を車輪の半径とすると，車輪の回転速度$\omega_{\mathrm{l}}, \omega_{\mathrm{r}}$と左右車輪の地面に対する速度の関係は式(3.11)と(3.12)で表される．

$$v_{\mathrm{r}} = r_{\mathrm{w}} \cdot \omega_{\mathrm{r}} \tag{3.11}$$
$$v_{\mathrm{l}} = r_{\mathrm{w}} \cdot \omega_{\mathrm{l}} \tag{3.12}$$

ロボット座標系における$\mathrm{X_r}$と$\mathrm{Y_r}$方向の移動量の時間微分を$\dot{x}_{\mathrm{r}}$と$\dot{y}_{\mathrm{r}}$で表し，向き変化の時間微分を$\dot{\theta}_{\mathrm{r}}$で表すと，ロボット座標系におけるロボットの運動学モデルは式(3.13)で表すことができる．

$$\begin{bmatrix} \dot{x}_{\mathrm{r}} \\ \dot{y}_{\mathrm{r}} \\ \dot{\theta}_{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{xr}} \\ v_{\mathrm{yr}} \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r_{\mathrm{w}}}{2} & \frac{r_{\mathrm{w}}}{2} \\ 0 & 0 \\ -\frac{r_{\mathrm{w}}}{b} & \frac{r_{\mathrm{w}}}{b} \end{bmatrix} \begin{bmatrix} \omega_{\mathrm{l}} \\ \omega_{\mathrm{r}} \end{bmatrix} \tag{3.13}$$

また，世界座標系における$\mathrm{X_w}$と$\mathrm{Y_w}$方向の移動量の時間微分を$\dot{x}_{\mathrm{w}}$と$\dot{y}_{\mathrm{w}}$で表し，向き変化の時間微分を$\dot{\theta}_{\mathrm{w}}$で表すと，世界座標系におけるロボットの運動学モデルは式(3.14)で表すことができる．このモデルは多くの研究に使用される[52]-[55]．本論文では，このモデルを tangent モデルと呼ぶ．

$$\begin{aligned} \begin{bmatrix} \dot{x}_{\mathrm{w}} \\ \dot{y}_{\mathrm{w}} \\ \dot{\theta}_{\mathrm{w}} \end{bmatrix} &= \begin{bmatrix} \cos\theta_{\mathrm{w}} & -\sin\theta_{\mathrm{w}} & 0 \\ \sin\theta_{\mathrm{w}} & \cos\theta_{\mathrm{w}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{\mathrm{r}} \\ \dot{y}_{\mathrm{r}} \\ \dot{\theta}_{\mathrm{r}} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta_{\mathrm{w}} & -\sin\theta_{\mathrm{w}} & 0 \\ \sin\theta_{\mathrm{w}} & \cos\theta_{\mathrm{w}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r_{\mathrm{w}}}{2} & \frac{r_{\mathrm{w}}}{2} \\ 0 & 0 \\ -\frac{r_{\mathrm{w}}}{b} & \frac{r_{\mathrm{w}}}{b} \end{bmatrix} \begin{bmatrix} \omega_{\mathrm{l}} \\ \omega_{\mathrm{r}} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta_{\mathrm{w}} & 0 \\ \sin\theta_{\mathrm{w}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \end{aligned} \tag{3.14}$$

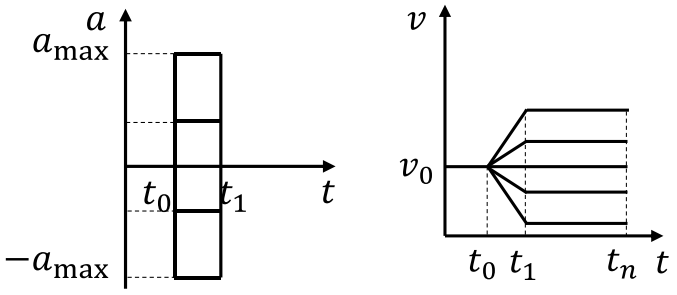
また，Yang らの SLAM に関する研究に式(3.20)に示す運動学モデルを使用した．本論文では，このモデルを secant モデルと呼ぶ．

$$\begin{bmatrix} \dot{x}_{\mathrm{w}} \\ \dot{y}_{\mathrm{w}} \\ \dot{\theta}_{\mathrm{w}} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{\mathrm{w}} + 0.5 \cdot \dot{\theta}_{\mathrm{w}}) & 0 \\ \sin(\theta_{\mathrm{w}} + 0.5 \cdot \dot{\theta}_{\mathrm{w}}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.15}$$

3.4　速度モデル

3.4.1　速度制御モデル

DWA は計算量を削減するために，予測区間における速度の変化を一定のルールに従わせる．図 3.14 に示すように，従来の DWA では，予測区間$t_0 \sim t_n$における，最初のサンプリング時間$t_0 \sim t_1$のみに速度は変化させ，残りの予測区間$t_1 \sim t_n$に不変となる．予測期間における，すべての時刻の速度と角速度は式(3.16)と(3.17)により計算できる．そのため，本論文ではその速度変化ルールを速度制御モデルと呼ぶ.



(a)　加速度　　(b)　速度

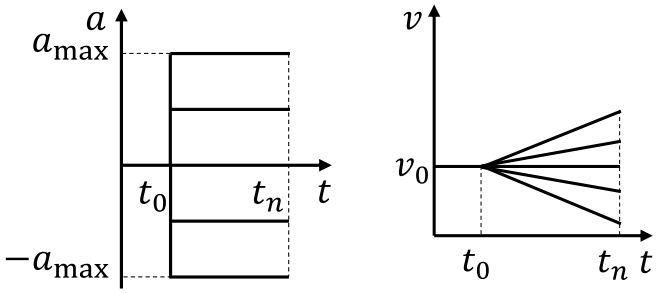
図 3.14　速度制御モデルの概念図

$$v_n = v_0 + a^{\mathrm{V}} \cdot \Delta t \tag{3.16}$$

$$\omega_n = \omega_0 + a^{\Omega} \cdot \Delta t \tag{3.17}$$

3.4.2　加速度制御モデル

一方，Gerkey らの研究では，図 3.15 のように，予測区間$t_0 \sim t_n$における，加速度不変，速度可変とすると，$t_0 \sim t_n$の間，加速，減速し続けることが可能となるモデルを使用した[57]．Gerkey ら手法はよく Trajectory Rollout と呼ばれ，速度制御モデルを使用する DWA の違いを示している[58]．予測期間における，すべての時刻の速度と角速度は式(3.18)と(3.19)により計算できる．a^{V}とa^{Ω}は選択された加速度と角加速度を表す．本論文では，その速度変化ルールを加速度制御モデルと呼ぶ．速度制御モデルと比べ，加速度制御モデルでは，速度を変化させて障害物を回避するような予測経路を生成できる．加減速により障害物を回避するのはロボットの実際の動作により近い.



(a)　加速度　　(b)　速度

図 3.15　加速度制御モデルの概念図

$$v_n = v_{n-1} + a^{\mathrm{V}} \cdot \Delta t \tag{3.18}$$

$$\omega_n = \omega_{n-1} + a^{\Omega} \cdot \Delta t \tag{3.19}$$

ここで注意すべきことは，ロボットの複数の自由度の制御指令入力は同様の速度モデルに従う必要がない．例えば，差動車輪付きロボットは並進速度と角速度の 2 つの自由度の制御指令入力を持ち，2 つの自由度のそれぞれに異なる速度モデルを適用することも可能である．

3.5　評価関数

DWA は予測経路を評価し，最適経路に対応する制御指令を出力する．このアプローチの利点は柔軟性にある．様々なニーズに応じて評価項目を自由に組み合わせることができる．しかし，良い制御効果を得るためには，評価項目間の重みの設定に時間がかかるという問題がある．先行研究で使用された評価関数は heading，dist，vel で構成され，評価値は区間[0,1]までの正規化が行われた．図 3.16 示すように，θは予測経路の終点におけるロボットの向きとゴールの差を表す．先行研究で使用された heading の計算式を式(3.20)に示す．

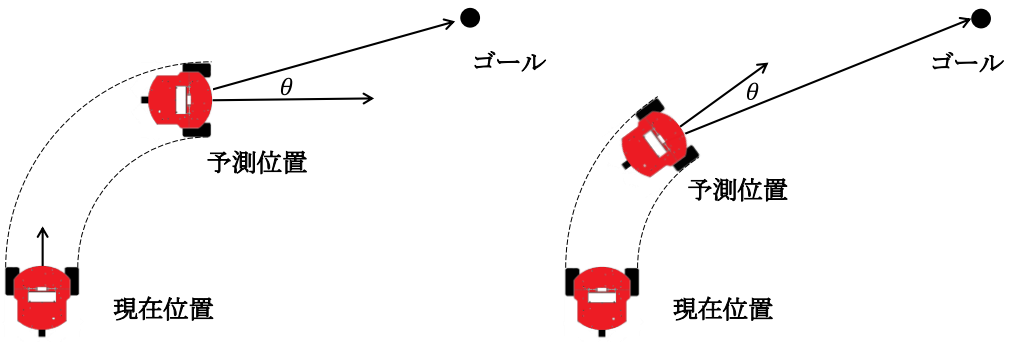


図 3.16　heading の概念図

$$\text{heading} = 1 - \frac{|\theta|}{\pi} \tag{3.20}$$

評価関数 dist は衝突判定に使用される． 図 3.17 に示すように，d_oは各障害物点から予測経路までの距離を表す．dその中の最小値を表す．先行研究では，予測経路と障害物が交差する場合のみの距離を計算する．交差しない場合，評価は大きな定数$\mathrm{N_o}$に設定される．値が 0〜1 に収まるように定数$\mathrm{N_o}$で正規化した dist の計算式を式(3.21)に示す．

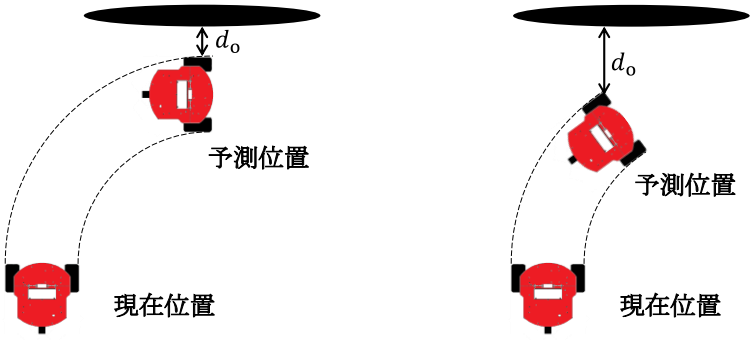


図 3.17　dist の概念図

$$\text{dist} = \begin{cases} \dfrac{d}{\mathrm{N_o}}, \text{予測経路と障害物が交差する} \\ 1\ , otherwise \end{cases} \tag{3.21}$$

関数 vel はロボットの並進速度を評価する．ロボットの移動時間を最小限に抑えたいため，速い速度の評価値が高くなる．vel の計算式は式(3.21)で示す．$v_{\max}$はロボットの最大並進速度を表す．

$$\text{vel} = \frac{v}{v_{\max}} \tag{3.22}$$

関数 dist と vel を最大化した場合，ロボットは常に障害物のない空間内を走行する．heading のみを最大化した場合，ロボットはすぐに進路を遮る最初の障害物に阻まれる．そのため Fox らは評価関数の 3 つの要素すべてが必要だと考えているが，なぜ 3 つの要素を二つの組に分ける理由を説明しなかった．ロボットの初期位置付近は障害物のない空間と仮定し，dist のみを最大化した場合，すべての予測経路の評価値は同様のため，ロボットは初期位置で動かない．vel のみを最大化した場合，ロボットはすぐに進路を遮る最初の障害物に阻まれる．heading のみを最大化した場合，ロボットはすぐに進路を遮る最初の障害物に阻まれる．dist，vel，heading を単独で使用した場合の結果を，障害

物を回避とゴールに移動インセンティブの有無に基づいて

表 3.2 にまとめた．dist と vel を最大化した場合，ゴールに移動するインセンティブを提供しないため，ロボットは常に自由空間で走行する．heading のみを最大化した場合，障害物を回避するインセンティブを提供しないため，ロボットはすぐに進路を遮る最初の障害物に阻まれる．すなわち，タスクを達成するたに評価関数は少なくとも，障害物回避のインセンティブとゴールに移動するためのインセンティブの 2 種類のインセンティブを与える必要がある．

表 3.2　評価項目を単独使用する場合の動作と原因

	動作	障害物を回避するインセンティブ	ゴールに移動するインセンティブ
dist のみ	動かない	あり	なし
vel のみ	阻まれる	なし	なし
heading のみ	阻まれる	なし	あり

3.5.1　基本評価項目

基本評価項目は自律移動タスクを達成するための最小限の評価項目と定義する．回避するインセンティブを提供する評価項目により，ロボットの安全性を保証できる．DWA では，予測経路の点群と障害物の点群の最小距離により衝突判定を行う．最小距離が設定した安全距離より小さい場合，その予測経路は危険とみなし破棄する．ほかには格子地図(Grid Map)に基づいた衝突判定手法も存在する．

衝突判定を行うだけではロボットを移動させるには足りない．それ以外にゴールに移動するインセンティブを提供する必要がある．DWA では，heading によりゴールに移動するインセンティブを提供する．しかし，ゴールがロボットの真正面にある場合，ロボットが動かない可能性がある．ほかには，図 3.18 に示す予測経路の終点からサブゴールまでの距離d_sによりインセンティブを提供する togoal がある．サブゴールに近い経路の評価値が高くなるため，togoal は heading より直接的にゴールに移動するインセンティブを提供できる．togoal の計算式を式(3.23)に示す．ただし，N_sは正規化に使用される定数を表す．

なお，ROS の公式自律移動パッケージ Navigation における，DWA の評価関数は togoal，topath，dist で構成される．その中の topath は経路の到着点とグローバル経路の距離を評価する．これは障害物を回避やゴールに移動するインセ

ンティブを提供しない. 本論文ではこのような評価項目をオプション評価項目と呼ぶ.

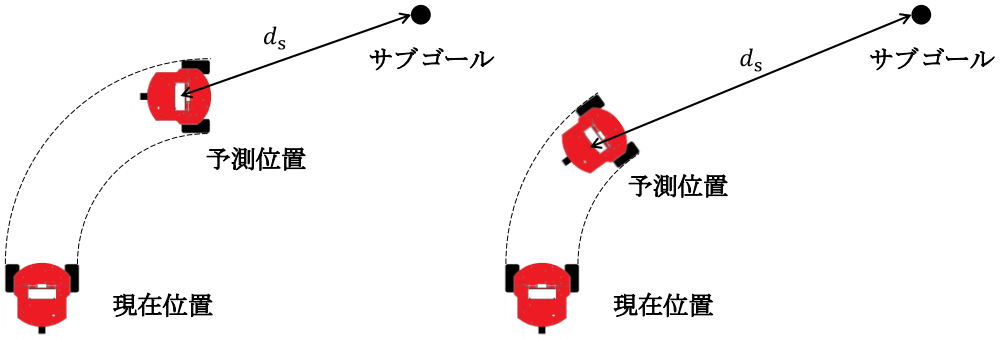


図 3.18　togoal の概念図

$$\text{togoal} = 1 - \frac{d_s}{\mathrm{N_s}} \tag{3.23}$$

3.5.2　オプション評価項目

我々は自律タスクを達成することに加えて, ロボットの実際の表現を改善する評価項目をオプション評価項目と呼ぶ. 例えば, 従来の DWA で使用された vel は大きい並進速度にボーナスを与えることで, ロボットを速く移動させることができる. ロボットの経路が曲折すぎる場合, 大きい角速度にペナルティを与えることで抑えることもできる.

topath は予測経路の到着点とグローバル経路との距離を評価するための関数であるが, これにより, ロボットの実際の経路がグローバル経路に近づくと考えられる.

第 4 章　躍度制御モデルに基づいた DWA の改良

4.1　解決したい課題

DWAは，候補となる速度指令を実行した場合の経路を予測・評価し，最適な速度指令を選択する手法である．DWAはROSでも実装されており，ロボット制御において広く使われている．また，DWAを運転支援や自動運転に応用する研究も存在する[59],[60]．しかし，DWAには二つの問題がある．一つ目は，実現可能な経路の一部しか予測できないという問題である．DWAでは，経路を予測する際に，速度制御モデルを用い，時刻t_1以降に速度が一定であることを仮定している．そのため，加減速を伴う経路は計画することができず，障害物の回避に失敗する場合がある．二つ目の問題は，躍度が制御できないという点である．躍度(Jerk)とは加速度の時間微分である．躍度が大きくなると，力積の増大を招くため，ロボットが転倒したり，内部の部品にダメージを与えたりする[9]-[13]．また，乗り物の場合には，躍度が大きくなると乗客に不快感を与えることも知られている[14]．したがって，躍度を制御できるようになれば，ロボットの安全性や頑健性を高めるだけでなく，感性品質の向上にも寄与する．

障害物回避の性能を向上させるための方法として，Kobayashiらは，Virtual manipulatorsとDWAを併用する手法を提案した[61]．この手法では，二種類のVirtual manipulatorsを生成する．一つはLeader manipulatorsとよばれ，経路追跡を担当する．もう一つは，Assistant manipulatorとよばれ，ロボットと障害物の距離が閾値より小さい場合，障害物と反対方向の速度を提供することで衝突を回避する．この手法を用いることで，DWAでは回避経路が生成できない場合に，Assistant manipulatorにより衝突を避けることができる．ただし，その際にロボットの躍度は制御されていない．

また，躍度制御に関して，Haschkeらは躍度制御モデルによりロボットアームを制御する手法を提案した[62]．この手法では，ロボットアームの運動をいくつかの状態に分け，躍度制御モデルにより2状態間の速度変化を計画する．この手法では，ロボットの動作開始から終了までの一連の動作を把握する必要があり，逐次的に計画を繰り返すローカル経路計画への適用方法については議論されていない．

そこで本章では，躍度で制御可能なDWAを提案する．従来のDWAでは並進速度と角速度を選択するが，提案手法では，並進躍度と角躍度を選択する．また，従来は予測期間中の速度が一定であることを仮定するが，提案手法では，予測期間中の躍度が一定であることを仮定し，速度と加速度には一定を仮定しない．これにより，従来よりも現実に即した経路計画が可能となる．さらに，躍度を抑えた滑らかな加減速を計画することもできる．従来，別々に扱われていた二つの問題を，同時に解決することができる点が，本研究の新規性である．

4.2　提案手法

4.2.1　ロボットの任意の点で生じる躍度

本研究では，差動車輪付きロボットの剛体モデルを対象とする．剛体では，任意な運動を並進運動と回転運動に分解できる[]. 図 4.19を用いてロボット上の任意の点に発生する躍度を考える．X_wとY_wは世界座標系のX軸とY軸を表す．X_rとY_rはロボット標系のX軸とY軸を表す．ロボットの並進速度をv，角速度をω，回転中心をOで表す．差動ロボットのため，並進方向は常にX_r軸の方向と一致する．

このとき発生する並進躍度と角躍度を$j^{\mathrm{V}}[\mathrm{m/s^3}]$と$j^{\Omega}[\mathrm{rad/s^3}]$で表す．角躍度$j^{\mathrm{V}}$は点$O$周りの躍度である．ここで，任意の点$P$で発生する躍度ベクトル$j_P$を考える．点$P$では，$j^{\mathrm{V}}$と方向と大きさが等しい躍度ベクトル$j_P^{\mathrm{V}} = (j^{\mathrm{V}}, 0)$が生じる．また，角躍度$j^{\Omega}$により点$P$では線分$OP$に直行した躍度ベクトル$j_P^{\Omega} = \left(r_P \cdot j^{\Omega} \cdot \sin(\alpha_P), r_P \cdot j^{\Omega} \cdot \cos(\alpha_P)\right)$が生じる．ただし，$r_P$は線分$OP$の長さ，$\alpha_P$は$\mathrm{X}_\mathrm{r}$軸と線分$OP$の角度である．したがって，$j_P$は$j_P^{\mathrm{V}}$と j_P^{Ω}の合成ベクトル

$$\begin{aligned} j_P &= j_P^{\mathrm{V}} + j_P^{\Omega} \\ &= j^{V} + r_P \cdot j^{\Omega} \\ &= \left(j^{\mathrm{V}} + r_P \cdot j^{\Omega} \cdot \sin(\alpha_P), r_P \cdot j^{\Omega} \cdot \cos(\alpha_P)\right) \end{aligned} \tag{4.24}$$

として定義できる．また，j_Pの大きさは式(4.25)で計算できる．

$$|j_P| = \mathrm{J}(j^{\mathrm{V}}, j^{\Omega}, P) = \sqrt{\left(j^{\mathrm{V}} + r_P \cdot j^{\Omega} \cdot \sin(\alpha_P)\right)^2 + \left(r_P \cdot j^{\Omega} \cdot \cos(\alpha_P)\right)^2} \tag{4.25}$$

本研究では，式(4.26)に示すように，ロボット上で発生する最大躍度が，事前に設定した躍度の最大値$j_{\max}$以下になるように速度制御することを目的とする．Qはロボットの外周上の点集合を表す．

$$\max_{P \in Q} \mathrm{J}(j^{\mathrm{V}}, j^{\Omega}, P) \leq j_{\max} \tag{4.26}$$

躍度j^{V}，j^{Ω}が決定したとき，最大躍度が発生するロボット上の点は，ロボットの形状に依存する．例えば，図 4.19のロボットの場合，j^{V}が正，j^{Ω}が負の場合には，図中の点Pで最大躍度が発生するが，j^{V}とj^{Ω}ともに正の場合には点Pの$\mathrm{X_r}$軸対称にある点（すなわち，右車輪前方）で最大躍度が発生する．そこで本研究では，事前にロボットの外周上の点を用いて，j^{V}，j^{Ω}，Pの組み合わせを変えながら$\mathrm{J}(j^{\mathrm{V}}, j^{\Omega}, P)$を計算し，式(4.26)を満たす$j^{\mathrm{V}}$と$j^{\Omega}$の組み合わせのみを抽出し，動作制御に用いる．

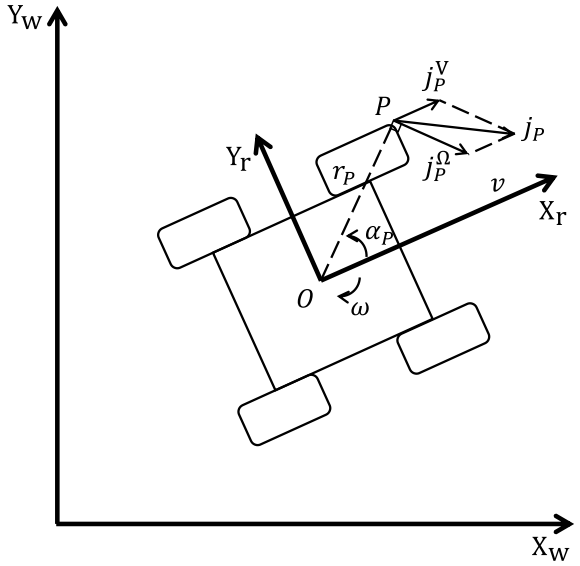


図 4.19　ロボットに発生した加加速度

4.2.2　Dynamic Windowの変更と速度系列の生成

従来のDWAは，予測期間内の速度を不変としていた．本論文では，予測期間内の躍度を不変とし，加速度と速度を可変とした躍度制御モデルを提案する．躍度制御のモデルを説明する前に，まずは，予測期間内の加速度を不変とし，速度を可変とした加速度制御モデルを考える．予測期間内の並進速度系列を$v_1, \ldots, v_n$，角速度系列を$\omega_1, \ldots, \omega_n$で表す．従来手法では式(3.16)と(3.17)に示した加速度制御モデルを使用する．その加速度制御モデルから速度不変の仮定を消すと式(3.18)と(3.19)になる．

提案する躍度制御モデルでは，式(4.26)を満たした躍度の範囲でDWを構成し，DWの中から等間隔に並進躍度と角躍度のペア$(j^{\mathrm{V}}, j^{\Omega})$を選択する．先述の通り，

式(4.26)を満たすか否かはロボットの形状に依存する．図 4.19のロボットのDWは図 4.20に示す形状となる．なお，図中のA'～D'は4.3.2節で説明する．

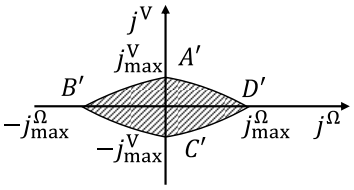
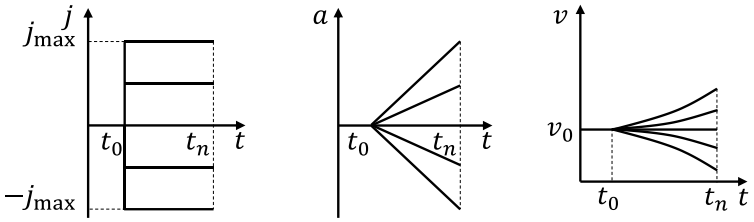


図 4.20　躍度のサンプリング空間

躍度，加速度，速度の関係を図 4.21に示す．提案手法では，予測期間中の躍度は不変と仮定するが，この仮定を置いても，図 4.21 (b)のように加速度は線形に変化するため, 図 4.21 (c)のような非線形な速度系列の生成が可能となる．

選択した並進躍度，角躍度から，式(4.27)～(4.30)を用いて並進速度と角速度の系列を算出する．ただし，現在のロボットの並進加速度をa_0^{V}，角加速度をa_0^{Ω}で表す．この速度系列を用いて位置座標と向きの系列を生成する．

なお，躍度を一定にせずに経路を予測しようとすると，速度系列の総数は，躍度の候補数のn乗となり，実時間での実行が困難となる．



(a) 躍度　　(b) 加速度　　(c) 速度

図 4.21　躍度制御モデル

$$a_n^{\mathrm{V}} = a_{n-1}^{\mathrm{V}} + j^{\mathrm{V}} \cdot \Delta t \tag{4.27}$$
$$a_n^{\Omega} = a_{n-1}^{\Omega} + j^{\Omega} \cdot \Delta t \tag{4.28}$$
$$v_n = v_{n-1} + a_{n-1}^{\mathrm{V}} \cdot \Delta t \tag{4.29}$$
$$\omega_n = \omega_{n-1} + a_{n-1}^{\Omega} \cdot \Delta t \tag{4.30}$$

4.2.3　評価関数

ロボットの自律走行にはできるだけ早く, かつ過度な躍度は発生させたくないという要求がある．すなわち両者はトレードオフの関係にある．そこで，提案手法では, 式(3.4)に示した評価関数に並進躍度と角躍度の評価関数を追加する．提案する評価関数を式(4.34)に示す．並進躍度の評価関数を式(4.31)に，角躍度の評価関数を式(4.32)に示す．また，関数velは障害物回避のインセンティ

ブとゴールに移動するためのインセンティブを提供しないため, 本研究で使用した評価関数にvelは含めなかった.

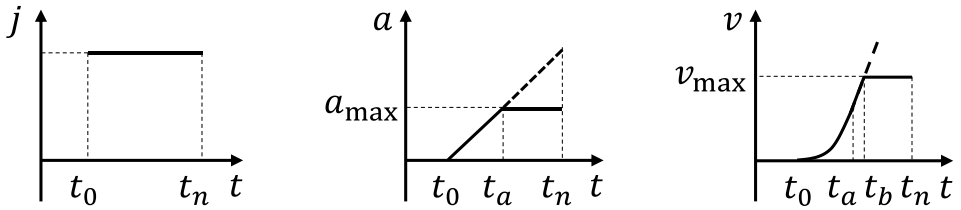
$$\mathrm{jerk}^{\mathrm{V}}(j^{\mathrm{V}})=\begin{cases}1-\dfrac{|j^{\mathrm{V}}|}{j^{\mathrm{V}}_{\max}} & ,|j^{\mathrm{V}}|\le j^{\mathrm{V}}_{\max}\\ -1.0\times10^{4} & ,|j^{\mathrm{V}}|>j^{\mathrm{V}}_{\max}\end{cases} \tag{4.31}$$

$$\mathrm{jerk}^{\Omega}(j^{\Omega})=\begin{cases}1-\dfrac{|j^{\Omega}|}{j^{\Omega}_{\max}} & ,|j^{\Omega}|\le j^{\Omega}_{\max}\\ -1.0\times10^{4} & ,|j^{\Omega}|>j^{\Omega}_{\max}\end{cases} \tag{4.32}$$

$$\mathrm{togoal}(j^{\mathrm{V}},j^{\Omega})=1-\frac{\sqrt{(x_{\mathrm{p}}-x_{\mathrm{g}})^2+(y_{\mathrm{p}}-y_{\mathrm{g}})^2}}{r_{\mathrm{sensor}}} \tag{4.33}$$

$$\mathrm{G}(j^{\mathrm{V}},j^{\Omega})=\alpha\cdot\mathrm{togoal}(j^{\mathrm{V}},j^{\Omega})+\beta\cdot\mathrm{dist}(j^{\mathrm{V}},j^{\Omega})+\gamma\cdot\mathrm{jerk}^{\mathrm{V}}(j^{\mathrm{V}})+\delta\cdot\mathrm{jerk}^{\Omega}(j^{\Omega}) \tag{4.34}$$

まず，式(4.31)と(4.32)の躍度の評価関数について説明する．この式では，躍度は0に近いほど評価値が高く, 躍度の絶対値が規定値$j^{\mathrm{V}}_{\max}, j^{\Omega}_{\max}$超える場合に，ペナルティとして大きな負の値を与える．躍度はDWの範囲で選択されるが，予測期間中にこの範囲を超える場合がある. これは加速度や速度にはハードウェア上の上限が存在するためである．例えば，図 4.22 (a)に示すように，正の躍度jが選択された場合，予測期間t_0~t_nで躍度が一定として，各時刻の加速度と速度が計算される．この時，図 4.22(b)のように，時刻t_aで加速度が上限に達すると，それ以降の加速度は$a_{\max}$になる．また，図 4.22 (c)のように，時刻t_bで速度が上限に達すると，それ以降の速度は$v_{\max}$になる．このように，加速度や速度が上限に達すると, 当初選択した躍度とは異なる負の躍度が発生することになる．そこで，この評価関数で用いる躍度jには，当初選択した躍度ではなく，予測時に発生した躍度のうち絶対値が最大のものを用いる．



(a) 躍度　　(b) 加速度　　(c) 速度

図 4.22　速度や加速度が上限に達する場合

本研究で用いる$\mathrm{togoal}(j^{\mathrm{V}},j^{\Omega})$の定義を式(4.33)に示す．$j^{\mathrm{V}},j^{\Omega}$を選択した場合の到着点を$x_{\mathrm{p}},y_{\mathrm{p}}$，局所的目標の座標を$x_{\mathrm{g}},y_{\mathrm{g}}$で表す．到着点と局所的目標の座

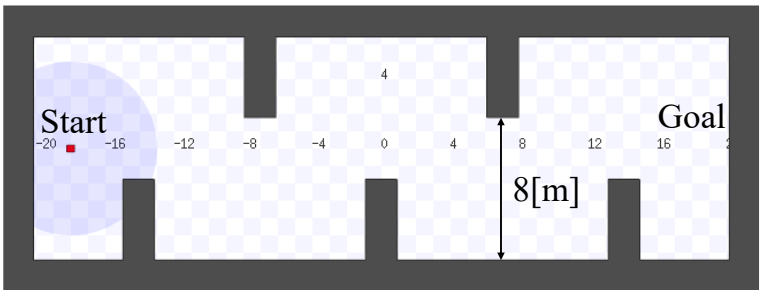
標を用い，式(3.23)のd_sを算出する．N_sにはセンサーの計測距離r_{sensor}を使用する．後述の実験では，A-starを用いて初期位置から大局的目標までのグローバル経路を計画しており，その経路を辿るように局所的目標を設定する．

4.3 実験

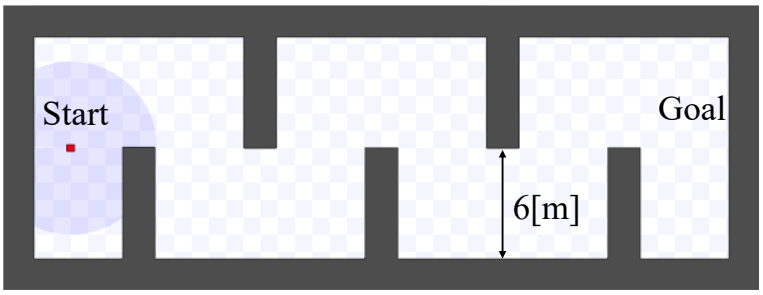
4.3.1 実験条件

従来手法の加速度を小さく設定することで，躍度を抑えることが可能だが，走行時間が非常に長くなるという問題がある．そのため，本研究では，主に躍度を制御する効果と走行時間から，提案手法の有効性を評価する．制御プログラムはROSを用いて構築する．本論文では，Stage[63]を用いたシミュレーション実験（実験1～3）と，実機実験（実験4）を行う．実験1では，速度制御モデル, 加速度制御モデルと躍度制御モデルで生成される経路の到着点の範囲を比較する. 実験2では, 図 4.23に示す環境で走行中の躍度を比較する. 実験3では, 式(20)に示した評価関数における躍度の重みの違いが与える影響を考察する．実験4では躍度制御モデルの効果を実機で検証する．ロボットにはMobileRobots社のPioneer-3ATを用いる．ロボットの距離センサーには，北陽電機株式会社製のレーザーレンジファインダURG-04LXを使用する．使用したロボットと走行環境を図 4.24に示す．写真中央の柱の手前約2[m]をロボットの初期位置，柱の奥約2[m]を目標位置とする．

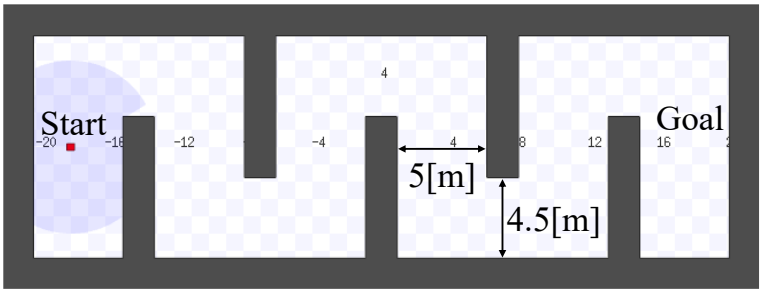
シミュレーション用のロボットのパラメータは実機実験で用いるPioneer-3ATを模して設定する．各パラメータを表 4.3に示す．最大速度，加速度についてはPioneer-3ATの仕様上の最大値を設定している. 実機実験では安全性のため, 最大速度と加速度を仕様限界よりも低く設定した．Pioneer-3ATの加速度の限界値から，発生する躍度の最大値は53[m/s^3]となる．シミュレーション実験では，発生する躍度の最大値を0.5[m/s^3]に，実機実験では，発生する躍度の最大値を0.3[m/s^3]に抑えることを目標とする．ロボットの制御周期は0.1[s]，DWAによる予測期間は2.0[s]，サンプリング時間Δtは0.1[s]に設定する．速度制御モデル，加速度制御モデル，躍度制御モデルともに，1回の経路予測で25本の経路を生成できるように，DWを縦横に当分割する．評価関数の重み$\alpha, \beta, \gamma, \delta$はそれぞれ1.0，1.0，0.1，0.1に設定した．なお，γとδの適切な値については実験3で調査している． 速度制御モデル加速度制御モデル



(a) 環境 1



(b) 環境 2



(c) 環境 3

図 4.23　シミュレーション環境

(a) ロボットの様子

(b) 環境の様子

図 4.24　実環境

表 4.3　ロボットのパラメータ

	シミュレーション環境 (実験 1 - 3)	実環境 (実験 4)
寸法 [m]	0.50×0.49×0.26	
最大躍度 [m/s^3]	0.5	0.3
最大並進速度 [m/s]	2.0	0.6
最大角速度 [rad/s]	6.28	2.27
最大並進加速度[m/s^2]	1.0	0.3
最大角加速度[rad/s^2]	5.24	2.27
計測範囲 [deg]	360	
角度分解能 [deg]	0.5	

4.3.2　実験 1：生成される経路の到着点の比較

本実験では，各モデルで予測可能な到着点の範囲を比較する．ロボットが到着可能な理論上の範囲は，予測期間での速度の変化に制限を付けない場合の到着点の範囲である．本実験で比較する３つのモデルは，それぞれが速度の変化に制限を付けている．加速度制御モデルでの到着点の範囲は，ロボットの速度と加速度制限のみに影響されるため，一番広い範囲になる．　ロボットの初期位置を(0,0)，現在の並進速度を1.0[m/s]，角速度を0.0[rad/s]とすると，表 4.3の速度制約を持つロボットが1.0[s]，1.5[s]，または2.0[s]で到達可能な範囲は表4.4の青線内となる．また，速度制御モデルと躍度制御モデルで予測できる到達点の範囲をそれぞれ黒線と赤線で示す．図中のA～DおよびA'～D'は，図 3.10および図 4.20で示したDWの4点である．参考のため，最大躍度を0.5[m/s^3]とした場合の各到着点での速度(v_N, ω_N)を表 4.5に示す．

まず表 4.4の青線と黒線を比較すると，黒線の範囲が狭いことが確認できる．これは，等速での経路のみ予測可能であり，加速や減速を伴う経路が予測できないためである．

一方で躍度制御モデルは，表 4.4に示すように最大躍度が小さく，さらに予測期間を短く設定した場合には，速度制御モデルよりも狭い範囲の経路しか予測できない．これは躍度制御モデルでは，躍度を制限するため，加減速の切り替え時（実験1の場合は予測開始直後）の速度変化は小さくなるためである．しかし，予測期間を長くする，または最大躍度を大きくすることで，予測可能な到達点の範囲が速度制御モデルよりも拡大する．また，表 4.4のAやDに到達するためには，前進しながら回転する必要があるが，その際に0.5[m/s^3]を超える躍度が発生する．　躍度制御モデル

躍度制御モデルで設定する最大躍度・予測時間と，予測可能な到達点の範囲の関係を図 4.22に示す．予測可能な到達点の範囲は，実際に到達可能な範囲（表 4.3の点線の枠内の面積）との比率を示している．最大躍度を0.5[m/s^3]に設定した場合，予測期間が1.7[s]以上になると速度制御モデル（図中の黒線）よりも予測範囲が広くなる．最大躍度を2.0[m/s^3]に設定した場合には，予測期間が1.6[s]以上で速度制御モデルの予測範囲を超え，2.0[s]では実際に到達可能な範囲の2割程度を予測可能になることが確認できる．

表 4.4　到着点の範囲の比較

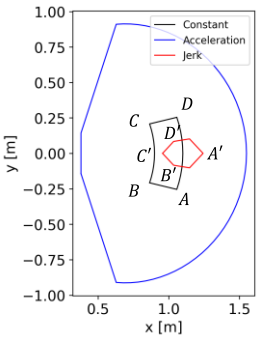
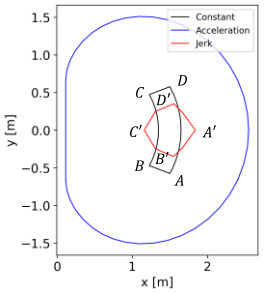
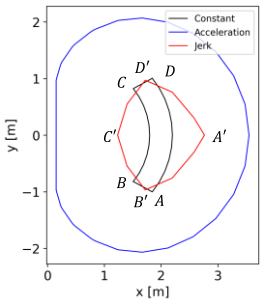
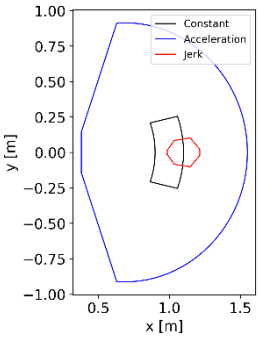
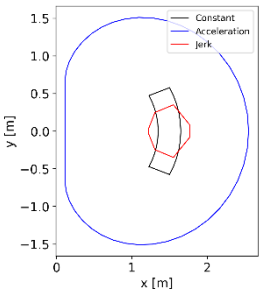
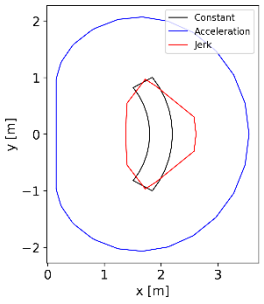
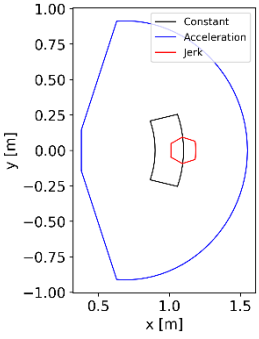
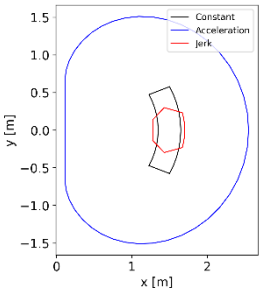
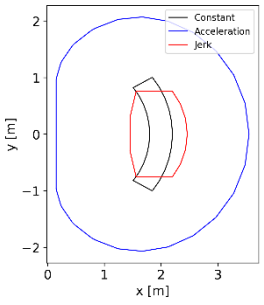
予測期間 / 最大躍度	1.0[s]	1.5[s]	2.0[s]
0.5[m/s^3]			
1.0[m/s^3]			
1.5[m/s^3]			

表 4.5　到着点での速度

予測期間	1.0[s]		1.5[s]		2.0[s]	
A/A'	(1.1, -0.5)	(1.1, 0.0)	(1.1, -0.5)	(1.6, 0.0)	(1.1, -0.5)	(2.0, 0.0)
B/B'	(0.9, -0.5)	(1.0,-0.4)	(0.9, -0.5)	(1.0,-1.6)	(0.9, -0.5)	(1.0,-2.8)
C/C'	(0.9, 0.5)	(0.9, 0.0)	(0.9, 0.5)	(0.4, 0.0)	(0.9, 0.5)	(0.0, 0.0)
D/D'	(1.1, 0.5)	(1.0,0.4)	(1.1, 0.5)	(1.0,1.6)	(1.1, 0.5)	(1.0,2.8)

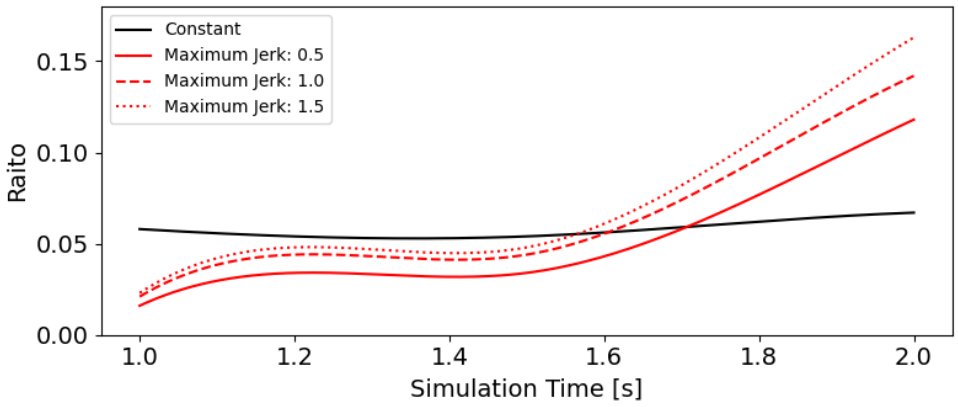


図 4.25　各モデルで生成される到達点範囲の理論値に対する割合

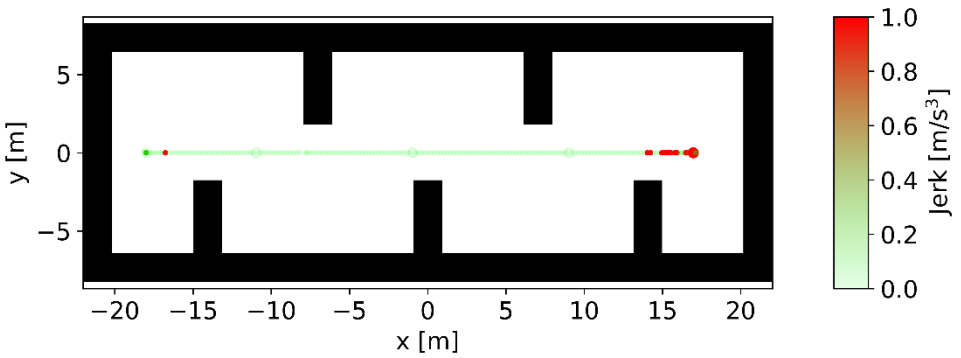
4.3.3　実験 2：躍度制御効果の検証

図 4.23に示した環境を用いて，躍度制御の効果を検証する．環境1は障害物を回避する必要がない環境，環境2は蛇行が必要な環境，環境3はより大きな蛇行が必要な環境として設計した．すべての環境で両手法の予測時間は2.0[s]に設定した．なお，図 4.23の座標の単位は[m]であり，ロボットの初期位置を(-18,0)に，大局的目標の位置を(18,0)に設定する．

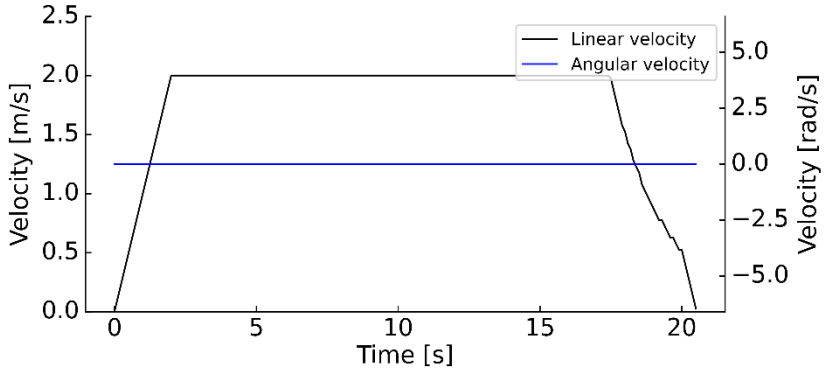
環境1で得られた経路を図 4.26(a)，図 4.27(a)，図 4.28(a)に示す．図では，経路上で発生した躍度を色で示している．後述の速度変化のグラフと対応が取りやすいように，5[s]ごとに大きな点をプロットしている．速度指令を図 4.26(b)，図 4.27(b)，図 4.28(b)に示す．速度指令は，ロボットの幾何学的中心の速度と考えることができる．ロボット上で最大躍度（最大躍度が等しい場合は最大加速度）が発生した箇所は両側のため，左側の速度，加速度，躍度を図 4.26(c)，図 4.27(c)，図 4.28(c)に示し，右側の速度，加速度，躍度を図 4.26(d)，図 4.27(d)，図 4.28(d)に示す．

図 4.26(a)，図 4.27(a)，図 4.28(a)に示すように，環境1では3つのモデルとも障害物を回避する必要がないため，経路はほぼ直線となった．速度制御モデルと加速度制御モデルでは，図 4.26(b)と図 4.27(b)に示すように，角速度指令は

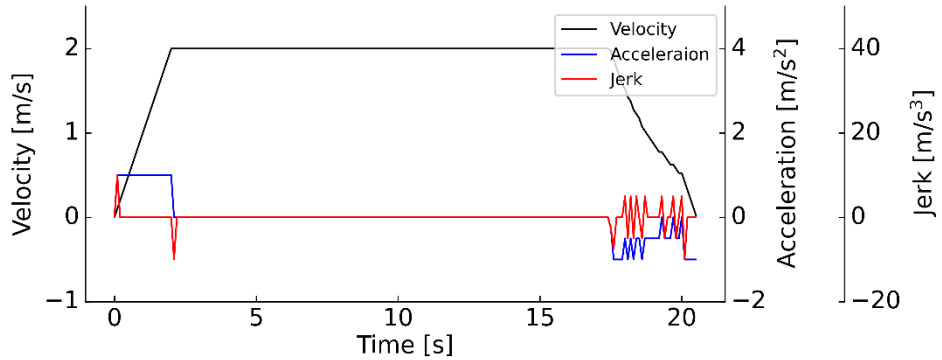
常に0[rad/s]のため，ロボットは単純に直線運動する．加速開始の2.5[s]後，最大速度に達した．途中の2.5，17.5[s]付近で速度変化が発生し，大きい躍度が発生した．躍度の最大値は10.0[m/s^3]であった．躍度制御モデルでは，図 4.28 (a)に示すように，ゴール付近で回転運動が発生した．これはゴール付近で急停止しない経路が選択された結果である．図 4.28 (c)と(d)に示すように，最大躍度は常に±0.5[m/s^3]の範囲に制御できていた．



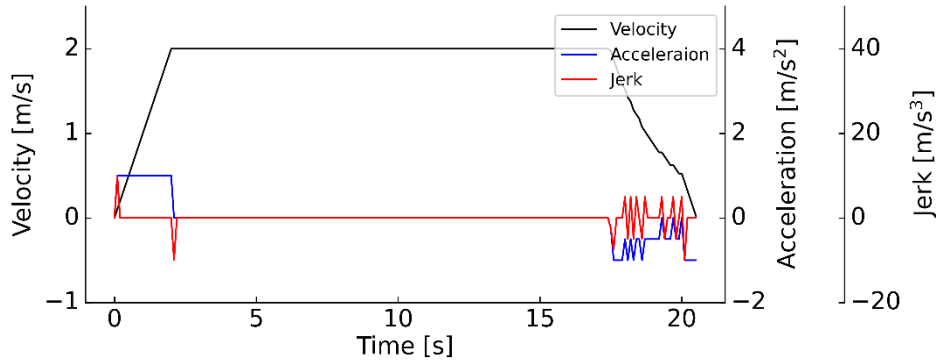
(a) 移動経路



(b) 速度の命令値

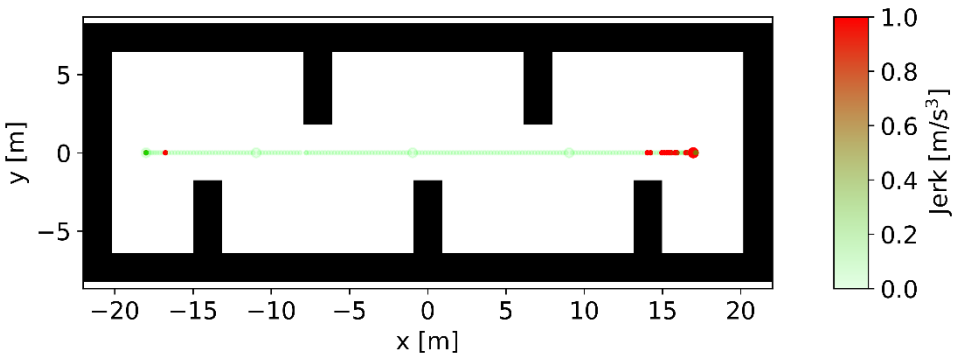


(c) ロボットの左側で発生した速度，加速度と躍度

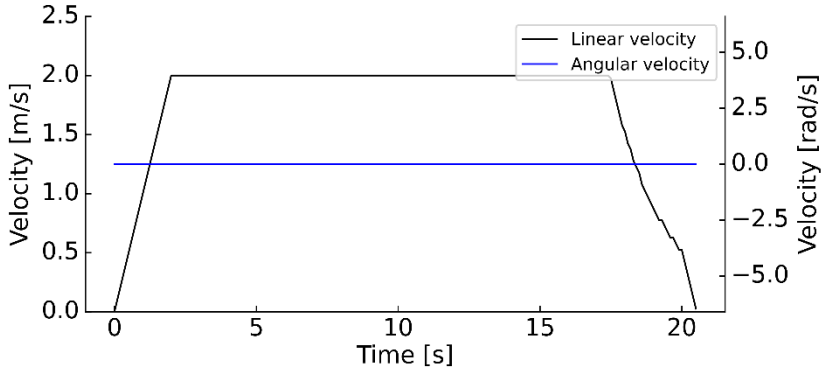


(d) ロボットの右側で発生した速度，加速度と躍度

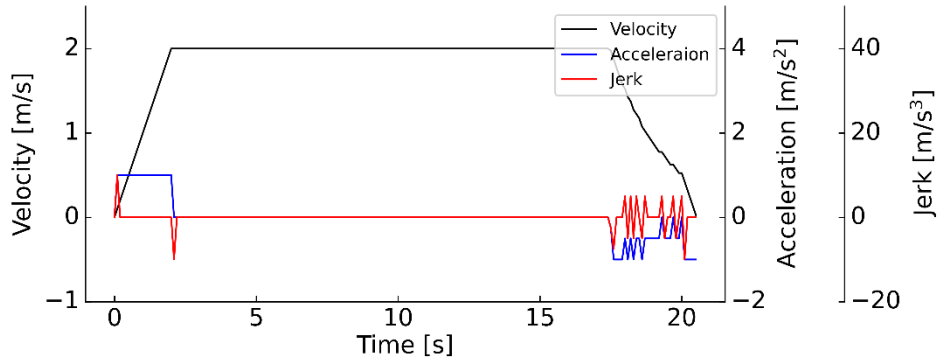
図 4.26　環境 1 における速度制御モデルでの結果



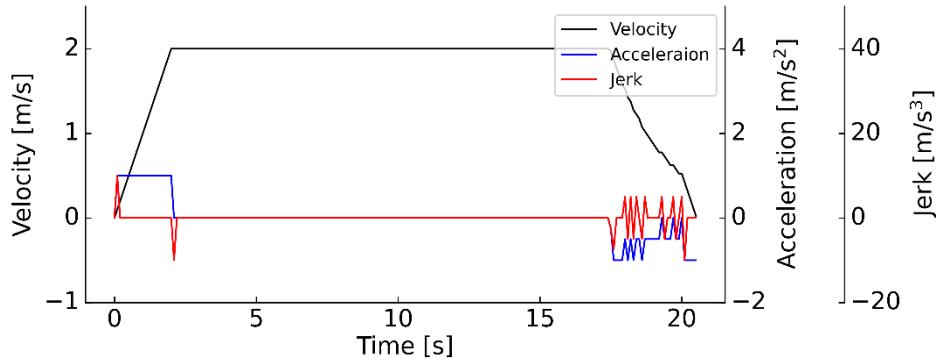
(a) 移動経路



(b) 速度の命令値

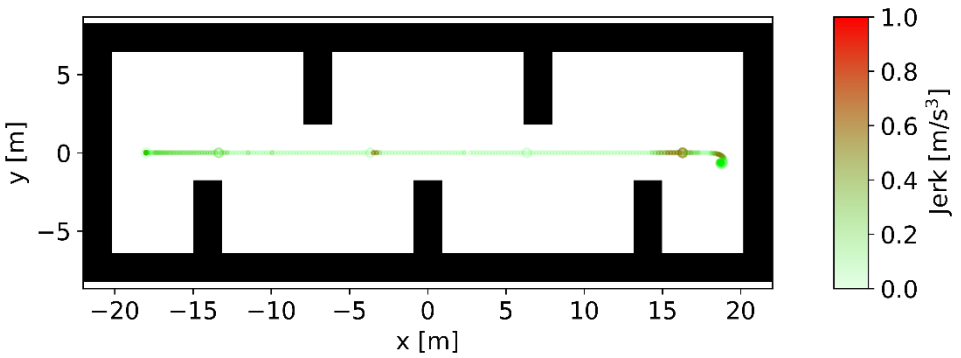


(c) ロボットの左側で発生した速度，加速度と躍度

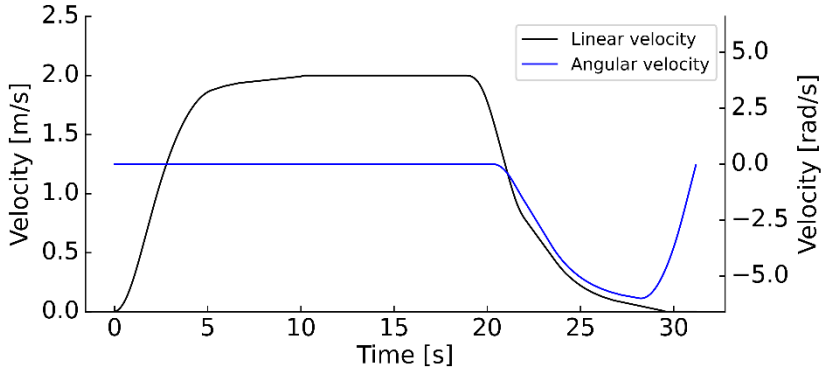


(d) ロボットの右側で発生した速度，加速度と躍度

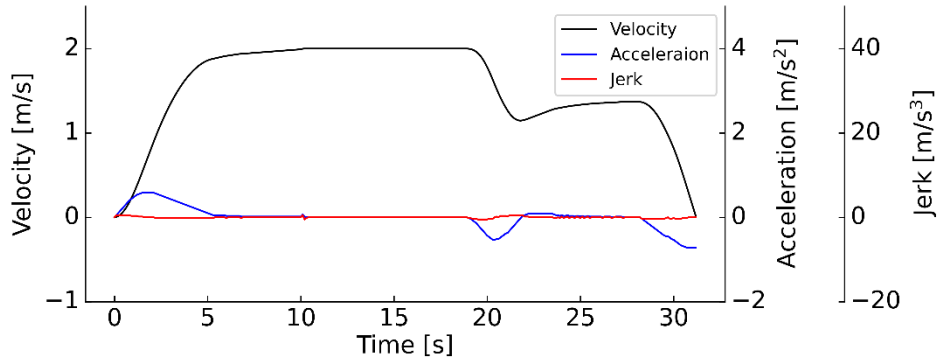
図 4.27　環境 1 における加速度制御モデルでの結果



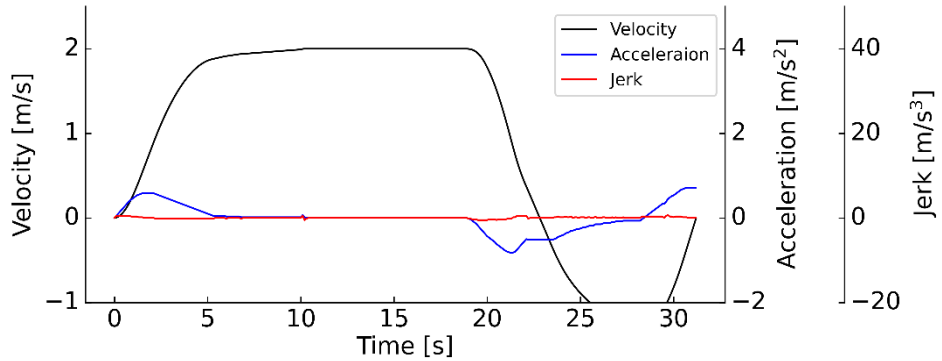
(a) 移動経路



(b) 速度の命令値



(c) ロボットの左側で発生した速度，加速度と躍度

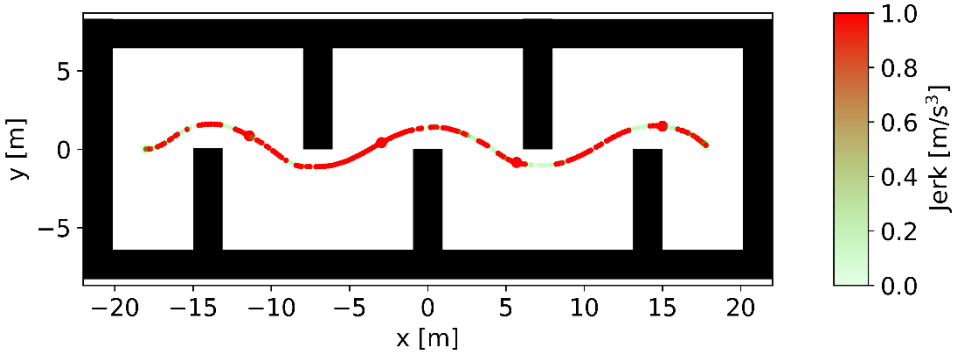


(d) ロボットの右側で発生した速度，加速度と躍度

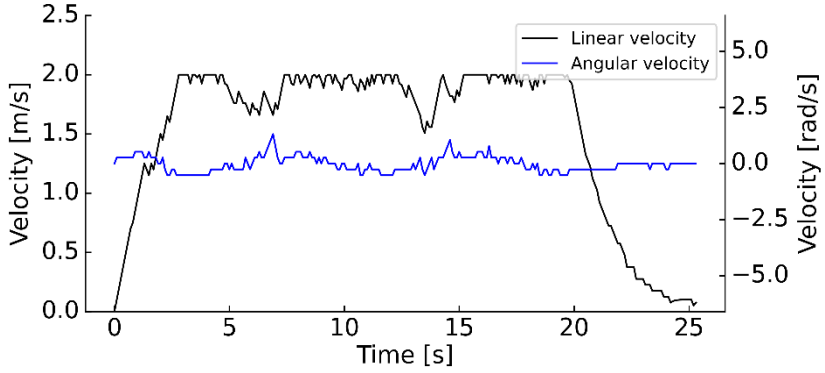
図 4.28　環境 1 における躍度制御モデルでの結果

次に，環境2の結果を図 4.29～図 4.31に示す．速度制御モデルと加速度制御モデルでは，障害物を回避するために速度が大きく変化しており，20[m/s^3]程度の躍度が複数回発生し，最大躍度は43[m/s^3]になった．また，加速度制御モデルでは，加減速しながら障害物を回避する経路を生成できるため，速度制御モデルのような歯状の速度変化が発生しない．大きい加加速度の発生頻度も低くなった．一方で，躍度制御モデルでは，軽微な速度変化により障害物を回避できており，最大躍度も常に±0.5[m/s^3]の範囲に収まっていた．

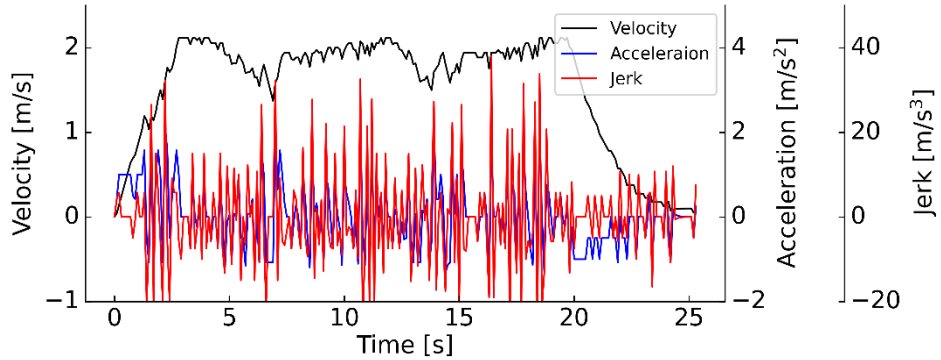
最後に環境3の結果を図 4.32～図 4.34に示す．速度制御モデルと加速度制御モデルでは，速度変化の幅と頻度が環境2よりも大きく，最大躍度は37[m/s^3]になった．速度制御モデルと比べ加速度制御モデルで大きい加加速度の発生頻度は低くなった．躍度制御モデルでは，これまでの結果と同様に，最大躍度が常に±0.5[m/s^3]の範囲に収まっていた．



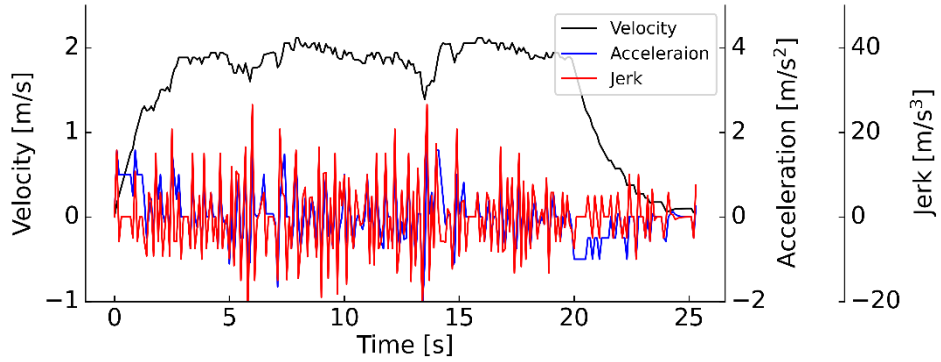
(a) 移動経路



(b) 速度の命令値

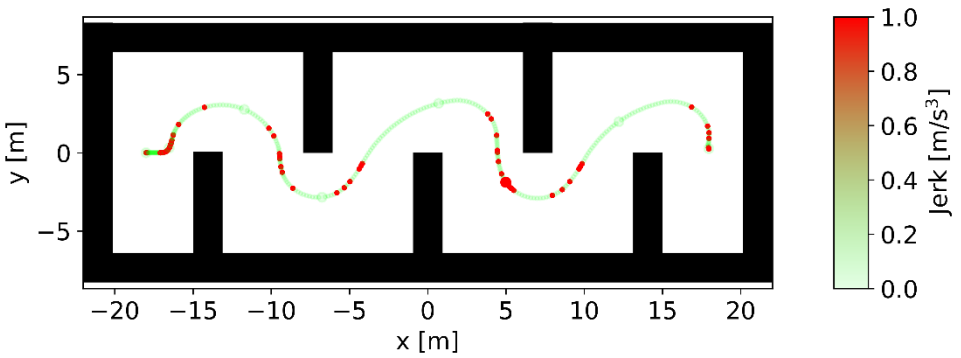


(c) ロボットの左側で発生した速度，加速度と躍度

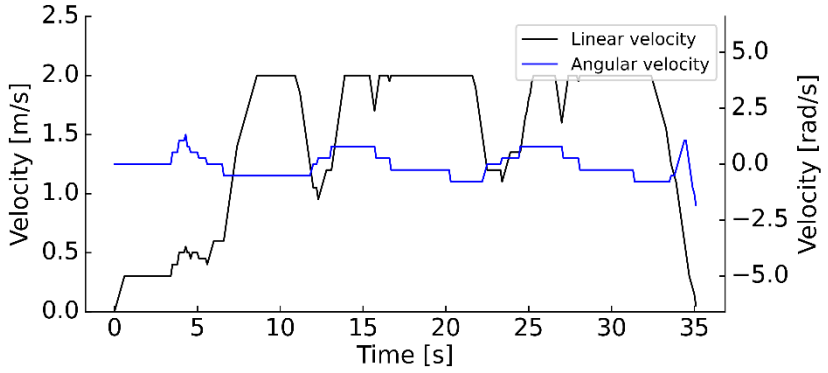


(d) ロボットの右側で発生した速度，加速度と躍度

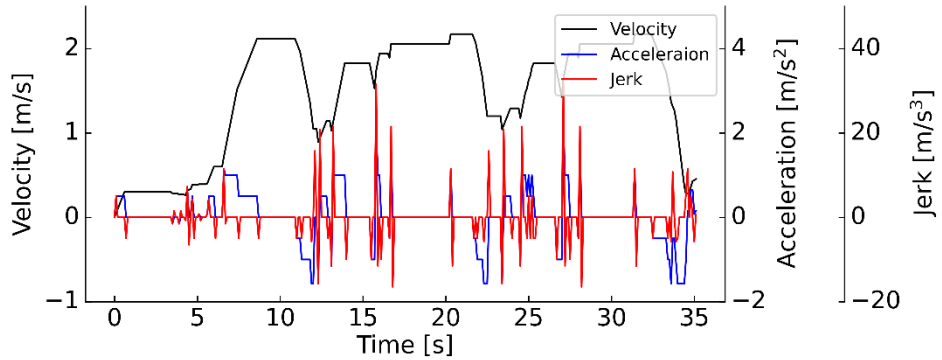
図 4.29　環境 2 における速度制御モデルでの結果



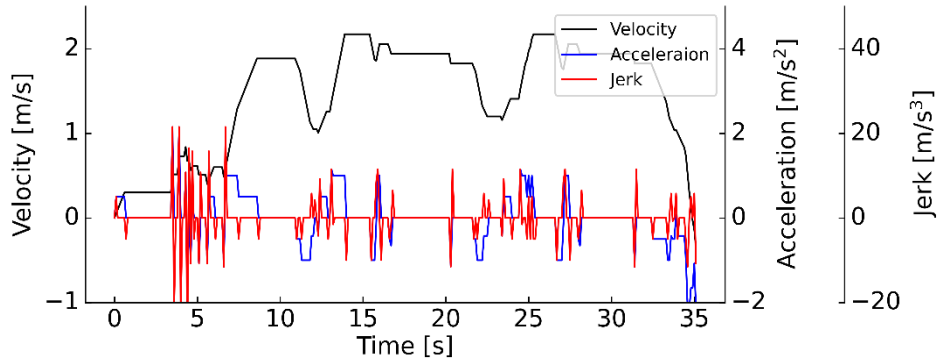
(a) 移動経路



(b) 速度の命令値

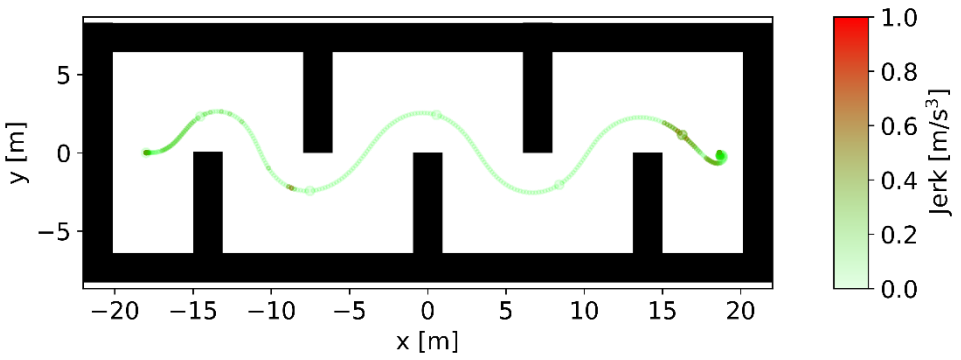


(c) ロボットの左側で発生した速度，加速度と躍度

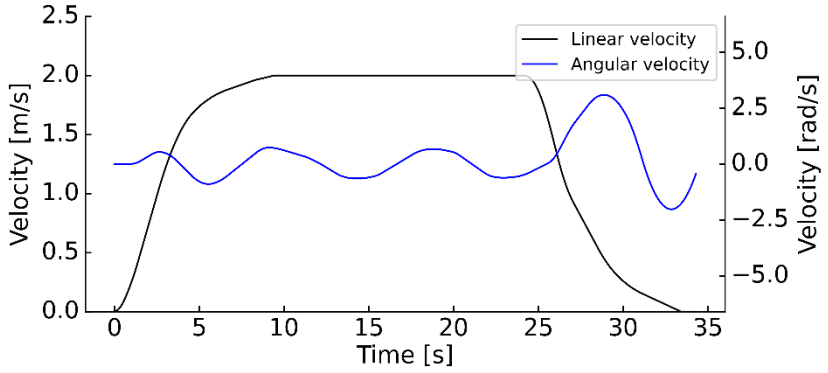


(d) ロボットの右側で発生した速度，加速度と躍度

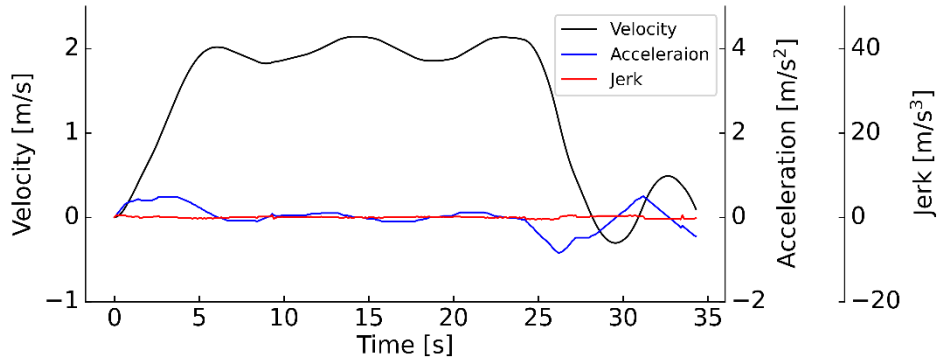
図 4.30　環境 2 における加速度制御モデルでの結果



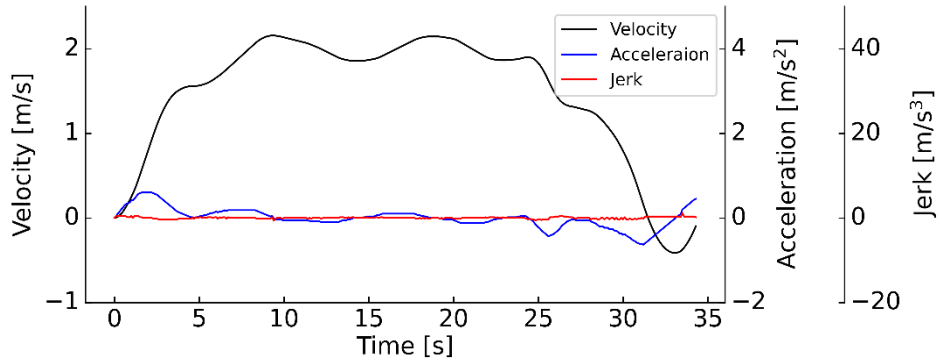
(a) 移動経路



(b) 速度の命令値

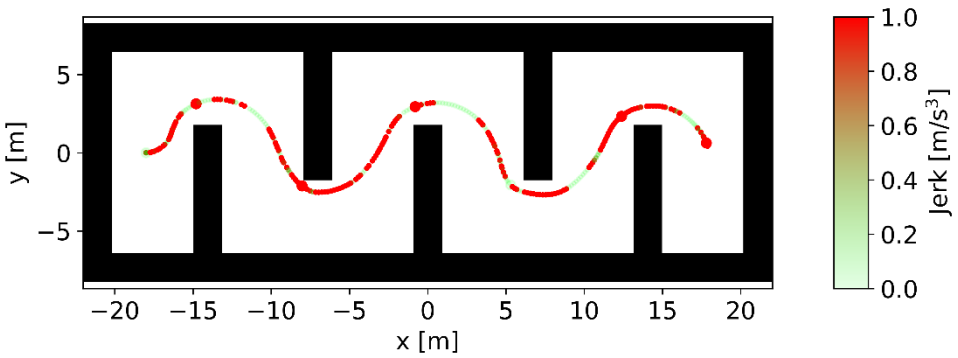


(c) ロボットの左側で発生した速度，加速度と躍度

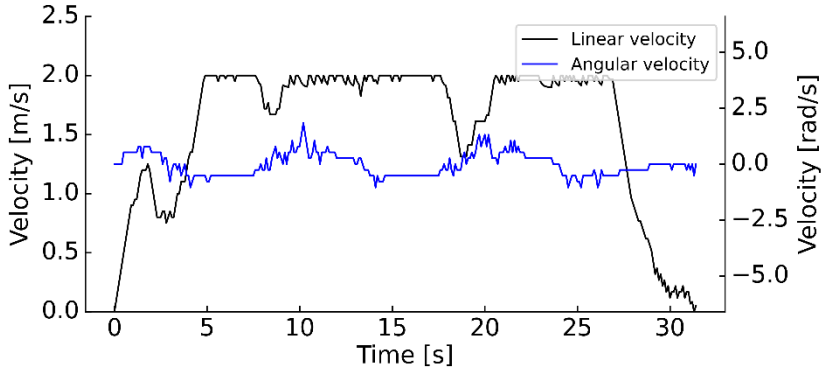


(d) ロボットの右側で発生した速度，加速度と躍度

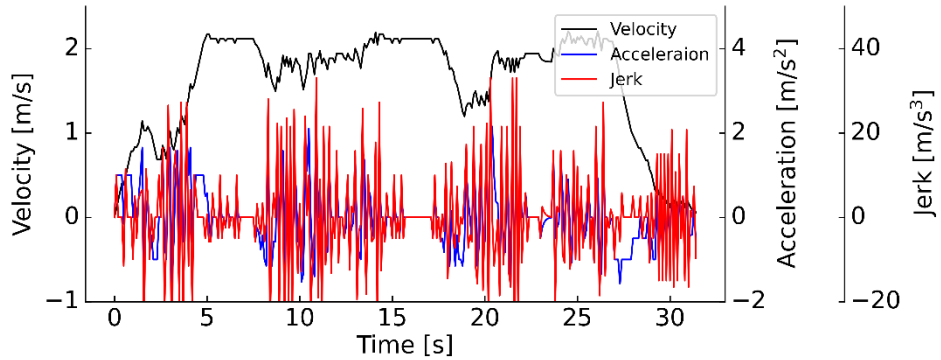
図 4.31　環境 2 における躍度制御モデルでの結果



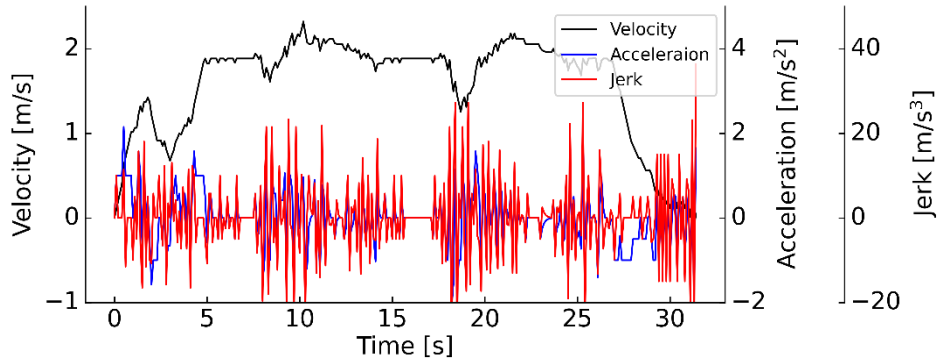
(a) 移動経路



(b) 速度の命令値

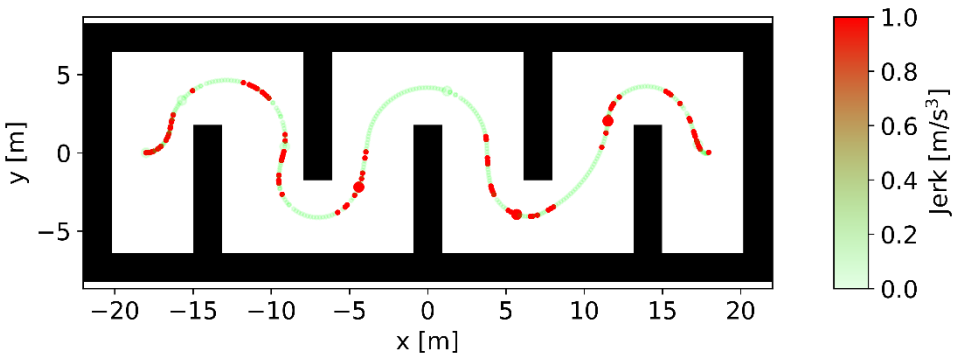


(c) ロボットの左側で発生した速度，加速度と躍度

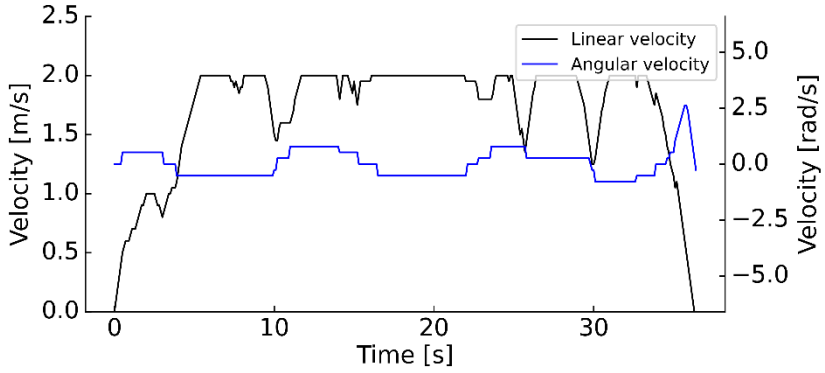


(d) ロボットの右側で発生した速度，加速度と躍度

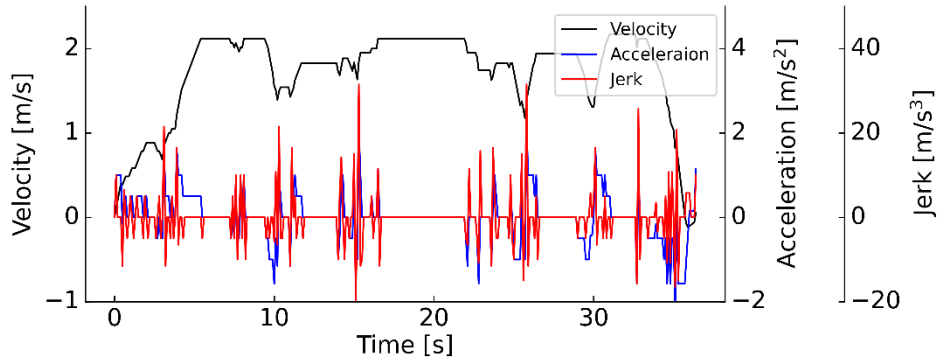
図 4.32　環境 3 における速度制御モデルでの結果



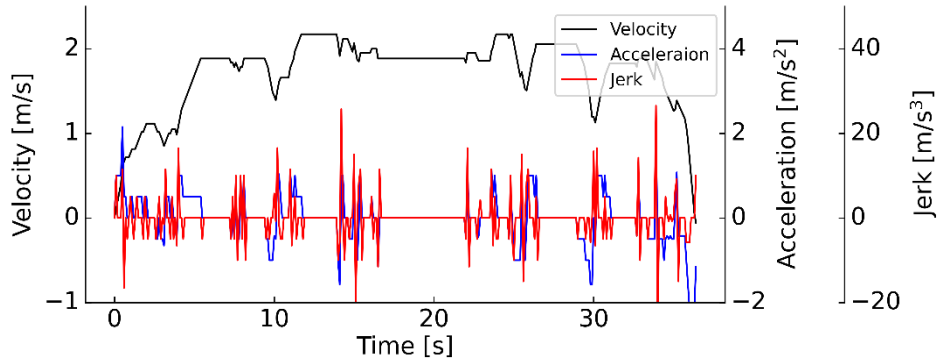
(a) 移動経路



(b) 速度の命令値

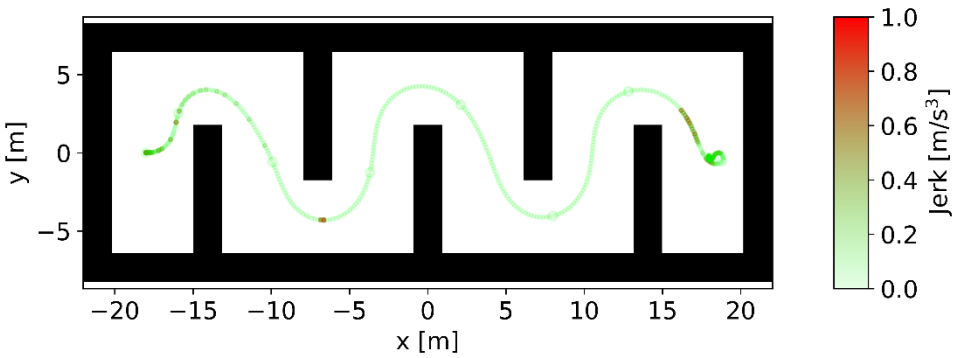


(c) ロボットの左側で発生した速度，加速度と躍度

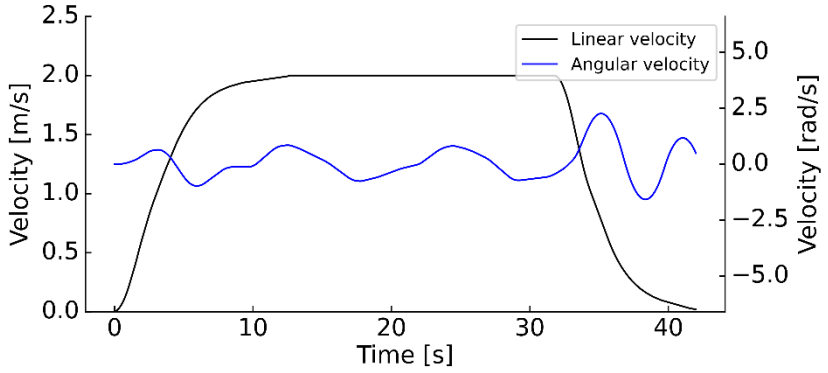


(d) ロボットの右側で発生した速度，加速度と躍度

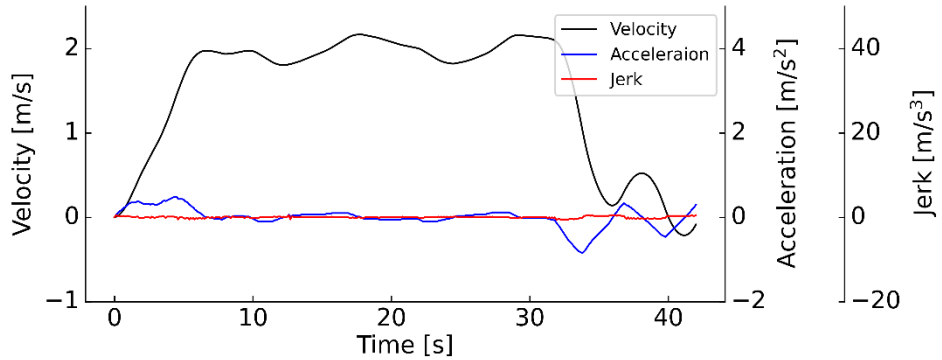
図 4.33　環境 3 における加速度制御モデルでの結果



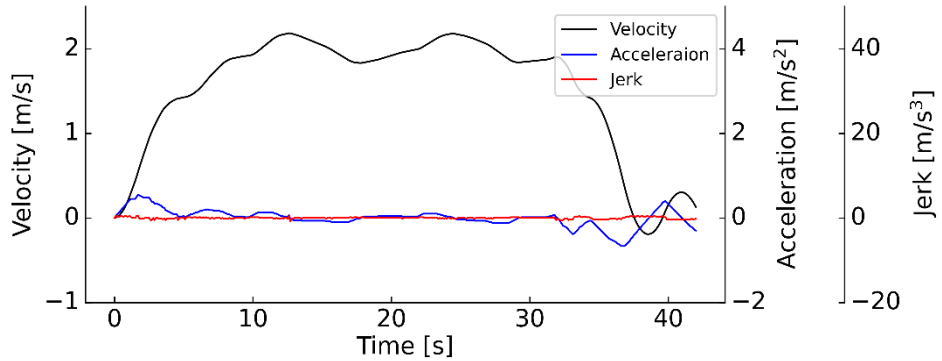
(a) 移動経路



(b) 速度の命令値



(c) ロボットの左側で発生した速度，加速度と躍度



(d) ロボットの右側で発生した速度，加速度と躍度

図 4.34　環境 3 における躍度制御モデルでの結果

ここまで，躍度制御モデルによる躍度の抑制効果を検証したが，躍度制御モデルには加減速過程では速度制御モデルや加速度制御モデルよりもロボットの走行時間が長くなるというデメリットがある．そこで，走行時間を定量的に比較する．ロボットは，x軸方向に移動することから，ロボットのx座標を基準に走行期間を三つに分けて，その間の走行時間を比較する．初期位置から$x = -15$までを「期間1」，$x = -15$から$x = 15$までを「期間2」，$x = 15$から大局的目標までを「期間3」とする．「期間1」は動作開始時の加速，「期間3」は大局的目標付近の減速，「期間2」は中間を表している．

各環境で三つのモデルの実験を各10回行い，走行時間の平均値に有意な差があるかを，等分散を仮定しないt検定により確認した．結果を表 4.6，表 4.7と表 4.8に示す．表中の「**」は有意水準0.01，「*」は有意水準0.05で有意であることを意味する．躍度制御モデルでの走行時間が短く，有意差がある場合に緑色で，躍度制御モデルでの走行時間が長く，有意差がない場合に黄色で表す．

速度制御モデルと加速度制御モデルは急加速や急減速を行わないため，環境1における，動作開始直後の「期間1」や，停止直後の「期間3」では，躍度制御モデルよりも走行時間が長くなる．しかし，「期間2」においては速度制御モデルとの差が小さい．環境2では逆に躍度制御モデルの方が，加速度制御モデルと比べ走行時間が短くなっている．これは図 4.30と図 4.31からわかるように，速度制御モデルは速度変化の幅が大きく無駄な加減速を行っているのに対し，躍度制御モデルでは速度変化の幅が小さく，無駄な加減速を行わずに障害物の回避が実現できている．

また，環境3では躍度制御モデルと速度制御モデルの走行時間の有意差はなくなった．このことから，躍度制御モデルでは，最大躍度を制限できるだけでなく，無駄な速度変化を抑制することで，最大速度で走破出来ないような複雑な環境では，速度制御モデルと加速度制御モデルよりも走行時間を短縮できる可能性があると言える．

表 4.6　環境 1 における走行時間の比較

	期間 1	期間 2	期間 3	全体
速度制御モデル	2.9	15.1	3.2	21.2
躍度制御モデル	4.0 (+1.1)	15.2 (+0.1)	11.2 (+8.0)	30.5 (+9.3)
有意差	**	*	**	**
加速度制御モデル	3.0	15.0	2.6	20.0
躍度制御モデル	4.0 (+1.0)	15.2 (+0.2)	11.2 (+8.6)	30.5 (+10.5)
有意差	**	*	**	**

表 4.7　環境 2 における走行時間の比較

	期間 1	期間 2	期間 3	全体
速度制御モデル	3.0	16.7	3.7	23.4
躍度制御モデル	4.6 (+1.6)	19.5 (+2.8)	9.0 (+5.3)	33.1 (+9.7)
有意差	**	**	**	**
加速度制御モデル	7.4	22.9	3.8	34.1
躍度制御モデル	4.6 (-2.8)	19.5 (-3.4)	9.0 (+5.2)	33.1 (-1.0)
有意差	**	**	**	

表 4.8　環境 3 における走行時間の比較

	期間 1	期間 2	期間 3	全体
速度制御モデル	5.5	24.2	3.8	33.5
躍度制御モデル	5.9 (+0.4)	25.2 (+1.0)	14.8 (+11.0)	45.9 (+12.4)
有意差			**	
加速度制御モデル	6.8	29.3	4.1	40.2
躍度制御モデル	5.9 (-0.9)	25.2 (-4.1)	14.8 (+10.7)	45.9 (+5.7)
有意差	**	**	*	

4.3.4　実験 3：評価関数における躍度の重みの影響

図 4.23(b)に示す環境を用いて，式(4.34)に示した評価関数における並進躍度の重みγと角躍度の重みδの違いが与える影響を考察する．並進躍度や角躍度の重みを0にすると，式(4.31)と(4.32)に示したペナルティがなくなるため，図 4.35に示すように，最大躍度を超えた経路が生成される．

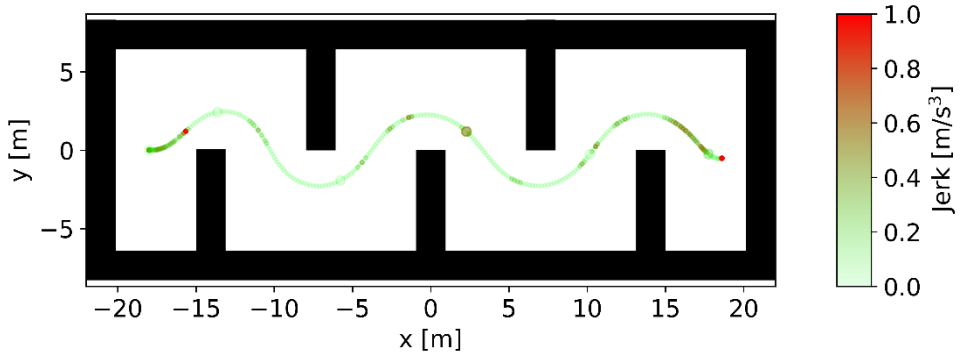


図 4.35　躍度を制御できない例($\gamma = 0,\ \delta = 0$)

また，重みを大きくすると，ロボットは目標への移動や障害物回避よりも，躍度の上昇によるペナルティの大きさが勝り，初期位置で停止したままとなったり，障害物回避ができなくなったりする．本実験条件の場合，図 4.36に示すように，γが0.20の場合，ロボットは初期位置で停止したままとなる．また，図 4.37に示すように，δが0.16を超えると目標への到達が困難となり，図 4.38に示すように，δが0.20を超えると最初の壁も回避できなくなる．

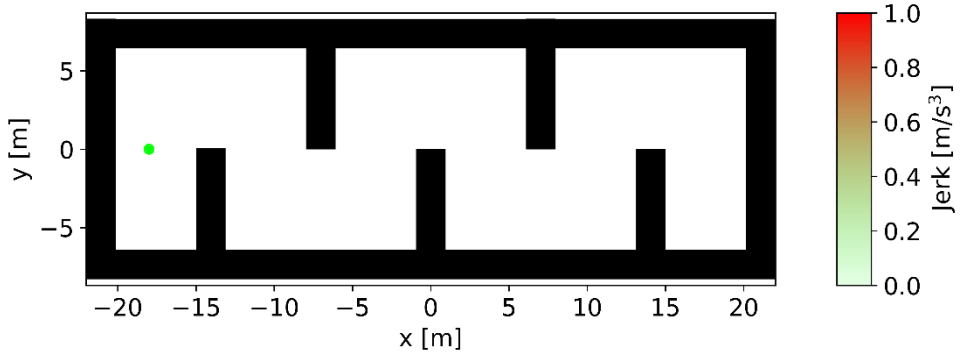


図 4.36　自動走行の失敗例($\gamma = 0.20, \delta = 0.10$)

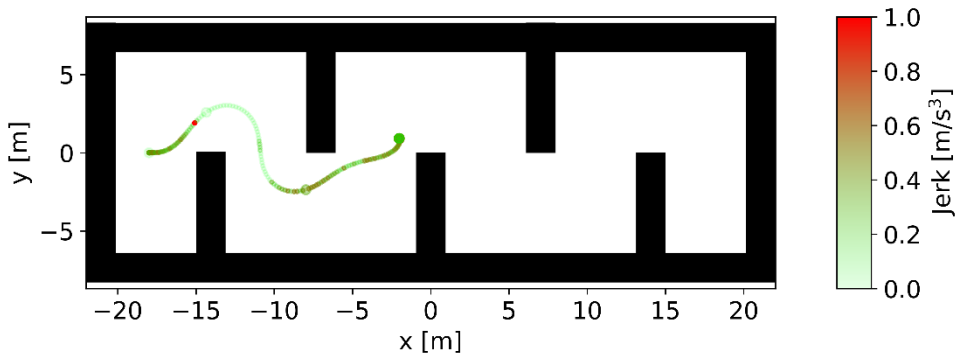


図 4.37　自動走行の失敗例($\gamma = 0.10, \delta = 0.16$)

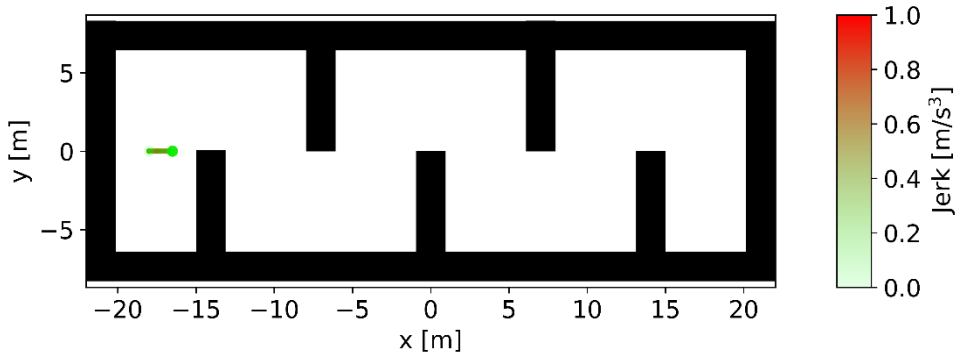


図 4.38　自動走行の失敗例($\gamma = 0.10, \delta = 0.20$)

最大躍度を制限でき，ロボットが正常に動作する範囲で，γ，δの最適値をグリッドサーチした結果を表 4.9と表 4.9示す．表 4.9の値は，各条件で10回実験を行い，その平均走行時間[s]を示している．表 4.9の値は，その平均走行距離[m]を示している．本実験環境においては両重みがともに0.75の場合に，走行時間と走行距離が最小となることが確認できる．前述の結果から，重みの最適化問題は，二次元空間の探索問題とみなすことができる．したがって，二分法によって，最適な重みは速やかに求められると考えられる．ただし，ロボットの制約や動作環境により最適な重みが異なることが考えられるため，各環境で重みの最適化が必要である．

表 4.9　躍度制御モデルにおける異なる重みの平均走行時間

δ \ γ	0.025	0.050	0.075	0.100	0.125	0.150
0.025	34.4	34.2	34.3	34.5	34.8	52.7
0.050	35.0	34.5	34.2	34.5	44.5	60.1
0.075	39.0	34.3	34.0	34.5	34.9	58.6
0.100	34.0	35.4	34.4	34.6	52.9	63.7
0.125	39.2	34.1	34.4	43.4	34.9	62.4
0.150	36.8	34.8	37.3	34.8	44.4	67.5

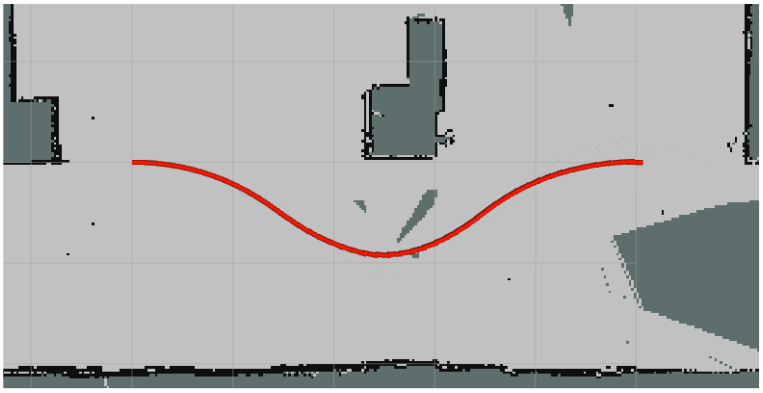
表 4.10　躍度制御モデルにおける異なる重みの平均走行距離

δ \ γ	0.025	0.050	0.075	0.100	0.125	0.150
0.025	48.5	48.4	48.8	49.4	48.1	48.8
0.050	48.8	48.7	48.9	49.3	49.8	50.2
0.075	49.9	48.8	48.0	49.2	48.6	49.9
0.100	48.2	48.7	48.5	49.2	49.5	53.8
0.125	50.1	48.4	49.1	51.6	48.0	55.5
0.150	49.3	48.4	48.8	49.0	48.8	58.5

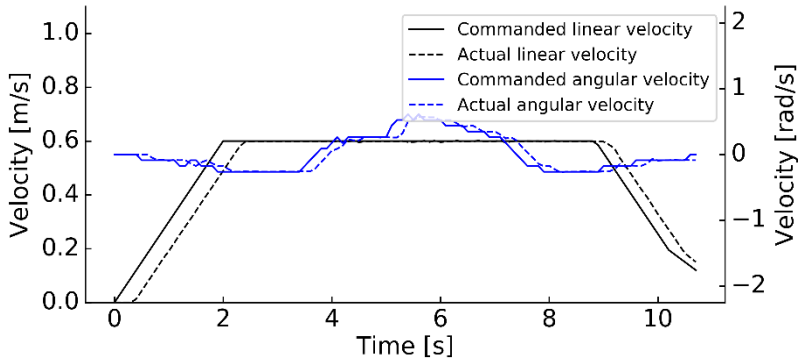
4.3.5　実験 4：実機実験

最後に躍度制御モデルの効果を実機で検証する．実機実験では，シミュレーション実験と違い，ロボットの実際の速度とDWAが出した速度の命令値に誤差が出る可能性がある．そのため，本実験ではロボットのオドメトリから読み取った速度（以下，応答値とよぶ）を実際の速度と仮定し，制御と評価を行う．

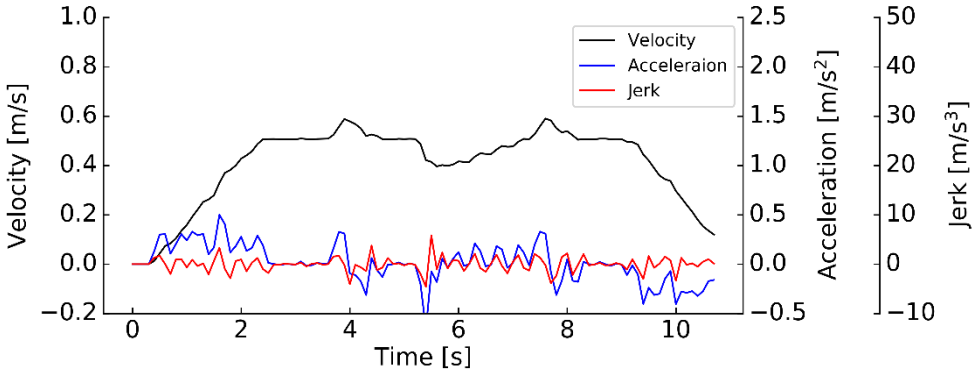
図 4.24の環境で各モデルの実験を各10回行った．結果の一例を図 4.39～図 4.41に示す．図 4.39(b)，図 4.40(b)，図 4.41(b)に示すように，速度の命令値と応答値の間に約0.3[s]の遅延が存在していたが，いずれの条件においても，ロボットは柱を回避して目標まで辿り着くことができた．図 4.39(a)，図 4.40(a)，図 4.41(a)を比較すると，躍度制御モデルでは，角躍度の制限により障害物の回避動作が鈍くなっていることが確認できる．図 4.39(c)，図 4.40(c)，図 4.41(c)は，最大躍度が発生した箇所における，左側で発生した速度，加速度，躍度を表し，図 4.39(d)，図 4.40(d)，図 4.41(d)は，左側で発生した速度，加速度，躍度を表す．速度制御モデルと加速度では，加減速の過程で大きい躍度が発生している．この例で発生した最大躍度は約6.0[$\mathrm{m/s^3}$]であった．躍度制御モデルでは，非線形な速度変化が実現しているが，各サンプリング時間での加速度変化はまだ十分に滑らかではなく，設定値以上の躍度が発生した．その最大値は約3.0[$\mathrm{m/s^3}$]であった．



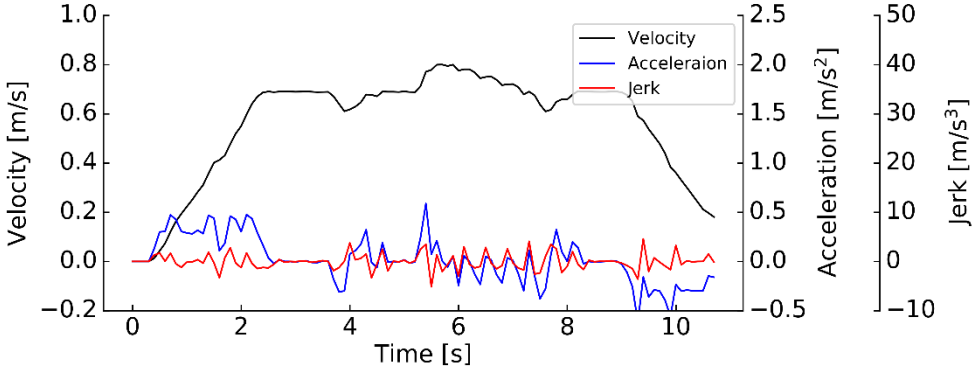
(a) 移動経路



(b) 速度の命令値と応答値の比較

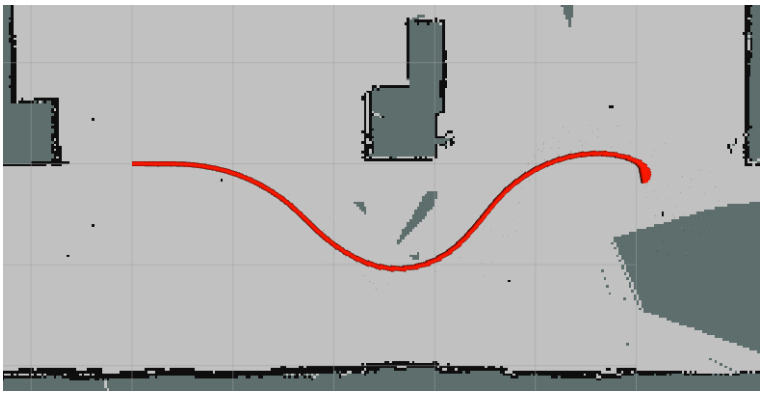


(c) ロボットの左側で発生した速度，加速度と躍度

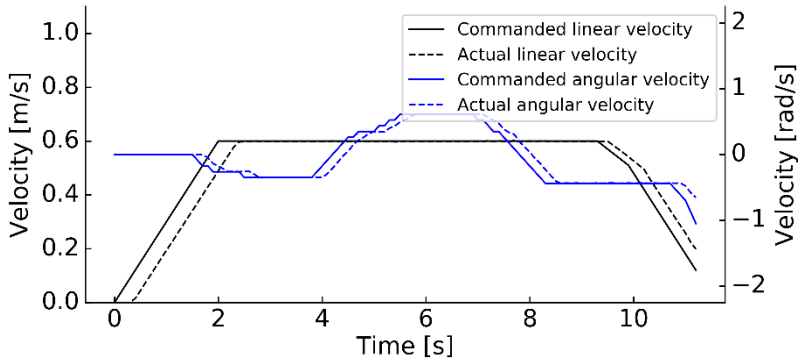


(d) ロボットの右側で発生した速度，加速度と躍度

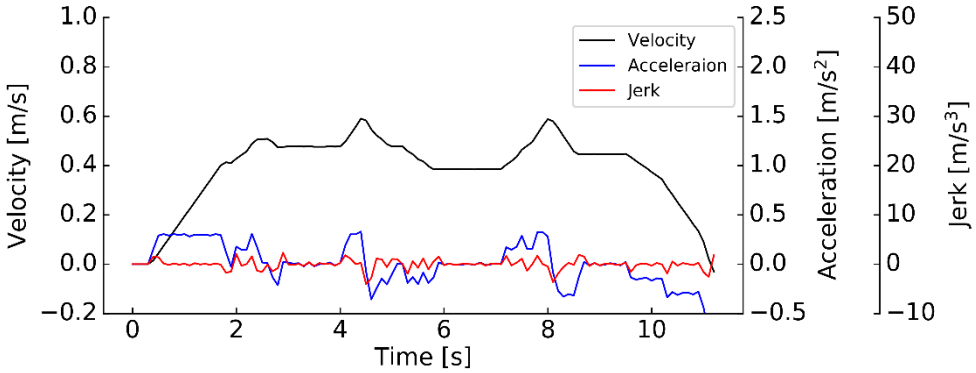
図 4.39　実環境における速度制御モデルの結果



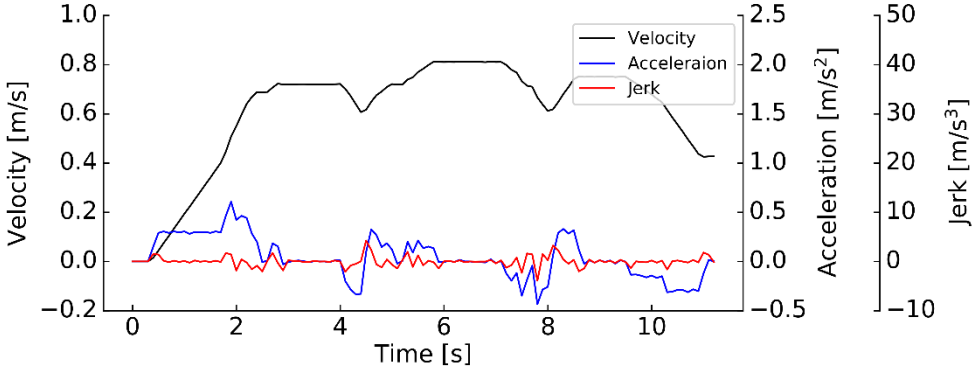
(a) 移動経路



(b) 速度の命令値と応答値の比較

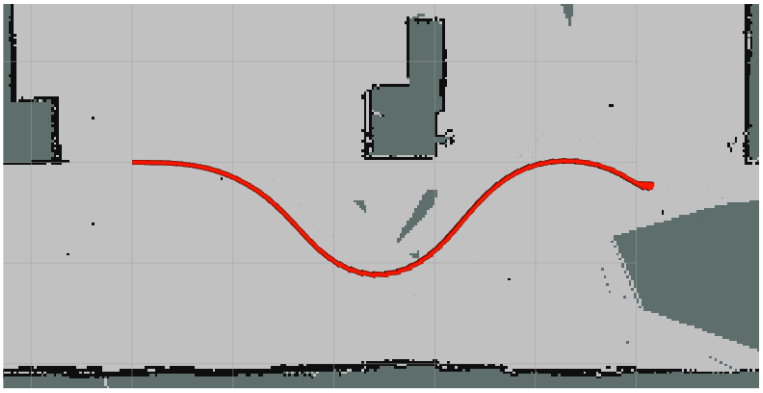


(c) ロボットの左側で発生した速度，加速度と躍度

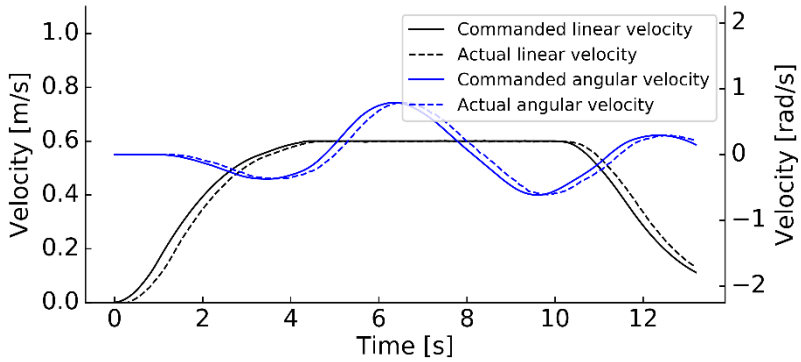


(d) ロボットの右側で発生した速度，加速度と躍度

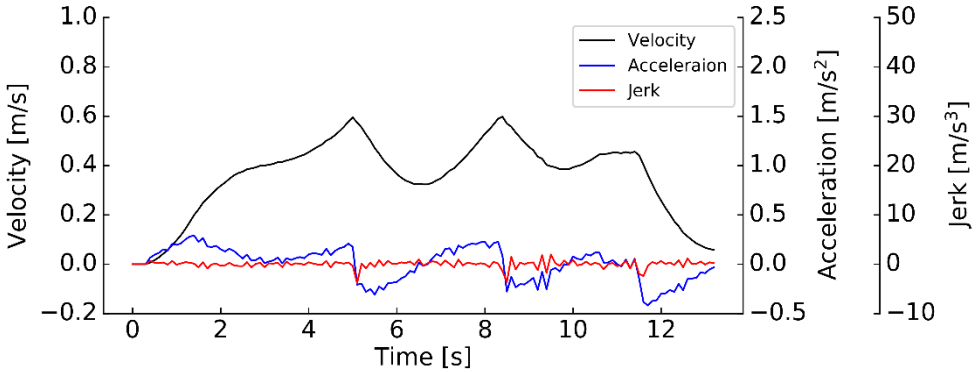
図 4.40　実環境における加速度制御モデルの結果



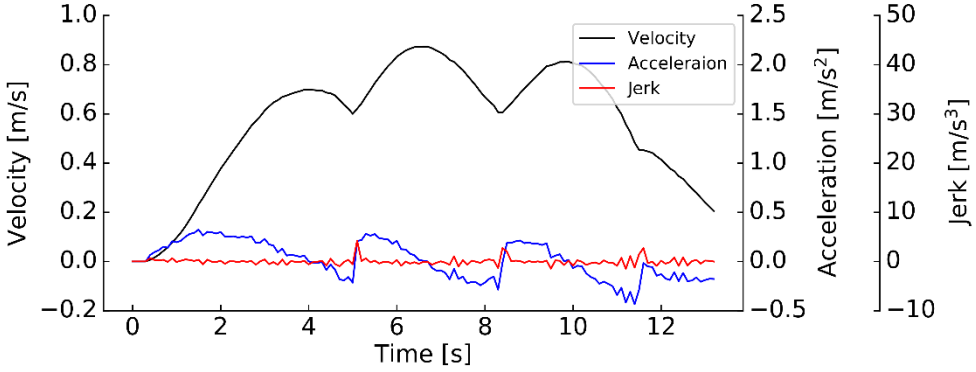
(a) 移動経路



(b) 速度の命令値と応答値の比較



(c) ロボットの左側で発生した速度，加速度と躍度

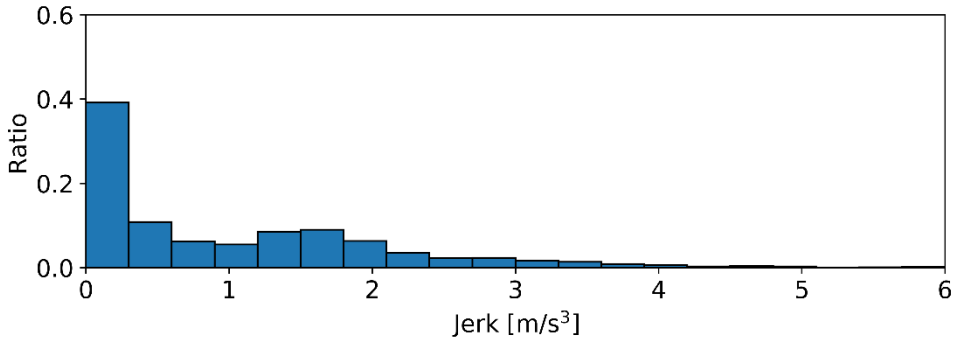


(d) ロボットの右側で発生した速度，加速度と躍度

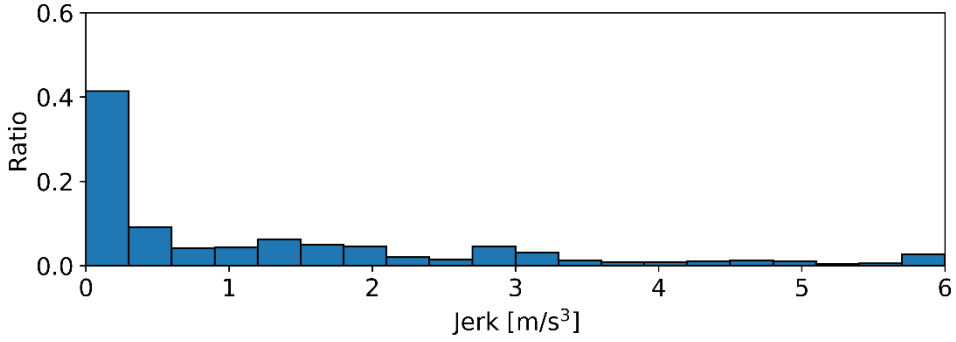
図 4.41　実環境における躍度制御モデルの結果

10回実験で発生した躍度のヒストグラムを図 4.42に示す. 横軸は躍度, 縦軸は発生する割合の分布を表す. また, ビンの幅は設定した最大躍度0.3[m/s³]と一致する. すなわち, 左から1つ目のビンに入れば最大躍度以下になっている. 速度制御モデルでは39%の躍度が，加速度制御モデルでは41%の躍度が，躍度制御モデルでは48%の躍度が設定した最大値以内に収まることが確認できる. 発生した躍度の最大値は速度制御モデルで8.4[m/s³]，加速度制御モデルで8.1[m/s³], 躍度制御モデルで6.4[m/s³]であった. また平均値は, 速度制御モデルで1.0[m/s³]，速度制御モデルで1.5[m/s³]，躍度制御モデルで0.4[m/s³]であった．等分散を仮定しないt検定により平均値の差を検定したところ，有意水準0.01で有意差が見られた．この結果から躍度制御モデルは発生する躍度を速度制御モデルと加速度制御モデルよりも減少させる効果があるといえる.

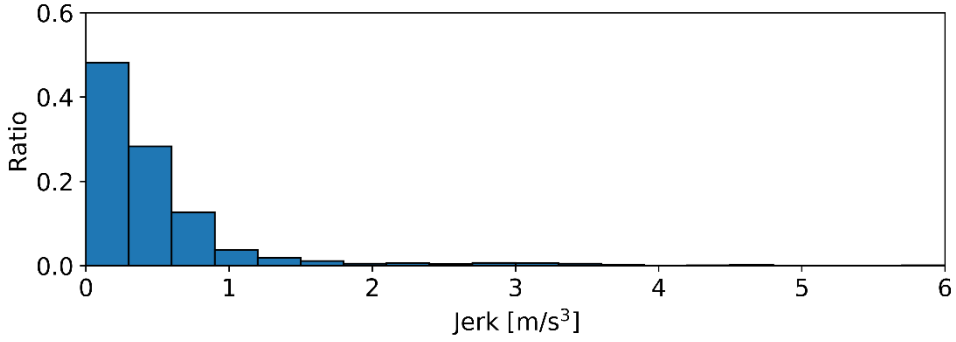
今回の実験では, 躍度を設定した最大値内に完全に制御することはできなかった. これは今回使用したロボットの速度制御の精度に起因していると考えられる．一方で，躍度制御モデルを用いることで，平均躍度の低減が可能となることや，これまでのDWAでは実現できなかった非線形な速度制御が実現できることが本実験から確認できた.



(a) 速度制御モデル



(b) 加速度制御モデル



(c) 躍度制御モデル

図 4.42　躍度の平均ヒストグラム

4.4　考察

従来のDWAでは，予測期間において，最初の期間のみで等加速させ，その後速度が不変と仮定するため，予測経路の到着点の範囲が狭いという問題と，躍度を滑らかに制御できないという問題があった. それらの問題を同時に解決する手法として，本研究では躍度制御モデルに基づいたDWAの改良手法を提案した. 提案手法は, 躍度制御モデルにより, 最大躍度を制限することができる. これにより，滑らかな加速度の変化を実現する.

まず，実験1の結果は，速度と角速度が速度制御モデル，加速度制御モデル，躍度制御モデルに従って変化する．加速度制御モデルと躍度制御モデルは，速度制御モデルよりも広い到着点の範囲が得られる．予測期間における速度の変化を考慮することで，予測経路の到着点の範囲が広がることが確認できた．また，速度変化にハードウェアの制限のみを考えると，ロボットの到着範囲はさらに広くなる．その到着範囲を得るために，MPCの使用を検討すべきと考えられる．

実験2の速度制御モデルの結果を見ると，速度が鋸歯のように変化することが確認できる．それは障害物を回避するため，速度が頻繁に変化する必要がある．加速度制御モデルと躍度制御モデルでは，予測期間における，速度変化が可能な経路を生成できるため，速度制御モデルのように速度をすぐに調整する必要がない．そのため，加速度制御モデルでの速度変化は速度制御モデルに比べ滑らかになる．図 4.28，図 4.31，図 4.34の結果を確認すると，躍度を抑えながら回避経路を計画する目標は達成できたと考えられる．

しかし，図 4.31(a)と図 4.33(a)に示す移動経路を見ると，加速度制御モデルでの回転は最も鈍い．本研究では，生成された予測経路の数を揃えるため，3つのモデルともに，サンプリング空間を横縦5等分に設定したが，加速度制御モデルのサンプリング空間の解像度を増加することで，回転の鈍さが改善できると考えられる．

実験3では，グリッドサーチにより，最適な重みが得られた．評価関数における躍度の重みの影響を調査する実験を行う前に，走行距離と時間が躍度の重みに対する関数が凸関数と想定したが，実験結果に明らかの凸性は確認できない．そのため，重みの自動調整は困難だと考えられる．ロボットが運用される環境とロボットパラメータに応じた重みの手動設定には時間がかかる．提案手法では，並進躍度と角躍度を別々に評価した．もし合成躍度を評価対象とすることで，同様な躍度制御効果が得られれば，調整すべき重みを1つ減少できる．

最後に，図 4.41(b)の結果を見ると，速度の応答値は非線形に変化している．速度の命令値と応答値に遅延がある．そのため，ロボットの速度は命令値通りに正確かつ迅速に変化することができず，実際に発生した躍度は規定値より大きくなる原因と考えられる．図 4.42の結果を観察すると，完全に規定値以下に抑えた躍度は9%向上し，遅延があっても提案手法で発生した躍度は全体的に小さくなる傾向が確認できる．本実験で使用されたPioneer-3ATはおよそ10年前の機種であり速度制御能力が十分ではないことがその原因であると考えられ

る．提案手法を速度制御能力の高いロボットに適用することで，より優れた性能を発揮することが期待できる．

第 5 章 経路の円弧近似による DWA の高速化

5.1 解決したい課題

DWAはロボットの現在速度に基づいて，ロボットの性能制約を考慮しながら，DWAは実現可能な速度指令をサンプリングし，それに基づいて予測経路を生成する．これらの予測経路を評価し，最適な予測経路に対する速度指令出力することを繰り返すことで，ロボットを移動させる．

DWAは経路計画と動作計画を統合する．前述のように，DWAはMPCの簡略として, 予測期間における速度系列を運動学モデルに繰り返し代入することで, 離散的な経路点が得られる．また，衝突検知のために，検知されたすべての障害物点から経路点までの距離を計算する必要がある．DWAの計算量は，ほぼこの2ステップから生じている．多くのDWAに関する研究では，経路生成と評価手法はそのまま使用されている. ロボットの応答性と搭載したセンサーの解像度に対する要求が高まっているため, 処理速度の問題を解決する必要がある.

そこで, 本章では, 経路の円弧近似によりDWAの高速化手法を提案する．提案手法では，DWAに速度制御モデルを用いて予測経路をする場合，円弧を用いて予測経路の離散的な経路点をモデリングし，衝突検知を行う．さらに，DWAに加速度制御モデルを用いて予測経路をする場合，複数の円弧を用いて予測経路を近似し，衝突検知を行うように拡張する．

5.2 提案手法

5.2.1 速度制御モデルに対する手法

従来のDWA では，予測期間における，速度制御モデルを用い速度系列を算出する．ロボットの速度は最初のサンプリング時間の後に不変であるため，予測期間にロボットの予測経路は円弧となる. 円弧の半径と並進速度と角速度の関係は式(3.10)に示す．従って，円弧経路の中心$(x_{\mathrm{p}}, y_{\mathrm{p}})$を原点とする極座標系で経路が表現できる．続いて，ロボット座標系における，(13), (14) を用い，障害物の座標$(x_{\mathrm{o}}, y_{\mathrm{o}})$を，極座標系 $(r_{\mathrm{o}}, \theta_{\mathrm{o}})$ に変換する．変換後の角度の計算には関数atan2を使用する．

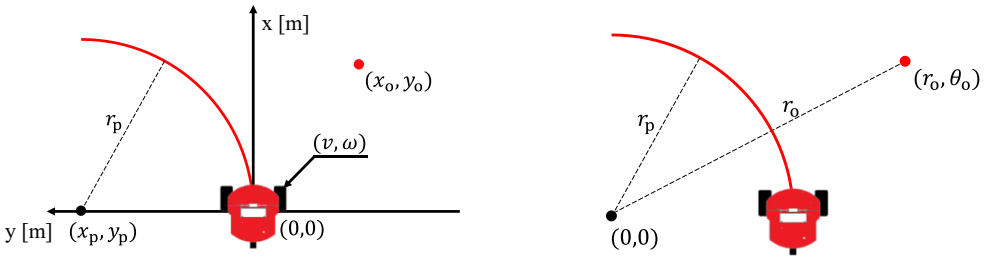
$$r_{\mathrm{o}} = \sqrt{\left(x_{\mathrm{o}} - x_{\mathrm{p}}\right)^2 + \left(y_{\mathrm{o}} - y_{\mathrm{p}}\right)^2} \tag{5.35}$$

$$\theta_{\mathrm{o}} = \mathrm{atan2}\left(y_{\mathrm{o}} - y_{\mathrm{p}}, x_{\mathrm{o}} - x_{\mathrm{p}}\right) \tag{5.36}$$

最後に，障害物の半径r_oと予測経路の半径r_pを用い，式(5.37)で障害物から予測経路までの距離d_{o}を計算する．

$$d_{\mathrm{o}} = \left|r_{\mathrm{o}} - r_{\mathrm{p}}\right| \tag{5.37}$$

図 5.43には，この過程を示している．$(x_{\mathrm{o}}, y_{\mathrm{o}})$は障害物の座標，$\left(x_{\mathrm{p}}, y_{\mathrm{p}}\right)$はロボット座標系における円弧経路の中心を表す．図 5.43(b)に示すように，変換後の障害物の座標は$(r_{\mathrm{o}}, \theta_{\mathrm{o}})$となる．次に，すべての障害物から予測経路までの距離d_{o}を計算する．距離の最小値dが設定した安全距離より小さい場合，その予測経路を危険とみなし破棄する．



(a) ロボット座標系　　(b) 極座標系

図 5.43　速度制御モデルに適用する場合

提案手法のフローチャートを図 5.44に示す．従来のDWAとは対照的に，提案手法は離散的な経路点を生成しないため，式(3.14)や(3.15)を繰り返し計算は省略できる．ステップ(B)では，円弧経路の中心の座標を計算する．計算時間は$t_{\mathrm{P1}}(n_{\mathrm{c}})$で表す．ここで，円弧経路の中心の数は$n_{\mathrm{c}}$で表す．速度制御モデルを用い，経路を予測する場合，n_{c}は1の定数と見なせる．ステップ(C)では，提案手法はすべての障害点に対して極座標変換を行うため，ステップ(C)の計算時間は円弧経路の中心の数n_{c}と検出された障害物の点の数n_{o}に依存する．従って，ステップ(C)の計算時間は$t_{\mathrm{P2}}(n_{\mathrm{c}}, n_{\mathrm{o}})$で表すことができる．提案手法の総計算時間は式(5.38)で表す．n_{c}はn_{p}よりかなり小さいため，提案手法の計算時間は従来手法の計算時間より短いと推定される．

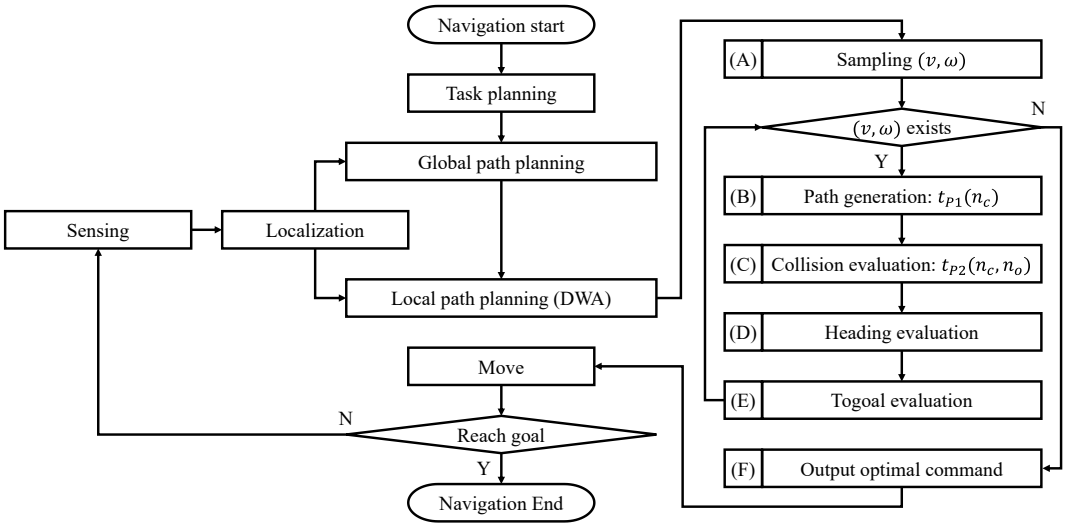


図 5.44　提案手法の流れ

$$t_{\mathrm{P}}(n_{\mathrm{c}}, n_{\mathrm{o}}) = t_{\mathrm{P1}}(n_{\mathrm{c}}) + t_{\mathrm{P2}}(n_{\mathrm{c}}, n_{\mathrm{o}}) \tag{5.38}$$

5.2.2　変速モデルに対する手法

加速度制御モデルや躍度制御モデルのよう，可変速度経路を生成すると，図5.45に示すように，円弧経路は曲率が単調に増加・減少する曲線経路になる．円弧経路は赤線で表す．加速と減速時の経路はそれぞれ黒線と灰色線で表す．ロボットの初期の並進速度と角速度をv_0，ω_0，経路の終端における並進速度と角速度をv_n，ω_nで表と，経路の曲率は，時刻t_0ではv_0/ω_0であり，時刻t_nではv_n/ω_nとなる．

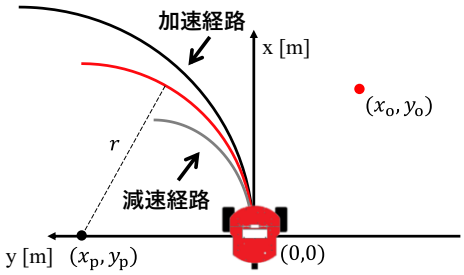


図 5.45　変速経路の様子

図 5.46に示すように，障害物から加速時の経路に引かれた垂線は，障害物から極座標系の原点に引かれた直線と平行にならないため，式(5.37)を用いて障害物から経路までの距離を算出することができない．そこで，本研究では，複数の円弧を用いて，障害物から曲線経路までの距離を近似する方法を提案する．この手法は，曲率が単調に増加または減少する経路に適用する．

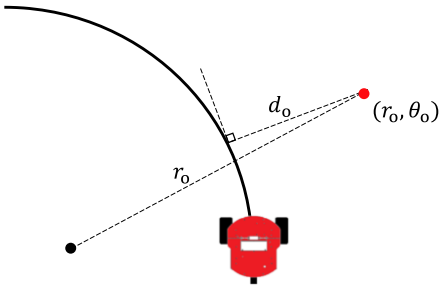
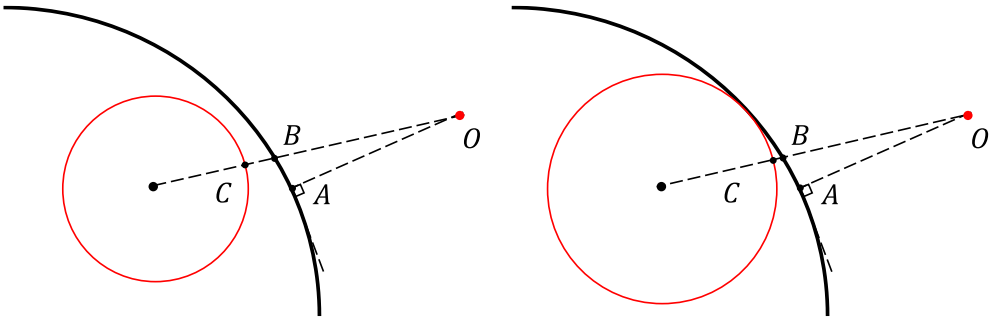


図 5.46　そのまま提案手法を変速経路に適用する場合

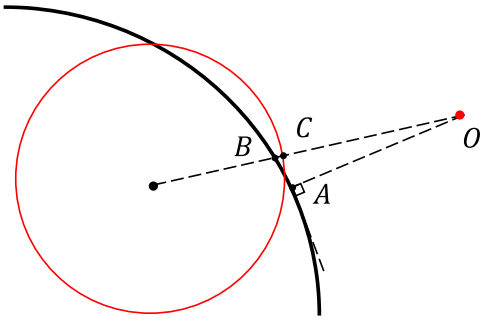
図 5.47は, 加速度制御モデルによって生成された経路の例である. 赤い曲線が予測経路を表す. ここでは，予測経路の曲率中心のある側を内側，反対側を外側と呼ぶ. 黒い円が予測経路の内側に生成された円弧を表す. 最初に極座標系を設定し，経路の内側にある任意の点を原点とし，この原点を中心とした円を描く. この円は，図 5.47(a)のように経路から離れる，接する，または図5.47(b)のように交差することが可能である.

障害物から円の中心に向かう線分は，点Cで円と交差し，点Bで経路と交差する. 障害物から経路までの垂線の足は点Aで表す. 線分OAの長さは障害物から経路までの最短距離のため，OAは線分OBの長さより短い. 図 5.47 (a)に示すように，円が経路から離れる，または経路と接する場合，OBは線分OCの長さより短い. 従って，OAはOCより短いことが分かる. しかし，図 5.47 (b)に示すように，円が経路と交差する場合は，障害物の位置により，OAとOCの大小関係が変化する. 従って提案手法で距離推定のために生成する円弧は，経路と交差しないことを保証する必要がある.

さらに，円の半径が大きいほどOCは小さくなり，OAに近づく. 距離推定精度を向上させるためには，経路と交差しない最大の円を採用する必要がある. 経路の曲率範囲は$[v_0/\omega_0, v_n/\omega_n]$であるため，ロボットが加速する場合，経路と交差しない最大の円の半径はv_0/ω_0，減速の場合，半径はv_n/ω_nである.



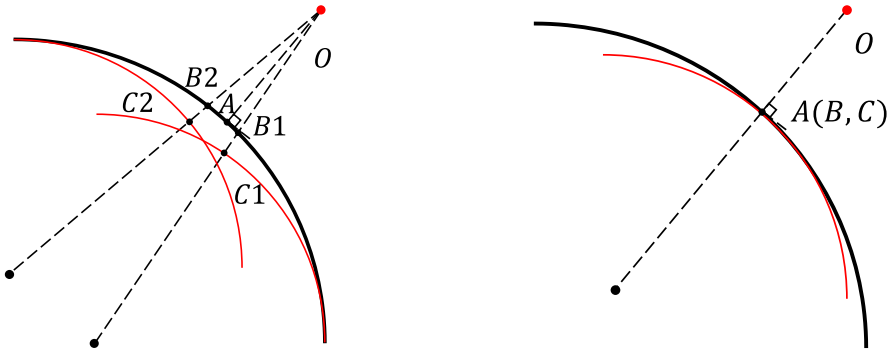
(a) 距離推定が可能な位置関係



(b) 距離推定が不可能な位置関係

図 5.47　経路の内側に円弧を生成する例

図 5.48(a)に示すように，経路に接する円弧を経路の内側に複数生成した場合，　OAは min(OC1, OC2)より短くなる．これは，生成される円が多いほど，min(OC1, OC2, ...)がOAに近づく可能性が高くなることを意味する．また，図 5.48 (b)に示すように，点Aが円弧と経路の両方の接点の場合，点A，点Bと点Cが一点になる．OAとOCは等しくなり，　OAの真値を算出できる．

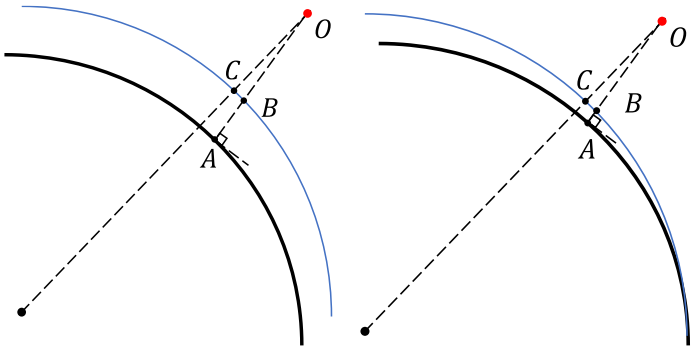


(a) 複数の円弧を生成する場合　　(b) 点 A が円弧と経路の接点の場合

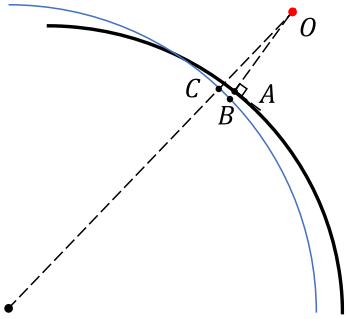
図 5.48　内側に円弧を生成する場合の誤差分析

この時点で, 障害物から経路までの距離の片側の境界が決定される. 同様に, 図 5.49に示すように, 経路の内側にある任意の点を原点とし, この原点を中心とした円弧を経路の外側に描く. この外側の円弧は, 経路から離れる, 接する, または交差することが可能である. 障害物から円の中心に向かう線分は円と点

Cで交差する．障害物から経路までの垂線の足は点Aで表す．線分OAは円と点Bで交差する．この場合，OBは線分OCより長い．図 5.49(a)に示すように，円が経路から離れる，または経路と接する場合，線分OAとOBは平行，線分OAの長さは障害物から経路までの最短距離のため，OAは線分OBの長さより長い．従って，OAは線分OCより長いことが分かる．一方で，図 5.49(b)に示すように，円が経路と交差する場合には，障害物の位置により，OAとOBの大小関係が変化する．従って，提案手法で距離推定のために生成される外側の円弧は，経路と交差しないことを保証する必要がある．さらに，円の半径が小さいほどOCは大きくなり，OAに近づく．距離推定精度を向上させるためには，経路と交差しない最小の円を採用する必要がある．経路の曲率範囲は$[v_0/\omega_0, v_n/\omega_n]$であるため，ロボットが加速する場合，経路と交差しない最大の円の半径はv_n/ω_n，減速の場合，半径はv_0/ω_0である．



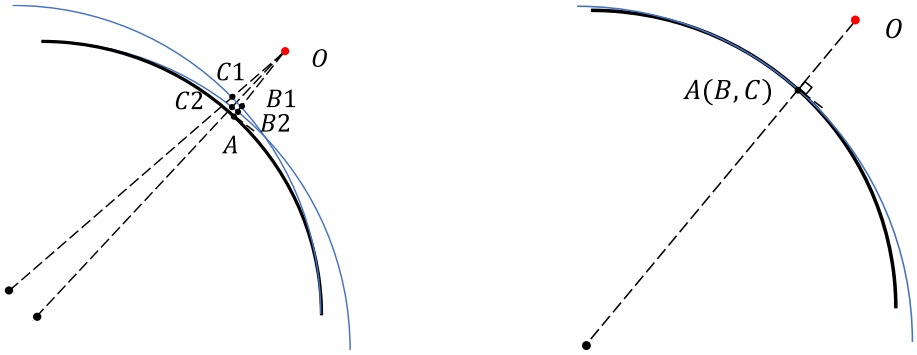
(a) 距離推定が可能な位置関係



(b) 距離推定が不可能な位置関係

図 5.49　経路の外側に円弧を生成する例

図 5.50(a)に示すように，経路に接する円弧を経路の外側に複数生成した場合，OAはmax(OC1, OC2)より長くなる．これは，生成される円が多いほど，max(OC1, OC2, ...)がOAに近づく可能性が高くなることを意味する．また，図 5.50 (b)に示すように，点Aが円と経路の両方の接点の場合，点A，点B，点Cが一点になる．OAとOCは等しくなり，OAの真値を算出できる．



(a) 複数の円弧を生成する場合　　(b) 点 A が円弧と経路の接点の場合

図 5.50　外側に円弧を生成する場合の誤差分析

経路の内側と外側に円弧を複数生成することで, 障害物から経路までの距離の範囲を求めることができる．複数の円を利用するため，式(5.38)のn_cは1ではない．それでも，n_cはn_pよりかなり小さいため，提案手法は従来手法と比べ，計算時間の短縮が期待できる．

5.2.3　計算方法

提案手法では，円弧の中心の位置と半径を計算する必要がある．予測経路と接する円弧を使用するため，接点座標の計算に予測経路の式x_{path}とy_{path}が必要である．x_{path}とy_{path}は速度モデルを式(3.14)や(3.15)に代入することで得られる．

本研究では，積分結果に初等関数ではない関数が出ないように，式(5.39)と(5.40)に示す速度が加速度制御モデル，角速度が速度制御モデルに従わせる．また，角速度が短い間t_0～t_1にのみ変化できるため，角速度が変化する過程は省略できると考えられる．変化後の期間$[\Delta t, t_p]$のx_{path}とy_{path}のみを計算する．期間$[\Delta t, t_p]$における，v_1とω_1は最初の速度と角速度を表す．

$$v(t) = \begin{cases} v_0 + a^V \cdot t & \Delta t \le t \le t_{ea} \text{ or } \Delta t \le t \le t_{ed} \\ v_{\max} & t_{ea} < t \le t_p \text{ and } a^V > 0 \\ 0 & t_{ed} < t \le t_p \text{ and } a^V \le 0 \end{cases} \tag{5.39}$$

$$\omega(t) = \begin{cases} \omega_0 + a^\Omega \cdot t & t < \Delta t \\ \omega_1 & \Delta t \le t < t_p \end{cases} \tag{5.40}$$

v_0はロボットの初期並進速度，$v_{\max}$は最大並進速度, a^{V}とa^{Ω}は選択された並進加速度($a^{\mathrm{V}} \in [-a^{\mathrm{V}}_{\max}, a^{\mathrm{V}}_{\max}]$)と角加速度($a^{\Omega} \in [-a^{\Omega}_{\max}, a^{\Omega}_{\max}]$)である．　t_pは予

測期間である．簡単のため，$(v_{\max}-v_0)/a^{\mathrm{V}}$を$t_{\mathrm{ea}}$，$v_0/a^{\mathrm{V}}$を$t_{\mathrm{ed}}$とする．ここで，ロボットが加速する場合$(a^{\mathrm{V}}>0)$の計算結果の例を示す．式(3.14)に式(5.39)と(5.40)を代入し，式(5.41)，(5.42)と(5.43)が得られる．予測期間における，$x_{\mathrm{path}}(t)$と$y_{\mathrm{path}}(t)$は任意時刻tでロボットの座標，$\theta_{\mathrm{path}}(t)$はロボットの向きを表す．

$$x_{\mathrm{path}}(t)=\begin{cases}\dfrac{\omega_1\cdot(v_0+a\cdot t)\sin(\omega_1\cdot t)+a\cdot\cos(\omega_1\cdot t)}{{\omega_0}^2}+c_1\\ \dfrac{v_{\max}}{\omega_1}\sin(\omega_1\cdot t)+c_2\end{cases} \tag{5.41}$$

$$y_{\mathrm{path}}(t)=\begin{cases}\dfrac{-\omega_1\cdot(v_0+a\cdot t)\cos(\omega_1\cdot t)+a\cdot\sin(\omega_1\cdot t)}{{\omega_0}^2}+c_3\\ -\dfrac{v_{\max}}{\omega_1}\cos(\omega_1\cdot t)+c_4\end{cases} \tag{5.42}$$

$$\theta_{\mathrm{path}}(t)=\omega_1\cdot t \tag{5.43}$$

次に，式(5.41)と(5.42)の積分定数c_1，c_2，c_3，c_4を計算する．式(5.41)の第1区間を$A(t)+c_1$，第2区間を$B(t)+c_2$，(5.42)の第1区間を$C(t)+c_3$，第2区間を$D(t)+c_4$として簡略化する．まず，ロボット座標系における，$t=0$の場合，ロボットは原点にある．そのため，式(5.44)，(5.45)を用い，積分定数c_1とc_3計算できる．その後，予測経路は連続すべきため，式(5.41)の第1区間と第2区間は連続すべきである．そのため，式(5.46)と(5.47)を用い，積分定数c_2とc_4を計算できる．

$$\lim_{t\to 0^+}A(t)+c_1=0 \tag{5.44}$$

$$\lim_{t\to 0^+}C(t)+c_3=0 \tag{5.45}$$

$$\lim_{t\to t_{\mathrm{ea}}^-}A(t)+c_1=\lim_{t\to t_{\mathrm{ea}}^+}B(t)+c_2 \tag{5.46}$$

$$\lim_{t\to t_{\mathrm{ed}}^-}C(t)+c_3=\lim_{t\to t_{\mathrm{ed}}^+}D(t)+c_4 \tag{5.47}$$

円弧の半径が決定された後に，中心の位置を計算する必要がある．まず，経路の起点に接する一対の円弧を生成する．式(5.48)と(5.49)は内側の円弧と外側の円弧のパラメトリック方程式それぞれを表す．円弧の中心はそれぞれ$(0,v_1/\omega_1)$と$(0,v_n/\omega_1)$である．

$$\begin{cases} x_{\text{in}}(t) = \dfrac{v_1}{\omega_1}\sin(\omega_1 \cdot t) \\ y_{\text{in}}(t) = \dfrac{v_1}{\omega_1}(1 - \cos(\omega_1 \cdot t)) \end{cases} \tag{5.48}$$

$$\begin{cases} x_{\text{out}}(t) = \dfrac{v_n}{\omega_1}\sin(\omega_1 \cdot t) \\ y_{\text{out}}(t) = \dfrac{v_n}{\omega_1}(1 - \cos(\omega_1 \cdot t)) \end{cases} \tag{5.49}$$

ほかの円弧はこの一対の円弧の平行移動と考えることができ，移動量は式(5.50)～(5.53)を用いて計算する．ほかの円弧の中心はそれぞれ$\left(0 - \Delta x_{\text{in}}(t), (v_1/\omega_1) - \Delta y_{\text{in}}(t)\right)$と$\left(0 - \Delta x_{\text{out}}(t), (v_n/\omega_1) - \Delta y_{\text{out}}(t)\right)$である．

$$\Delta x_{\text{in}}(t) = x_{\text{in}}(t) - x_{\text{path}}(t) \tag{5.50}$$

$$\Delta y_{\text{in}}(t) = y_{\text{in}}(t) - y_{\text{path}}(t) \tag{5.51}$$

$$\Delta x_{\text{out}}(t) = x_{\text{out}}(t) - x_{\text{path}}(t) \tag{5.52}$$

$$\Delta y_{\text{out}}(t) = y_{\text{out}}(t) - y_{\text{path}}(t) \tag{5.53}$$

二対の円弧を生成する例を図 5.51に示す．この例では，経路の起点と終点を接点として円弧を生成した．式(5.54)～(5.57)は障害物Oから各円弧の中心までの距離を計算する．

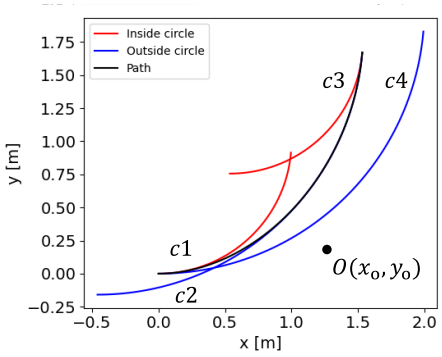


図 5.51　2 対の円弧を生成する例

$$r_{\text{in1}} = \left\|(x_{\text{o}}, y_{\text{o}}) - \left(\Delta x_{\text{in}}(0), \Delta y_{\text{in}}(0)\right)\right\|_2 \tag{5.54}$$

$$r_{\text{in2}} = \left\|(x_{\text{o}}, y_{\text{o}}) - \left(\Delta x_{\text{in}}(t_{\text{p}}), \Delta y_{\text{in}}(t_{\text{p}})\right)\right\|_2 \tag{5.55}$$

$$r_{\text{out1}} = \left\|(x_{\text{o}}, y_{\text{o}}) - \left(\Delta x_{\text{out}}(0), \Delta y_{\text{out}}(0)\right)\right\|_2 \tag{5.56}$$

$$r_{\text{out2}} = \left\|(x_{\text{o}}, y_{\text{o}}) - \left(\Delta x_{\text{out}}(t_{\text{p}}), \Delta y_{\text{out}}(t_{\text{p}})\right)\right\|_2 \tag{5.57}$$

障害物Oから経路までの距離dは式(5.58)で表す．

$$\max\left(r_{\text{out1}}-\frac{v_n}{v_1}, r_{\text{out2}}-\frac{v_n}{v_1}\right) \le d \le \min\left(r_{\text{in1}}-\frac{v_1}{v_1}, r_{\text{in2}}-\frac{v_1}{v_1}\right) \qquad (5.58)$$

提案手法では，平面上のすべての障害物点との距離を算出する必要がない．障害物点と経路の相対位置により，その一部の距離は簡単に算出できる．図5.52に示すように，経路の起点と終点の法線の交点を中心とする極座標系が作成された．2つの法線が平面を4つの領域に分割する．極座標系における，障害物が位置する方向から，障害物がどの領域にあるかを判別できる．灰色領域にあるO1やO4のような障害物にとって，経路までの距離は，障害物から経路の起点と終点までの距離の最小値となる．白い領域にあるO2やO3のような障害物の場合，提案手法を用いて，障害物から経路までの距離値が位置する区間を決定し，その区間の中央値を距離値とする．

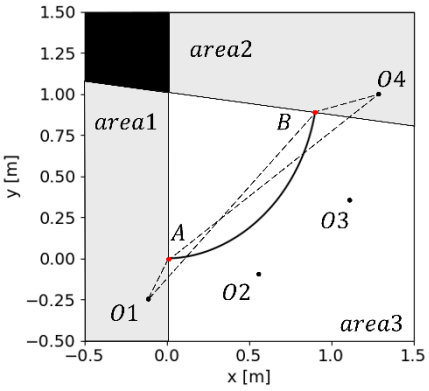


図 5.52　領域によって異なる距離計算手法

5.3　実験

制御プログラムはROSを用いて構築する．本研究では，Stageを用いたシミュレーション実験を行う．障害物点から経路までの距離の真値の計算には，多くの計算時間がかかるため，自律移動実験を行いながらでは，すべての距離の真値を算出できない．そのため，距離推定の精度と時間の計測は2つの実験に分けて実施した．実験1では，ロボットを静止させ，経路予測のみを行い，従来手法と提案手法の各手法の距離推定精度を比較する．実験2では，自律移動を行いながら，各手法の計算時間を計測する．最後に，2つの実験の結果を合わせて分析する．

5.3.1　実験１：距離推定精度の評価

図 5.53は実験1の環境を示している．ロボットの初期位置を$(0, 0)$とし，黒線で予測経路を表す．ロボットの前方に 100 個の障害物点をランダムに生成する．障害物点を生成する範囲はロボットを中心とした半径5[m]の範囲とした．これはレーザーレンジファインダーの検出距離と一致する．予測期間t_pは2.0[s]に設定した．初期並進速度は1.0[m/s]，角速度は1.0[rad/s]に設定した．予測期間における並進加速度は-1.0, -0.5, 0.0, 0.5, 1.0$[\mathrm{m/s^2}]$とした．

提案手法における円弧の数は1，2，3組とした．すなわち，n_cは2，4，6となる．円弧は予測経路の起点，中点，終点に接するように生成する． n_cが2の場合，経路の起点(s)，中点(m)，終点(e)のいずれかに接するものとし3種類の手法を比較する．n_cが4の場合，起点と中点(s+m)，起点と終点(s+e)，中点と終点(m+e)に接する3つの手法を比較する．n_c= 6の場合は，経路の起点，中点，終点(s+m+e)に接する1手法のみとした．

従来手法では，サンプリング時間Δtは0.10，0.05，0.02[s]に設定した．すなわち，n_pの値はそれぞれ20，40，100となる．また，運動学モデルは経路生成の結果を影響するため，本実験では，tangentモデルとsecantモデルの結果を調査した．

すべての障害物から経路までの距離の真値は解析幾何学を用いて計算し，ベースラインとした．提案手法と従来手法によって得られた距離とベースラインとの差を誤差とし，その平均値を距離推定精度の評価値とした．

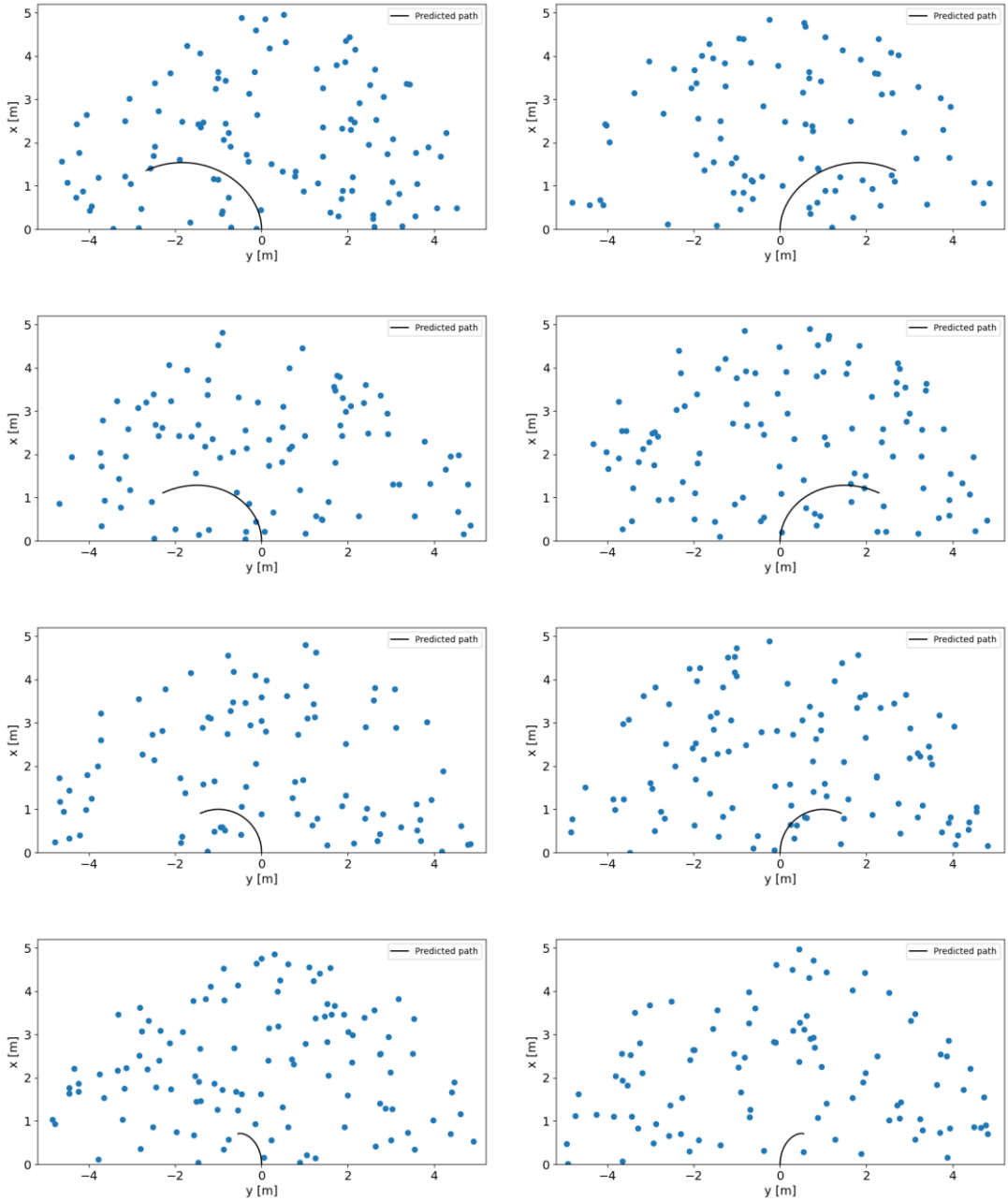


図 5.53　障害物を模擬するランタンな点を生成する様子

表 5.11に，従来手法での実験で得られた距離推定誤差の平均値をミリメートル単位で示す．水平方向を観察すると，両モデルともに，経路点の増加につれて，各加速度における距離推定誤差の平均値が減少していることが分かる．secantモデルを用いた距離推定誤差の平均値は，常にtangentモデルより小さい．縦方向を観察すると，両モデルともに，並進加速度が小さくなるにつれて，距離推定誤差の平均値も小さくなるという特性が示されている．

表 5.11　従来手法で距離推定誤差の平均値

並進加速度 $[\mathrm{m/s^2}]$	$n_\mathrm{p}=20$		$n_\mathrm{p}=40$		$n_\mathrm{p}=100$	
	tangent	secant	tangent	secant	tangent	secant
-1.0	55	51	42	40	34	33
-0.5	23	18	12	9	5	4
0.0	32	1	16	1	7	1
0.5	78	39	51	32	36	28
1.0	104	52	69	43	48	38
Mean	59	33	38	25	26	21
Std	29.5	19.9	21.5	17.1	17.1	15.6

表 5.12に，提案手法での実験で得られた距離推定誤差の平均値をミリメートル単位で示す．水平方向を観察すると，円弧の数が増えるにつれて，各加速度に対する距離推定誤差の平均値は減少する．具体的に，$n_\mathrm{c}=2$の場合，接点が経路の中点(m)にある円弧のほうが経路全体の傾向をよく表しており，誤差が小さくなると考えられる．同様に，$n_\mathrm{c}=4$の時，接点が経路の始点と中点(s+m)にある円弧のほうは，ほかの位置の円弧と比べ，経路の傾向をより良く表現できる．その結果，(s+m)の誤差が小さくなっている．$n_\mathrm{c}=6$の場合，並進加速度が-1.0～1.0$[\mathrm{m/s^2}]$の範囲で，(s+m+e)での距離推定誤差の平均値が最も小さい．これは，円弧の数が多いほど距離推定誤差が小さくなるという提案手法の狙いと一致する．縦方向を観察すると，提案手法では曲率が単調に増加または減少する曲線経路を複数の円弧経路に近似するため，速度の変動が少ない場合には，距離推定誤差が小さくなると想定できる．縦方向を観察すると，予測期間における，並進加速度が0.0$[\mathrm{m/s^2}]$，すなわちロボットの運動が等速円運動の場合，提案手法では障害物と経路の距離の真値を算出できる．

表 5.12　提案手法で距離推定誤差の平均値

並進加速度 $[\mathrm{m/s^2}]$	$n_\mathrm{c}=2$			$n_\mathrm{c}=4$			$n_\mathrm{c}=6$
	s	m	e	s+m	s+e	m+e	s+m+e
-1.0	34	28	181	29	30	29	31
-0.5	102	22	84	16	19	27	13
0.0	0	0	0	0	0	0	0
0.5	98	30	161	33	30	47	30
1.0	55	78	317	46	81	71	42
Mean	58	32	148	25	32	35	23
Std	38.9	25.5	105.6	15.7	27.0	23.6	14.9

図 5.54に示すように，提案手法で最も性能の良い結果，すなわちs，s+m，s+m+eの距離推定結果と従来手法での結果を比較した．図 5.54において，横軸は従来手法における経路点の数n_{p}，提案手法における円弧の数n_{c}を表す．縦軸は距離推定誤差の平均値を表す．エラーバーは加速度変化による標準偏差を表す．この比較から，提案手法は従来手法における優れた性能を発揮するsecantモデルと同程度の距離推定精度を達成することが分かる．

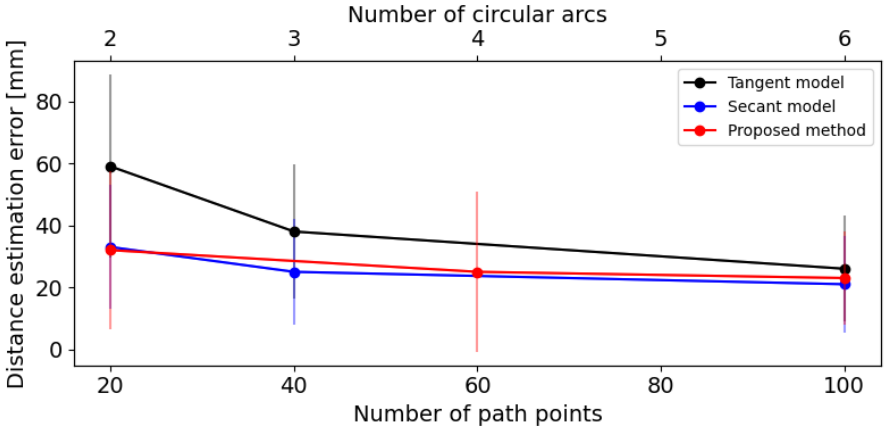


図 5.54　距離推定誤差の比較

5.3.2　実験 2：距離計算の性能評価

実験2では，自律移動中に要したDWAの計算時間により提案手法と従来手法の性能を比較する．表 5.13にPCとその他のライブラリのパラメータを示す．図 4.23(a)に示す環境を環境1，(b)に示す環境を環境2として使用した．ロボットのパラメータを表 5.14に示す．ロボットの初期位置は(-18, 0)，目標は(18, 0)に設定した．制御周波数は10Hzに設定した．この構成では，ループ内のすべての処理で消費される総時間が100[ms]以下の場合，システムは自動的に残り時間をパディングし，制御周波数を10Hzに保つ．レーザーレンジファインダーの角度分解能を1，2，3[num/°]に設定すると，環境1では，自律移動中に1フレームあたり検出される障害点の平均数n_{o}は，約130，260，390であり，環境2では，障害点の平均数n_{o}は，約210，420，630であった．n_{o}の値はレーザーレンジファインダーの角度分解能を調整することで制御しているため，n_{o}の結果は参考値である．予測期間を2[s]，従来手法でのサンプリング時間Δtを1.00，0.50，0.20，0.10，0.05，0.02[s]に設定した．式(3.1)よりn_{p}の値はそれぞれ2，4，10，20，40，100となる．提案手法での円弧の数を1，2，3対としたため，n_{c}の値は2，4，6となる．両手法ともに，選択可能な並進加速度と回転速度の範囲を5等分し，

同時に25本の予測経路を生成する．シミュレーション環境で10回の自律移動を行い，1ループあたりDWAの計算時間の平均値をミリ秒単位で集計する．$n_{\mathrm{p}} = 20$と$n_{\mathrm{c}} = 2$の場合，従来手法と提案手法による自律移動が全て成功したため，その結果をベースラインとして，n_{p}とn_{c}の増加に対して計算時間に有意な変化があるかどうかを調べた．「**」は有意水準0.01，「*」は有意水準0.05で有意であることを意味する．

表 5.13　実験環境の構成

システム構成	詳細
OS	Ubuntu 18.04.5 LTS
ROS	Melodic 1.14.9
Stage	4.3.0
CPU	Intel Core i7-8700 3.20Ghz×12
Numpy	1.16.4
Scipy	1.3.0

表 5.14　ロボットのパラメータ

パラメータ	値
最大並進速度 [m/s]	2.0
最大角速度 [rad/s]	6.28
最大並進加速度[m/s^2]	1.0
最大角加速度[rad/s^2]	5.24
計測範囲 [deg]	360
計測距離 [m]	5
角度分解能 [num/°]	1,2,3

移動距離と移動時間の比較を表 5.15と表 5.16に示す．ロボットの初期位置から目標までの直線距離は36[m]である．表 5.15の結果から，従来手法で$n_p < 10$の場合，移動距離と移動時間には明らかに差があることがわかる．これは，経路点少なすぎるため，自律移動中にロボットが道に迷うことが原因と考えられる．環境2は環境1よりも狭いため，表 5.16の結果を観察すると，n_{p}が20未満の場合，自律移動の成功は困難になる．従って，複雑な環境ほど，ロボットはより多くの経路点を必要とすると言える．

自律移動の成功を保証するのに十分な経路点が生成された場合，従来手法と提案手法の間に，移動距離と移動時間には明らかな差がないことが観察される．この特性は，実験が同一の制御周波数で行われたために生じる．実際のロボット操作では，毎制御ループにおける，ローカル経路計画だけでなく，環境の感知やリアルタイム画像処理のような特定のタスク処理も含む．従って，ローカ

ル経路計画に消費する時間を削減することで, 他の重要な処理のための時間を増やすことができる.

表 5.15　環境 1 における従来手法と提案手法の移動距離と時間

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_o \approx 130$		$n_o \approx 260$		$n_o \approx 390$	
		移動距離 [m]	移動時間 [s]	移動距離 [m]	移動時間 [s]	移動距離 [m]	移動時間 [s]
従来手法	$n_p = 2$	90.6	79.9	90.4	79.9	89.4	79.9
	$n_p = 4$	44.8	26.7	44.4	26.5	43.0	25.5
	$n_p = 10$	39.2	23.4	39.4	23.1	39.0	23.5
	$n_p = 20$	39.0	23.6	39.2	23.5	39.2	23.6
	$n_p = 40$	39.0	25.0	39.0	25.2	39.0	25.0
	$n_p = 100$	39.0	24.8	39.0	25.0	39.6	23.6
提案手法	$n_c = 2$	38.4	21.5	38.4	21.6	38.4	21.5
	$n_c = 4$	38.4	21.6	38.0	21.3	38.1	21.4
	$n_c = 6$	39.1	25.1	39.0	24.9	39.1	25.0

表 5.16　環境 2 における従来手法と提案手法の移動距離と時間

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_o \approx 130$		$n_o \approx 260$		$n_o \approx 390$	
		移動距離 [m]	移動時間 [s]	移動距離 [m]	移動時間 [s]	移動距離 [m]	移動時間 [s]
従来手法	$n_p = 2$	86.0	79.9	85.2	79.9	87.6	79.9
	$n_p = 4$	66.2	52.7	63.4	46.8	61.2	42.9
	$n_p = 10$	60.8	42.5	61.2	41.6	61.2	41.5
	$n_p = 20$	53.4	30.7	53.6	31.4	53.8	30.9
	$n_p = 40$	53.6	32.1	53.2	32.5	53.6	32.2
	$n_p = 100$	55.8	31.7	56.0	31.8	55.6	31.5
提案手法	$n_c = 2$	51.6	30.5	54.3	31.6	52.6	31.1
	$n_c = 4$	53.1	31.4	54.4	32.2	53.6	31.7
	$n_c = 6$	53.6	31.7	54.1	32.0	54.3	32.1

表 5.17と表 5.18に，1フレームあたりのDWAの計算時間の比較を示す．式(3.5)と(5.38)によれば，計算時間は検出された障害物点の数とともに上昇する．表 5.17と表 5.18の水平方向を観察すると，予想通り両手法とも検出された障害物点の数の増加に伴い，計算時間も増加することがわかる．縦方向を観察すると，従来手法の場合，経路点の数n_{p}の増加につれて，計算時間が有意的に増加する．提案の場合，円弧の数n_{c}の増加に対し，計算時間の増加が少ない．それには2つの原因がある．1つ目は従来手法では予測経路の生成に，繰り返し計算の時間が長いことである． 2つ目は，n_{c}はn_{p}よりかなり小さいためである．これらにより計算量が少なくなったと考えられる．また，従来手法では，n_{p}を減少させることにより，計算時間を短縮することは可能だが，前述したように，経路点が少なすぎると自律移動が失敗するリスクが高くなる．これと対照的に，提案手法での計算時間は円弧の数n_{c}が増加しても大きな変化は見られず，従来手法の計算時間よりも常に短いことが確認できる．提案手法では，式(5.41)と(5.42)を用い，有限個の円弧の接点の位置を確認するために経路の方程式に速度の式を代入するだけでよく，計算量は少ない．従って，提案手法では円弧の数を増やしても，計算時間が有意に増加することが少なくなる．

表 5.17　環境 1 における従来手法と提案手法の計算時間

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_{\mathrm{o}} \approx 130$		$n_{\mathrm{o}} \approx 260$		$n_{\mathrm{o}} \approx 390$	
		平均時間 [ms]	P 値	平均時間 [ms]	P 値	平均時間 [ms]	P 値
従来手法	$n_{\mathrm{p}} = 2$	16.1	-	20.6	-	24.0	-
	$n_{\mathrm{p}} = 4$	17.8	-	21.3	-	23.8	-
	$n_{\mathrm{p}} = 10$	25.4	-	30.9	-	32.9	-
	$n_{\mathrm{p}} = 20$	34.6	-	36.1	-	39.1	-
	$n_{\mathrm{p}} = 40$	37.8	*	38.7	*	39.3	
	$n_{\mathrm{p}} = 100$	47.3	**	47.5	**	48.8	**
提案手法	$n_{\mathrm{c}} = 2$	24.7	-	25.0	-	30.1	-
	$n_{\mathrm{c}} = 4$	25.1		28.7		31.5	
	$n_{\mathrm{c}} = 6$	29.4	*	30.2		34.7	

表 5.18　環境 2 における従来手法と提案手法の計算時間

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_o \approx 210$		$n_o \approx 420$		$n_o \approx 630$	
		平均時間 [ms]	P 値	平均時間 [ms]	P 値	平均時間 [ms]	P 値
従来手法	$n_p = 2$	21.0		25.6		29.7	
	$n_p = 4$	21.9		32.7		35.5	
	$n_p = 10$	31.0		36.0		37.6	
	$n_p = 20$	36.7		38.3		38.5	
	$n_p = 40$	37.8		40.1		40.2	
	$n_p = 100$	46.8	**	49.1	**	50.0	**
提案手法	$n_c = 2$	28.1		28.7		34.5	
	$n_c = 4$	28.8		32.9		36.1	
	$n_c = 6$	33.7	*	34.6		39.8	*

表 5.19と表 5.20に, 1フレームあたりの経路生成と衝突検知の計算時間の比較を示す．t_1は従来手法でのt_{B1}と提案手法でのt_{P1}を表す．t_2は従来手法でのt_{B2}と提案手法でのt_{P2}を表す．経路生成の計算時間t_1を観察すると，予想ほど減少しないことが確認できる．衝突検知の計算時間t_2を観察すると，提案手法での計算時間は従来手法よりもおよそ90%減少した．2つのステップの合計計算時間は表 5.17と表 5.18に示す1フレームあたりのDWAの計算時間の99%を占める．両手法ともに，衝突検知には簡単な点群間距離の計算を行うため，繰り返す回数の減少は計算時間を短縮できる. 例えば, 環境2における, $n_o \approx 130$, $n_c = 2$の場合，繰り返す回数は約260，およそ$n_p = 20$の場合の10%，計算時間は$n_p = 20$の場合の77%になる．経路生成の計算時間は予想ほど減少しなかった理由は以下と考えられる．5.2.3節で説明した計算手法を使用した．加速度に関する条件判別が多いため，計算は遅くなる．また，プログラムにも改善する余地がある．

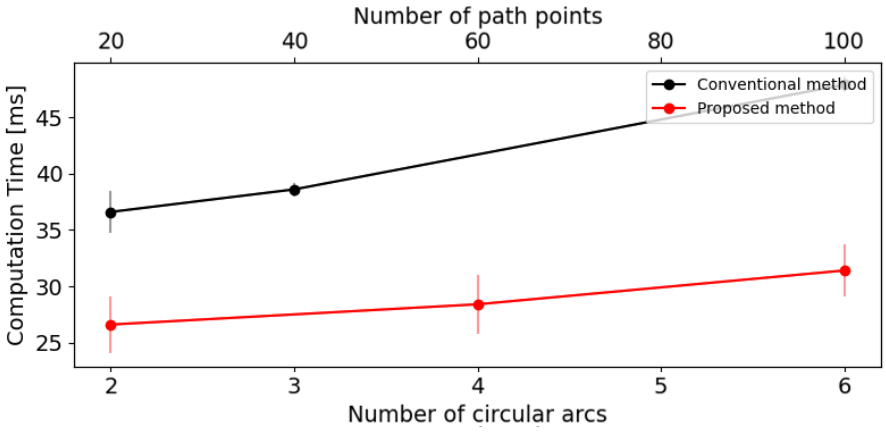
表 5.19　環境 1 における従来手法と提案手法の計算時間（ステップ別）

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_o \approx 130$		$n_o \approx 260$		$n_o \approx 390$	
		t_1	t_2	t_1	t_2	t_1	t_2
従来手法	$n_p = 20$	26.6	7.6	25.3	10.5	25.4	13.3
	$n_p = 40$	29.1	7.9	27.5	10.8	25.5	13.0
	$n_p = 100$	36.4	9.9	33.7	13.3	32.2	16.1
提案手法	$n_c = 2$	24.0	0.5	24.3	0.5	28.6	1.2
	$n_c = 4$	24.3	0.5	27.8	0.9	29.9	1.3
	$n_c = 6$	28.5	0.6	29.0	0.9	33.0	1.7

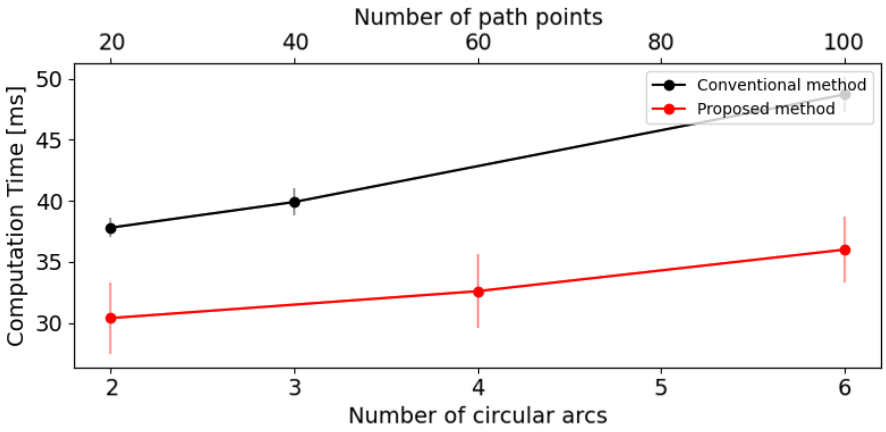
表 5.20　環境 2 における従来手法と提案手法の計算時間（ステップ別）

角度分解能 [nums/°]		1		2		3	
1 フレームあたり検出された障害物点の数		$n_o \approx 210$		$n_o \approx 420$		$n_o \approx 630$	
		t_1	t_2	t_1	t_2	t_1	t_2
従来手法	$n_p = 20$	28.3	8.1	26.8	11.1	25.0	13.1
	$n_p = 40$	29.1	7.9	28.5	11.2	26.1	13.3
	$n_p = 100$	36.0	9.8	34.9	13.7	33.0	16.5
提案手法	$n_c = 2$	27.3	0.6	27.8	0.6	32.8	1.4
	$n_c = 4$	27.9	0.6	31.9	1.0	34.3	1.4
	$n_c = 6$	32.7	0.7	33.2	1.0	37.8	2.0

与えられたn_pとn_cの値に基づいて, 異なるn_oの値に対する平均計算時間を計算し, n_pおよびn_cと計算時間の関係を調査した. 結果を図 5.55に表す. 図 5.55の横軸は従来手法における経路点の数n_p, 提案手法における円弧の数n_cを表す. 縦軸は1フレームあたりのDWAの計算時間を表す. エラーバーはn_oの変化による偏差を示す. n_oの値が大きくなるにつれて, 計算時間が全体的に増加した. 環境1と環境2ともに, 提案手法の計算時間は従来手法よりも常に短いことや, 従来手法ではn_pが大きくなるにつれて計算時間が大幅に増加していることがわかる.



(a) 環境 1



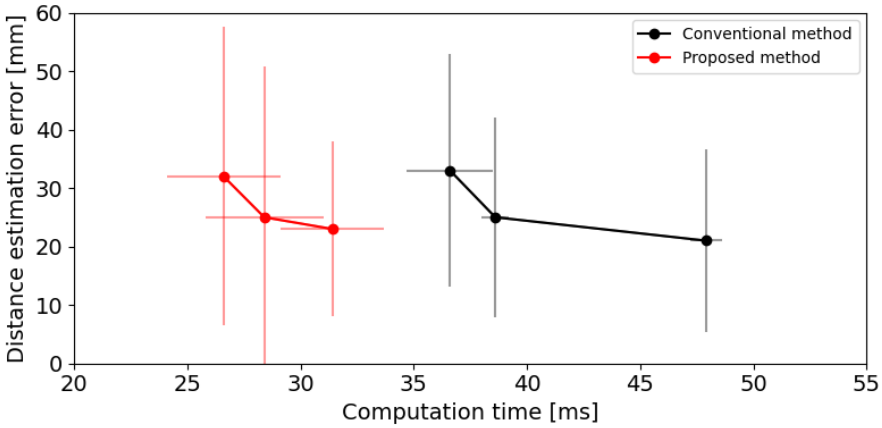
(b) 環境 2

図 5.55　計算時間の比較

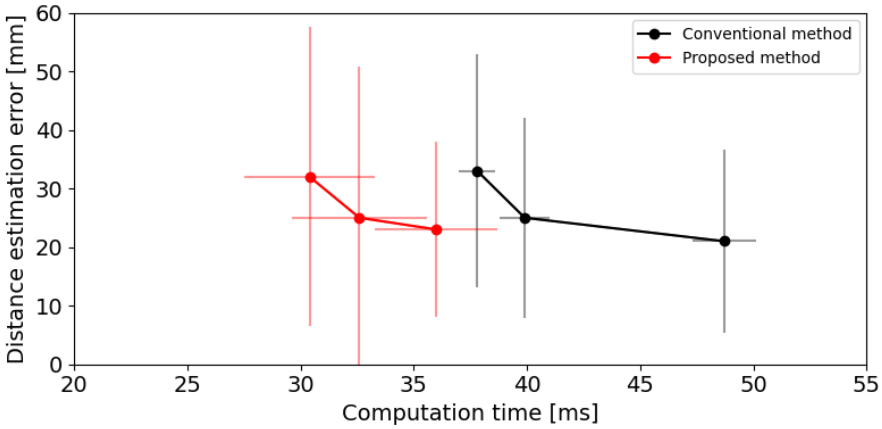
5.3.3　結果分析

実験1では，ランダムに生成された障害点から，各条件で生成された予測経路までの距離により，提案手法と従来手法の距離推定誤差を調査した．誤差の最大値を捕えるために，実験1のパラメータは，予測期間を通じてより広い速度変化範囲を包含するように選択された．従って，実験2で得られた距離推定誤差は，実験1の誤差を上回らなかったと考えられる．実験1で得られた距離推定精度と実験2で得られた計算時間を統合し，従来手法と提案手法の品質コスト(cost of quality)曲線を作成した．品質コスト曲線は，具体的なパラメータ設定を省き，計算時間と距離推定精度のトレードオフ関係を明らかにできる．結果を図 5.56に示す．図の横軸は計算時間，縦軸は距離推定精度を表す．手法の曲線が原点に近いほど優れた性能を示す．環境1と環境2の結果はともに，計算時間と距離推定精度の間に明らかなトレードオフ関係が見られる．さらに，提

案手法の曲線は，従来手法よりも原点に近く，勾配が急である．原点までの近さは，同等の計算時間でより高い距離推定精度を意味し，同等の距離推定精度の要求でより短い計算時間を意味する．また，より急な勾配は，同様の精度を向上させるために，消費した追加時間が少ないことを意味する．このように，提案手法は従来手法に比べて優れた性能を示す．



(a) 環境 1



(b) 環境 2

図 5.56　品質コスト曲線

5.4　考察

従来のDWAでは，選択された速度系列をロボットの運動学モデルに代入を繰り返すことで，離散的な経路点を計算し，それを予測期間におけるロボットの経路を表す．繰り返し計算と衝突検知ための点群間距離計算により，総計算時間が長い．そこで，本研究では，円弧を用いて予測経路の離散的な経路点をモデリングし，衝突検知を行う手法を提案した．提案手法では，経路予測に従

速度制御モデルを利用する場合，円弧を用い経路を表現し，衝突検出を簡単にする．さらに，経路予測に変速モデルを利用する場合，複数の円弧を用い経路を近似し，衝突検出を行うように手法を拡張した．

研究向けの移動ロボットプラットフォームも商業用ロボットも，一般に最大速度が2[m/s]以下，加速度が1[$\mathrm{m/s^2}$]以下，すなわち低速環境で動作していることがわかる[64]-[69]．そのため，実験1の実験条件に最大並進速度を2[m/s]，加速度を1[$\mathrm{m/s^2}$]に設定した．また，角速度を1[rad/s]に設定した，それは生成される予測経路が渦巻状になりすぎないようにするためである．多く研究では，はっきり書かれていないが，生成される予測経路が渦巻状になりすぎないように，パラメータをコントロールしていた．その理由は議論されてないが，本研究の場合，予測経路が渦巻状になりすぎると提案手法が失敗する可能性がある．

また，実験2では，衝突検知の計算時間を比較した．環境2で，提案手法$n_{\mathrm{c}} = 2$の場合の繰り返し回数は，従来手法$n_{\mathrm{p}} = 20$と比べ90%の減少があった．全体の計算時間は23%減少し，　t_1は4%，t_2は93%減少した．t_2の結果を見ると，円弧を用い予想経路を近似する手法は，衝突検知の計算時間の減少に有効であることが示される．しかし，円弧の中心を計算する手法は複雑すぎるため，t_1の減少は小さい．さらなる高速化のためには，円弧中心の計算方法を改善することが求められる．

本研究では，差動車輪付きロボットの自律移動タスクを用い，従来手法と提案手法の品質コストを比較した．経路の円弧近似により，衝突検知に必要な計算時間を短縮できることが確認された．本手法は自動車や全方位ロボットなどの様々な自律移動ロボットに適用可能であると考えられる．

第 6 章　結論

本論文では，広く用いられているローカル経路計画手法であるDWAに躍度制御モデルを加えて移動ロボットを滑らかに制御できる手法を提案した. また, 速度制御モデルおよび加速制御モデルで動作するＤＷＡの高速化手法を提案した．従来のDWAでは，実現可能な速度指令をサンプリングし，それに基づいて予測経路を生成する．これらの予測経路を評価し，最適な予測経路に対する速度指令出力することを繰り返すことで，ロボットを移動させる．しかし，その速度指令を実行する際にロボットにどの程度の躍度が生じるかが考慮されていない．躍度とは加速度の時間微分であり，躍度が大きくなると，力積の増大を招くため, ロボットが転倒したり, 内部の部品にダメージを与えたりする．そのため，移動時にロボットに発生する躍度を制御する必要がある．

第4章では，躍度で制御可能なDWAを提案した．従来のDWAでは並進速度と角速度を選択するが，提案手法では，並進躍度と角躍度を選択する．また，従来は，予測期間中の速度が一定であることを仮定するが，提案手法では，予測期間中の躍度が一定であることを仮定し, 速度と加速度には一定を仮定しない. これにより，従来よりも現実に即した経路計画が可能とした．さらに，躍度を抑えた滑らかな加減速を計画することもできる. シミュレーション環境実験により，提案手法は躍度を抑えながら，障害物を回避する経路を計画できることを示すと共に，実環境実験により，ロボット側の応答に遅れが発生する場合にも，躍度の制御が有効であることを示した.

また, 従来のDWAでは離散的な経路点を用いて予測経路を表しており, 経路点を生成するための繰り返し計算が必要である．衝突検知を行う際には，膨大な経路点と障害物点の点群間距離の最小値を求める必要があり時間がかかる.

第5章では, 予測経路を離散的な経路点ではなく円弧を用いてモデリングし, 衝突検知を行う手法を提案した．経路予測に速度制御モデルを利用する場合，円弧で経路を近似することで，衝突検出の計算を容易にする．ただし，経路予測に変速モデルを利用する場合には, その経路が円弧にはならないという問題がある．この問題に対して，複数の円弧を用いて経路を近似し，衝突検出を行う手法を提案した．実験により，提案手法は，従来手法に優れた性能を持つsecantモデルと同等の距離推定精度を持っていることを示した．提案手法の経路生成ステップの計算時間の減少量は少ないが, 衝突検知ステップの計算時間は大幅に減少させることができる. 提案手法は従来手法と同等の計算時間では

より高い距離推定精度を持ち, 従来手法と同等の距離推定精度の要求ではより短い計算時間を消費する.

本研究の成果は, 滑らかな動作制御と速やかな反応を持つロボットの発展に貢献すると考えられる.

今後の課題について述べる. 従来のDWAは動的な障害物もある程度回避する能力を持っている. そのため, 動的な環境で提案手法の効果を検証する必要がある. 提案手法は, 躍度を抑えながら動的な障害物を回避できたり, 障害物と衝突しないために緊急避難的にあえて急加減速を発生させたりすることができると考えている. ただし, 動的な障害物の速度, 密度, 移動経路, ロボットとの相対位置関係などの設定が結果に影響を与えるため, かなり複雑な課題と考えられる. また, 評価関数の重みの自動調整について, 4.3.4節の結果により, 凸最適化問題として扱うことができないため, 4.4節に述べた1つの重みの数を減少する手法を検証する価値があると考えられる.

円弧近似により高速化手法では, 躍度制御モデルを運動学モデルに代入して積分するとフレネル正弦積分項と余弦積分項[70]が出るため, 計算は相当複雑になる. 積分せずに円弧の中心の位置を計算する手法が望まれる. また, DWAはすべての予測経路を評価しているが, 無駄な評価も含まれていると考えられる. 現在の評価関数が凸関数の性質を持つかどうか検証し, そうでないのであれば, 凸関数の性質を持つ評価関数を作れば, ニュートン法のような凸最適化手法を用い, 最適な速度指令が迅速に求められると考えられる.

謝辞

本研究を行うにあたり，終始懇切丁寧なるご指導を頂きました名古屋工業大学大学 田口亮准教授に心から感謝いたします．

本論文の審査をしていただくとともに，有益なご意見をいただきました名古屋工業大学 加藤昇平教授，坂上文彦准教授，中部大学 梅崎太造教授にお礼申し上げます．

最後に，本研究の遂行にあたり，御協力と助言を頂きました田口研究室の皆様に感謝いたします．

参考文献

[1] Mohanty, Prases K., and Dayal R. Parhi. "Controlling the motion of an autonomous mobile robot using various techniques: a review." Journal of Advance Mechanical Engineering 1.1 (2013): 24-39.

[2] Pandey, Anish, Shalini Pandey, and D. R. Parhi. "Mobile robot navigation and obstacle avoidance techniques: A review." *Int Rob Auto J* 2.3 (2017): 00022.

[3] Gul, Faiza, Wan Rahiman, and Syed Sahal Nazli Alhady. "A comprehensive study for robot navigation techniques." *Cogent Engineering* 6.1 (2019): 1632046.

[4] Zhang, Han-ye, Wei-ming Lin, and Ai-xia Chen. "Path planning for the mobile robot: A review." Symmetry 10.10 (2018): 450.

[5] Patle, B. K., et al. "A review: On path planning strategies for navigation of mobile robot." Defence Technology 15.4 (2019): 582-606.

[6] Elbanhawi, Mohamed, and Milan Simic. "Sampling-based robot motion planning: A review." Ieee access 2 (2014): 56-77.

[7] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance." IEEE Robotics & Automation Magazine 4.1 (1997): 23-33.

[8] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[9] Bahón, Cecilio Angulo, and Gaspar Valls Solé. "A jerk threshold-based involuntary lateral movement algorithm." 2009 IEEE Conference on Emerging Technologies & Factory Automation. IEEE, 2009.

[10] Klenk, Jochen, et al. "Comparison of acceleration signals of simulated and real-world backward falls." Medical engineering & physics 33.3 (2011): 368-373.

[11] Deshmukh, Akshay, et al. "Study of frequency characteristics of vehicle motions for the derivation of inherent jerk." SAE International Journal of Passenger Cars-Mechanical Systems 9.2016-01-1681 (2016): 419-423.

[12] Hayati, H.; Eager, D.; Pendrill, A.-M.; Alberg, H. Jerk within the Context of Science and Engineering—A Systematic Review. Vibration 2020, 3, 371-409.

[13] Bertolin, Chiara, et al. "Analysis of Jerk as a novel tree-fails hazard index: A case study applied to tree monitoring in the archaeological park of the Colosseum in Rome (Italy)." International journal of disaster risk reduction 56 (2021): 102122.

[14] 王鋒，佐川貢一，猪岡光，“自動車の加減速と乗り心地の関係に関する研究,”人間工学, vol.36, no.4, pp.191-200, 2000.

[15] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. Vol. 1. Ieee, 2001.

[16] Lowe, David G. "Object recognition from local scale-invariant features." Proceedings of the seventh IEEE international conference on computer vision. Vol. 2. Ieee, 1999.

[17] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9. Springer Berlin Heidelberg, 2006.

[18] Vapnik, Vladimir N. "Pattern recognition using generalized portrait method." Automation and

remote control 24.6 (1963): 774-780.
[19] Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." Proceedings of the fifth annual workshop on Computational learning theory. 1992.
[20] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.
[21] Huang, Lichao, et al. "Densebox: Unifying landmark localization with end to end object detection." arXiv preprint arXiv:1509.04874 (2015).
[22] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[23] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.
[24] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
[25] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.
[26] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).
[27] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.
[28] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[29] Fox, Dieter, et al. "Monte carlo localization: Efficient position estimation for mobile robots." Aaai/iaai 1999.343-349 (1999): 2-2.
[30] Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter. Probabalistic Robotics. The MIT Press. p. 309.
[31] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022. 287-290.
[32] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.
[33] Stentz, Anthony. "Optimal and efficient path planning for partially-known environments." Proceedings of the 1994 IEEE international conference on robotics and automation. IEEE, 1994.
[34] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." The international journal of robotics research 5.1 (1986): 90-98.
[35] Borenstein, Johann, and Yoram Koren. "The vector field histogram-fast obstacle avoidance for mobile robots." IEEE transactions on robotics and automation 7.3 (1991): 278-288.
[36] Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. IEEE, 2000.
[37] LaValle, Steven M., James J. Kuffner, and B. R. Donald. "Rapidly-exploring random trees: Progress and prospects." Algorithmic and computational robotics: new directions 5 (2001): 293-308.
[38] Karaman, Sertac, and Emilio Frazzoli. "Incremental sampling-based algorithms for optimal motion planning." Robotics Science and Systems VI 104.2 (2010): 267-274.

[39] Simmons, Reid. "The curvature-velocity method for local obstacle avoidance." Proceedings of IEEE international conference on robotics and automation. Vol. 4. IEEE, 1996.

[40] Vilanova, Ramon, and Antonio Visioli. PID control in the third millennium. Vol. 75. No. 417. London: Springer, 2012.

[41] Kouvaritakis, B., Cannon, M. (2016). Introduction. In: Model Predictive Control. Advanced Textbooks in Control and Signal Processing. Springer, Cham.

[42] Kuhne, Felipe, Walter Fetter Lages, and J. Gomes da Silva Jr. "Model predictive control of a mobile robot using linearization." Proceedings of mechatronics and robotics. 2004.

[43] Künhe, F., J. Gomes, and W. Fetter. "Mobile robot trajectory tracking using model predictive control." II IEEE latin-american robotics symposium. Vol. 51. 2005.

[44] Kanjanawanishkul, Kiattisin, and Andreas Zell. "Path following for an omnidirectional mobile robot based on model predictive control." 2009 IEEE International Conference on Robotics and Automation. IEEE, 2009.

[45] Pacheco, Lluis, and Ningsu Luo. "Testing PID and MPC performance for mobile robot local path-following." International Journal of Advanced Robotic Systems 12.11 (2015): 155.

[46] Quang, Hiep Do, et al. "Design a nonlinear MPC controller for autonomous mobile robot navigation system based on ROS." International Journal of Mechanical Engineering and Robotics Research 11.6 (2022).

[47] Magni, Lalo, and Riccardo Scattolini. "On the solution of the tracking problem for non-linear systems with MPC." International journal of systems science 36.8 (2005): 477-484.

[48] Shah, Gaurang, and Sebastian Engell. "Tuning MPC for desired closed-loop performance for MIMO systems." Proceedings of the 2011 American Control Conference. IEEE, 2011.

[49] Dhar, Abhishek, and Shubhendu Bhasin. "Adaptive MPC for uncertain discrete-time LTI MIMO systems with incremental input constraints." IFAC-PapersOnLine 51.1 (2018): 329-334.

[50] Ferreau, Hans Joachim, Hans Georg Bock, and Moritz Diehl. "An online active set strategy to overcome the limitations of explicit MPC." International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 18.8 (2008): 816-830.

[51] Limón, Daniel, et al. "MPC for tracking piecewise constant references for constrained linear systems." Automatica 44.9 (2008): 2382-2387.

[52] Díaz-García, Gilberto, Luis Felipe Giraldo, and Santiago Jimenez-Leudo. "Dynamics of a differential wheeled robot: control and trajectory error bound." 2021 IEEE 5th Colombian Conference on Automatic Control (CCAC). IEEE, 2021.

[53] Mújica-Vargas, Dante, et al. "Navigation of a Differential Wheeled Robot Based on a Type-2 Fuzzy Inference Tree." Machines 10.8 (2022): 660.

[54] Raziei, Seyed Ata, and Zhenhua Jiang. "Nonlinear Model Predictive Motion Control of Differential Wheeled Robots." NAECON 2018-IEEE National Aerospace and Electronics Conference. IEEE, 2018.

[55] Chen, Qiyuan, Pengfei Zhang, and Tingting Yang. "Differential Wheel Robot Fixed Posture Positioning Control." 2022 34th Chinese Control and Decision Conference (CCDC). IEEE, 2022.

[56] Yang, Dongsheng, et al. "DRE-SLAM: Dynamic RGB-D encoder SLAM for a differential-drive robot." Remote Sensing 11.4 (2019): 380.

[57] Gerkey, Brian P., and Kurt Konolige. "Planning and control in unstructured terrain." ICRA workshop on path planning on costmaps. 2008.

[58] Dwa_local_planner - ROS Wiki. (n.d.). ROS.Org. http://wiki.ros.org/dwa_local_planner/ (accessed

on 26 November 2023).
[59] Y. Kang, D. A. de Lima and A. C. Victorino, "An approach of human driving behavior correction based on Dynamic Window Approach," 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, 2014, pp. 304-309.
[60] Wang, Jingke, et al. "Learning hierarchical behavior and motion planning for autonomous driving." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.
[61] Kobayashi, Masato, and Naoki Motoi. "Local path planning method based on virtual manipulators and dynamic window approach for a wheeled mobile robot." 2021 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2021.
[62] Haschke, Robert, Erik Weitnauer, and Helge Ritter. "On-line planning of time-optimal, jerk-limited trajectories." 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2008.
[63] Gerkey, Brian, Richard T. Vaughan, and Andrew Howard. "The player/stage project: Tools for multi-robot and distributed sensor systems." Proceedings of the 11th international conference on advanced robotics. Vol. 1. 2003.
[64] An Index of ROS Robots. Available online: https://robots.ros.org/ (accessed on 18 October 2023).
[65] Mobile Industrial Robots. Automate Your Internal Transportation. Available online: https://www.mobile-industrial-robots.com/ (accessed on 19 October 2023).
[66] Moving Robot. Available online: https://www.hansrobot.net/product-center/yidongjiqiren/ (accessed on 19 October 2023).
[67] Automated & Autonomous Mobile Robots|Robotnik®. Available online: https://robotnik.eu/products/mobile-robots/ (accessed on 19 October 2023).
[68] AITEN AGV (China) Official Site|—Satisfying Every Real Demand in Factory. Available online: https://www.szaiten.com/en/ProductIndex/ (accessed on 19 October 2023).
[69] AMR Autonomous Mobile Robots|AMS, Inc. Available online: https://www.ams-fa.com/autonomous-mobile-robots/ (accessed on 19 October 2023).
[70] Weisstein, Eric W. "Fresnel Integrals". mathworld.wolfram.com https://mathworld.wolfram.com/FresnelIntegrals.html/ (accessed on 27 November 2023).

研究業績一覧

学術論文（査読あり）　主：2，副：0

[1] **Ziang Lin**, Ryo Taguchi, "Faster Implementation of The Dynamic Window Approach Based on Non-Discrete Path Representation", Mathmatics, 11, No. 22 : 4424, October 2023. (doi.org/10.3390/math11214424)
https://www.mdpi.com/2227-7390/11/21/4424

[2] **林 子昂**, 田口 亮, “躍度モデルに基づいたDWAの改良手法”,日本ロボット学会誌 （2023年11月10日に採録通知）

国際会議（査読あり）　主：2，副：0

[3] **Ziang Lin**, Ryo Taguchi, "Improved Dynamic Window Approach using the jerk model", In Proc. of IEEE 2022 22nd International Conference on Control, Automation and Systems (ICCAS), pp.1193 -- 1198, 2022. (doi: 10.23919/ICCAS55662.2022.10003869)
https://ieeexplore.ieee.org/document/10003869

[4] **Ziang Lin**, Ryo Taguchi, "Faster Implementation of the Dynamic Window Approach via Polar Coordinate Transformation", In Proc. of IEEE 2023 International Conference on Mechatronics and Automation (ICMA), pp.525 -- 530, 2023. (Best Paper Finalist, doi: 10.1109/ICMA57826.2023.10216239)
https://ieeexplore.ieee.org/document/10216239.

国内会議（査読なし）　主：3，副：0

[5] 林 子昂，田口 亮，“車両外観検査ロボットのための画像認識を用いた制御システムの開発”,第37回日本ロボット学会学術講演会，1H2-07，2019.

[6] 林 子昂，田口 亮，“CRNN を用いた車両外観検査ロボットの制御システムの改良”,計測自動制御学会 システム・情報部門 学術講演会，GS02-11，2019.

[7] 林 子昂, 田口 亮, “加加速度モデルに基づいたDWAの改良手法”,第40回日本ロボット学会学術講演会，2I2-02，2022.