

卒業論文

Navigation2 における パラメータ調整でのロボットの挙動の調査

Investigation of robot behavior

when adjusting parameters in Navigation2

2025年12月30日 提出

指導教員 林原 靖男 教授

千葉工業大学 先進工学部 未来ロボティクス学科

22C1085 坪内優輝

概要

Navigation2 におけるパラメータ調整での ロボットの挙動の調査

本論文では、ROS 2 における Navigation2 を用いた自律移動ロボットのパラメータ調整が、ロボットの挙動に与える影響について調査する。本研究室では、屋外自律移動ロボットの研究を行っており、津田沼チャレンジやつくばチャレンジといった実環境での自律走行競技に参加している。

Navigation2 を用いた自律移動では、自己位置推定、経路計画、障害物回避などを実現するために多数のパラメータを適切に設定する必要がある。しかし、これらのパラメータがロボットの挙動にどのような影響を与えるかについては十分に整理されておらず、特に屋外環境においては調整が困難である。実際にロボットの調整を行う過程においても、パラメータ設定によって意図しない挙動や走行性能の低下が生じる問題が確認された。

そこで本研究では、Navigation2 における各種パラメータを変化させた際のロボットの挙動の変化を明らかにすることを目的とする。ロボットの挙動に影響を与えると考えられるパラメータを対象として走行実験を行い、それぞれの設定値における走行挙動を比較・分析した。

実験の結果、調査対象としたパラメータの中から、ロボットの走行安定性や経路追従性能に大きな影響を与えるものを確認することができた。これらの結果から、Navigation2 におけるパラメータ調整の指針を得ることができ、今後のロボット調整作業の効率化や、問題の挙動の原因究明に有用であることが示された。

キーワード: 屋外自律移動ロボット、ナビゲーション、パラメータ

abstract

Investigation of robot behavior when adjusting parameters in Navigation2

This thesis investigates the effects of parameter tuning in Navigation2 on the behavior of autonomous mobile robots using ROS 2. Our laboratory conducts research on outdoor autonomous mobile robots and participates in real-world autonomous navigation competitions such as the Tsudanuma Challenge and the Tsukuba Challenge. Autonomous navigation using Navigation2 requires appropriate tuning of numerous parameters to achieve functions such as self-localization, path planning, and obstacle avoidance. However, the influence of these parameters on robot behavior has not been sufficiently organized, and parameter tuning remains particularly challenging in outdoor environments. In practice, during the tuning process of the robot, issues such as unintended behavior and degradation of navigation performance were observed depending on parameter settings. Therefore, the objective of this study is to clarify the changes in robot behavior caused by variations in parameters in Navigation2. Driving experiments were conducted by adjusting parameters that are considered to significantly affect robot behavior, and the resulting navigation behaviors under different parameter settings were compared and analyzed. Experimental results confirmed that several of the investigated parameters have a significant impact on the stability of robot motion and path-following performance. These findings provide useful guidelines for parameter tuning in Navigation2 and contribute to improving the efficiency of robot configuration and identifying the causes of problematic behaviors.

keywords: outdoor autonomous mobile robot, navigation, parameters

目次

第 1 章	序論	1
1.1	背景	1
1.2	目的	2
1.3	論文の構成	2
第 2 章	要素技術	3
2.1	ROS2	3
2.2	Navigation2	3
2.3	emcl2	4
第 3 章	パラメータ概要	6
3.1	実験で使うパラメータ	6
3.2	パラメータの詳細	8
3.2.1	emcl2	8
3.2.2	Navigation2	9
第 4 章	実験	13
4.1	実験概要	13
4.2	実験結果	15
4.2.1	実験結果 (emcl2)	15
4.2.2	実験結果 (Nav2_controller)	36
4.2.3	実験結果 (Nav2_Costmap)	50
4.2.4	実験結果 (Nav2_Velocity Smoother)	51

4.2.5 実験結果 (Nav2_planner goal)	51
第5章 結論	66
5.1 まとめ	66
5.1.1 emcl2 パラメータ調整に関する総括	66
参考文献	68
付録	69
謝辞	70

図目次

2.1	Structure of Navigation2	4
4.1	Course map of the Tsudanuma Challenge 2025(source: [5])	14
4.2	ORNE-box3(source: [4])	14
4.3	sensor configuration	14
4.4	base orientation	16
4.5	base position	17
4.6	Robot orientation with fw_dev_per_fw = 0.01	17
4.7	Robot position with fw_dev_per_fw = 0.01	18
4.8	Robot orientation with fw_dev_per_fw = 0.05	18
4.9	Robot position with fw_dev_per_fw = 0.05	18
4.10	Robot orientation with fw_dev_per_fw = 0.1	19
4.11	Robot position with fw_dev_per_fw = 0.1	19
4.12	Robot orientation with fw_dev_per_fw = 0.3	19
4.13	Robot position with fw_dev_per_fw = 0.3	20
4.14	Robot orientation with fw_dev_per_fw = 0.4	20
4.15	Robot position with fw_dev_per_fw = 0.4	20
4.16	Robot orientation with fw_dev_per_fw = 0.5	21
4.17	Robot position with fw_dev_per_fw = 0.5	21
4.18	Robot orientation with fw_dev_per_rot = 0.001	22
4.19	Robot position with fw_dev_per_rot = 0.001	22
4.20	Robot orientation with fw_dev_per_rot = 0.01	22

4.21	Robot position with fw_dev_per_rot = 0.01	23
4.22	Robot orientation with fw_dev_per_rot = 0.1	23
4.23	Robot position with fw_dev_per_rot = 0.1	23
4.24	Robot orientation with fw_dev_per_rot = 0.5	24
4.25	Robot position with fw_dev_per_rot = 0.5	24
4.26	Robot orientation with rot_dev_per_fw = 0.01	25
4.27	Robot position with rot_dev_per_fw = 0.01	25
4.28	Robot orientation with rot_dev_per_fw = 0.2	25
4.29	Robot position with rot_dev_per_fw = 0.2	26
4.30	Robot orientation with rot_dev_per_fw = 0.3	26
4.31	Robot position with rot_dev_per_fw = 0.3	26
4.32	Robot orientation with rot_dev_per_fw = 0.4	27
4.33	Robot position with rot_dev_per_fw = 0.4	27
4.34	Robot orientation with rot_dev_per_fw = 0.5	27
4.35	Robot position with rot_dev_per_fw = 0.5	28
4.36	Robot orientation with rot_dev_per_rot = 0.1	29
4.37	Robot position with rot_dev_per_rot = 0.1	29
4.38	Robot orientation with rot_dev_per_rot = 0.3	29
4.39	Robot position with rot_dev_per_rot = 0.3	30
4.40	Robot orientation with rot_dev_per_rot = 0.4	30
4.41	Robot position with rot_dev_per_rot = 0.4	30
4.42	Robot orientation with rot_dev_per_rot = 0.5	31
4.43	Robot position with rot_dev_per_rot = 0.5	31
4.44	Robot orientation with laser_likelihood_max_dist = 0.5	32
4.45	Robot position with laser_likelihood_max_dist = 0.5	32
4.46	Robot orientation with laser_likelihood_max_dist = 1.0	33
4.47	Robot position with laser_likelihood_max_dist = 1.0	33
4.48	Robot orientation with range_threshold = 0.5	34
4.49	Robot position with range_threshold = 0.5	34

4.50	Robot orientation with range_threshold = 0.9	34
4.51	Robot position with range_threshold = 0.9	35
4.52	Robot orientation with max_vel_x = 0.2	36
4.53	Robot position with max_vel_x = 0.2	36
4.54	Robot orientation with max_vel_x = 1.0	37
4.55	Robot position with max_vel_x = 1.0	37
4.56	Robot orientation with max_vel_theta = 0.4	38
4.57	Robot position with max_vel_theta = 0.4	38
4.58	Robot orientation with max_vel_theta = 1.0	39
4.59	Robot position with max_vel_theta = 1.0	39
4.60	Robot orientation with max_speed_xy = 0.1	40
4.61	Robot position with max_speed_xy = 0.1	40
4.62	Robot orientation with max_speed_xy = 0.5	41
4.63	Robot position with max_speed_xy = 0.5	41
4.64	Robot orientation with max_speed_xy = 1.5	41
4.65	Robot position with max_speed_xy = 1.5	42
4.66	Robot orientation with min_theta_velocity_threshold = 0.01	43
4.67	Robot position with min_theta_velocity_threshold = 0.01	43
4.68	Robot orientation with min_theta_velocity_threshold = 0.1	43
4.69	Robot position with min_theta_velocity_threshold = 0.1	44
4.70	Robot orientation with min_theta_velocity_threshold = 1.0	44
4.71	Robot position with min_theta_velocity_threshold = 1.0	44
4.72	Robot orientation with acc_lim_x = 0.1	45
4.73	Robot position with acc_lim_x = 0.1	45
4.74	Robot orientation with acc_lim_x = 0.5	46
4.75	Robot position with acc_lim_x = 0.5	46
4.76	Robot orientation with acc_lim_x = 1.5	46
4.77	Robot position with acc_lim_x = 1.5	47
4.78	Robot orientation with acc_lim_theta = 0.1	47

4.79	Robot position with acc_lim_theta = 0.1	48
4.80	Robot orientation with acc_lim_theta = 1.0	48
4.81	Robot position with acc_lim_theta = 1.0	48
4.82	Robot orientation with acc_lim_theta = 4.0	49
4.83	Robot position with acc_lim_theta = 4.0	49
4.84	Robot orientation with linear_granularity = 0.01	52
4.85	Robot position with linear_granularity = 0.01	52
4.86	Robot orientation with linear_granularity = 0.1	52
4.87	Robot position with linear_granularity = 0.1	53
4.88	Robot orientation with linear_granularity = 0.5	53
4.89	Robot position with linear_granularity = 0.5	53
4.90	Robot orientation with angular_granularity = 0.001	54
4.91	Robot position with angular_granularity = 0.001	55
4.92	Robot orientation with angular_granularity = 0.01	55
4.93	Robot position with angular_granularity = 0.01	55
4.94	Robot orientation with angular_granularity = 0.04	56
4.95	Robot position with angular_granularity = 0.04	56
4.96	Robot orientation with xy_goal_tolerance = 0.01	57
4.97	Robot position with xy_goal_tolerance = 0.01	57
4.98	Robot orientation with xy_goal_tolerance = 0.1	58
4.99	Robot position with xy_goal_tolerance = 0.1	58
4.100	Robot orientation with xy_goal_tolerance = 0.4	58
4.101	Robot position with xy_goal_tolerance = 0.4	59
4.102	Robot orientation with trans_stopped_velocity = 0.01	59
4.103	Robot position with trans_stopped_velocity = 0.01	60
4.104	Robot orientation with trans_stopped_velocity = 0.1	60
4.105	Robot position with trans_stopped_velocity = 0.1	60
4.106	Robot orientation with trans_stopped_velocity = 0.4	61
4.107	Robot position with trans_stopped_velocity = 0.4	61

4.108	Robot orientation with planner_tolerance = 0.1	62
4.109	Robot position with planner_tolerance = 0.1	62
4.110	Robot orientation with planner_tolerance = 1.0	63
4.111	Robot position with planner_tolerance = 1.0	63
4.112	Robot speed with max_vel_x = 0.2	64
4.113	Robot speed with max_vel_x = 1.0	64

表目次

3.1	Parameter List in emcl2	6
3.2	Parameter List of the Nav2 Controller	7
3.3	Parameters of the Nav2 Costmap	7
3.4	Parameters of the Nav2 Velocity Smoother	7
3.5	Parameters of the Nav2 Planner and Goal Checker	8
4.1	パーティクル数の違いによるゴール到達可否	15
4.2	パーティクル数ごとのゴール到達成功率	15

第1章

序論

1.1 背景

近年，屋外自律移動ロボットの研究が盛んに行われてあり，ROS 2 を用いた自律移動システムの実環境への適用が進んでいる．本研究室では，屋外環境における自律移動ロボットの研究を行っており，津田沼チャレンジやつくばチャレンジといった実環境での自律走行競技に参加している．

これらの競技では，ロボットが屋外環境を安定して走行するために，高いナビゲーション性能が求められる．Navigation2 を用いた自律移動では，自己位置推定，経路計画，障害物回避などを実現するために，多数のパラメータを適切に調整する必要がある．しかし，ロボットの調整作業を行う過程において，パラメータ設定によって意図しない挙動や走行性能の低下が生じることが確認された．これらの問題は，パラメータの数が多く，それぞれがロボットの挙動に与える影響が分かりにくいことに起因していると考えられる．そのため，Navigation2 におけるパラメータを変化させた際のロボットの挙動を整理し，両者の関係を明らかにすることは，今後のパラメータ調整の効率化や，問題の挙動の原因究明において重要である．

1.2 目的

Navigation2におけるパラメータ調整でのロボットの挙動の変化を調べることを目的とする

1.3 論文の構成

本論文では以下のように構成される

2章では本研究で使用される要素技術

3章では調査するパラメータの概要

4章では実験について

5章では本論文の結論

第2章

要素技術

2.1 ROS2

ROS 2 (Robot Operating System 2) [1] は、自律移動ロボットをはじめとする様々なロボットシステムを構築するためのミドルウェアである。近年、ロボットは研究用途にとどまらず、幅広い商用分野や実環境での利用が進んでおり、高い信頼性や拡張性が求められている。

従来の ROS 1 は、モジュラーなフレームワークと豊富なオープンソースコンポーネントによってロボット研究の発展に大きく貢献してきたが、実運用を想定した通信の信頼性やリアルタイム性などの点で課題があった。ROS 2 はこれらの課題を解決するために設計され、分散システムとしての拡張性や信頼性を重視したアーキテクチャを備えている。

このように ROS 2 は、センシング、経路計画、移動制御、自律機能といったロボットに共通する要素を統合的に扱うことができ、研究から実運用まで幅広い用途に対応可能なロボット開発基盤である。

2.2 Navigation2

Navigation2 (Nav2) [2] は、自動運転車向けに開発されたナビゲーション技術を、移動ロボットおよび地上ロボット向けに導入・最適化・再構築した、ROS におけるナビゲーションスタックの後継フレームワークである。Nav2 を用いることで、移動ロボットは複雑な環境内を自律的に移動し、さまざまな運動学モデルを持つロボットに対してユーザー定義のタスクを実行することが可能となる。

Nav2は、単にロボットを地点Aから地点Bへ移動させるだけでなく、中間地点を含む経路の実行や、物体追跡、領域全体を網羅するカバレッジナビゲーションなど、多様なナビゲーションタスクを表現・実行できる。また、知覚、経路計画、制御、自己位置推定、可視化といった、自律移動に必要な機能を統合的に提供することで、高い信頼性を持つ自律移動システムの構築を可能にしている。

Nav2では、センサ情報や環境モデルを基に動的な経路計画を行い、障害物を回避しながらモータの速度指令を生成することで、状況に応じた柔軟な移動を実現する。さらに、複数の独立したモジュラー構成のサーバをビヘイビアツリー（Behavior Tree）によって統合・制御することで、経路計算や動作制御などの各機能を組み合わせた柔軟なナビゲーション動作を実現している。この構成により、ロボットは複雑かつ多様な自律移動タスクを実行することが可能となる。Fig. 2.1にNavigation2の構造を示す

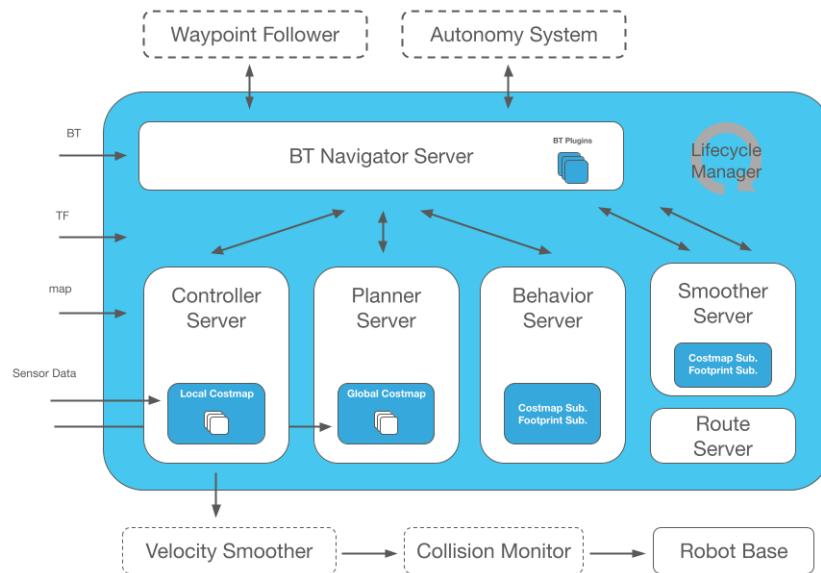


Fig. 2.1 Structure of Navigation2

2.3 emcl2

emcl2 (Extended Monte Carlo Localization 2) [3]は、ROS2環境において利用可能な自己位置推定手法の一つであり、パーティクルフィルタに基づくMonte Carlo Localization (MCL)を拡張したローカライゼーションアルゴリズムである。emcl2は、従来のAMCLと

同様に，事前に与えられた地図とセンサ情報を用いてロボットの自己位置を推定するが，より高いロバスト性と拡張性を持つ点に特徴がある。

emcl2 では，複数のパーティクルを用いてロボットの位置姿勢の確率分布を表現し，オドメトリ情報による状態遷移と，LiDAR などの距離センサを用いた観測モデルに基づいて，各パーティクルの尤度を評価する。これらの尤度に基づいて再サンプリングを行うことで，ロボットの自己位置を逐次的に推定する。また，emcl2 はセンサの外乱や環境変化に対しても安定した推定が行えるように設計されている。

第3章

パラメータ概要

3.1 実験で使うパラメータ

次に実験で使うパラメータを紹介する。下の表に示しているのが今回研究に使われたパラメータである

Table 3.1 Parameter List in emcl2

Parameter Name
num_particles
odom_fw_dev_per_fw
odom_fw_dev_per_rot
odom_rot_dev_per_fw
odom_rot_dev_per_rot
laser_likelihood_max_dist
range_threshold

Table 3.2 Parameter List of the Nav2 Controller

Parameter Name
max_vel_x
max_vel_theta
max_speed_xy
min_theta_velocity_threshold
acc_lim_x
acc_lim_theta

Table 3.3 Parameters of the Nav2 Costmap

Parameter Name
global_resolution
global_cost_scaling_factor
global_inflation_radius
local_resolution
local_cost_scaling_factor
local_inflation_radius

Table 3.4 Parameters of the Nav2 Velocity Smoother

Parameter Name
smoothing_frequency
max_velocity
max_accel_x
max_accel_theta

Table 3.5 Parameters of the Nav2 Planner and Goal Checker

Parameter Name
linear_granularity
angular_granularity
xy_goal_tolerance
trans_stopped_velocity
planner_tolerance

3.2 パラメータの詳細

3.2.1 emcl2

num_particles

本パラメータは、自己位置推定に用いるパーティクルの数を設定するものである。パーティクル数を増加させることで、自己位置分布をより詳細に表現でき、推定精度やロバスト性の向上が期待できる。一方で、計算負荷が増加するため、リアルタイム性とのトレードオフが存在する。

odom_fw_dev_per_fw

前進移動時における前進方向のオドメトリ誤差を表すパラメータである。値を大きくするとオドメトリを信頼しにくくなり、パーティクルの分散が大きくなる。

odom_fw_dev_per_rot

回転動作時における前進方向のオドメトリ誤差を表すパラメータである。値を大きくすると、回転時の前進誤差を大きく見積もるため、自己位置推定の不確実性が増加する。回転動作の多い環境では、推定安定性に影響を与える。

odom_rot_dev_per_fw

前進移動量に対する回転方向のオドメトリ誤差を表すパラメータである。値を大きく設定すると、直進時の姿勢角誤差を大きく考慮するようになる。これにより、直進走行中の姿勢推定のばらつきが増加する。

odom_rot_dev_per_rot

回転動作時における回転方向のオドメトリ誤差を表す。値を大きくすると回転量の信頼度が低下し、姿勢角の推定誤差が大きくなる。旋回動作の安定性に影響を与える重要なパラメータである。

laser_likelihood_max_dist

レーザスキャンを用いた尤度計算において考慮する最大距離を設定するパラメータである。値を大きくすると遠方の障害物まで尤度計算に含まれるが、計算誤差の影響を受けやすくなる。屋外環境では環境特性に応じた調整が必要である。

range_threshold

レーザスキャンデータの有効距離の上限を設定するパラメータである。この値を超える測距データは無効として扱われる。外乱やノイズの多い環境では、自己位置推定の安定化に寄与する。

3.2.2 Navigation2

controller**max_vel_x**

ロボットの前進方向の最大速度を設定するパラメータである。値を大きくすると高速移動が可能となるが、制御の不安定化や振動が生じやすくなる。安全性と走行性能のバランスが重要となる。

max_vel_theta

ロボットの最大回転速度を設定するパラメータである。値を大きくすると旋回が速くなるが、急激な姿勢変化が発生する可能性がある。狭い環境では慎重な調整が必要である。

max_speed_xy

平面移動における最大速度を制限するパラメータである。前進速度と横方向速度を含めた総合的な移動速度の上限を定める。移動の滑らかさに影響を与える。

min_theta_velocity_threshold

回転速度が停止と判定される最小閾値を表すパラメータである。値を大きくすると微小な回転が無視されやすくなり、姿勢制御が粗くなる。ゴール付近での姿勢安定性に影響を与える。

acc_lim_x

前進方向の加速度制限を設定するパラメータである。値を小さくすると加減速が緩やかになり、安定した走行が可能となる。一方で応答性は低下する。

acc_lim_theta

回転方向の加速度制限を設定するパラメータである。急激な旋回動作を抑制し、滑らかな姿勢制御を実現するために用いられる。

costmap**global_resolution**

Global Costmap の解像度を設定するパラメータである。値を小さくすると詳細な地図表現が可能となるが、計算負荷が増加する。

global_cost_scaling_factor

障害物からの距離に応じたコストの減衰率を設定するパラメータである。値を大きくすると障害物付近のコストが急激に低下する。

global_inflation_radius

障害物の周囲をどの程度膨張させるかを設定するパラメータである。安全距離の確保に直接影響する。

local_resolution

Local Costmap の解像度を設定するパラメータである。局所的な障害物回避性能に影響を与える。

local_cost_scaling_factor

Local Costmap における障害物コストの減衰率を設定する。回避挙動の敏感さに影響する。

local_inflation_radius

Local Costmap における障害物の膨張半径を設定する。障害物回避時の安全性を左右する。

Velocity Smoother**smoothing_frequency**

速度平滑化処理の更新周波数を設定するパラメータである。値を大きくすると滑らかな速度制御が可能となる。

max_velocity_x

max_velocity の第 1 要素は、ロボットの 前後方向 (x 方向) の最大並進速度を表す。この値を超える速度指令は制限されるため、ロボットの最高走行速度が規定される。値を大きくすると移動速度は向上するが、停止距離や制御安定性への影響が大きくなる。

max_velocity_theta

max_velocity の第 3 要素は、ロボットの 回転方向 (θ 方向) の最大角速度を表す。旋回時の最大回転速度を制限し、急激な回転動作を防ぐ役割を持つ。値を大きくすると旋回応答は向上するが、姿勢の安定性が低下する可能性がある。

max_accel_x

max_accel の第 1 要素は、並進方向の最大加速度を制限するパラメータである。走行時の急発進を抑制する。

max_accel_theta

max_accel の第 3 要素は、回転方向の最大加速度を制限するパラメータである。旋回動作の安定性向上に寄与する。

planner/ゴール判定**linear_granularity**

経路生成時の直線方向の分割間隔を設定するパラメータである。値を小さくすると経路が滑らかになるが、計算量が増加する。

angular_granularity

経路生成時の角度方向の分割間隔を設定する。旋回精度に影響を与える。

xy_goal_tolerance

ロボットがゴールに到達したと判定される位置誤差の許容範囲を表す。値を大きくするとゴール判定が緩くなる。

trans_stopped_velocity

並進速度が停止と判定される閾値を設定するパラメータである。ゴール付近での停止判定に影響を与える。

planner_tolerance

経路計画時に許容されるゴール周辺の誤差範囲を設定するパラメータである。局所的な計画失敗を防ぐ役割を持つ。

第4章

実験

4.1 実験概要

本研究では、本研究室で開発されているロボット ORNE-box3[4] を用いて走行実験を行った。実機ロボットの外観は Fig. 4.2 に示す。ORNE-box3 のセンサ構成については、Fig. 4.3 に示すように、PC として Jetson Orin NX 16GB を搭載しており、3D LiDAR:R-Fans-16、IMU : ADIS16465、Encoder : i-Cart middle を備えている。3D LiDAR は、自己位置推定と障害物検知に使用し、IMU とエンコーダは、emcl2 に対して情報を提供している。

走行ルートは Fig. 4.1 が示すように、津田沼校舎 2 号館前から食堂前に設置されたコーンまでとし、津田沼チャレンジのコースの一部を利用した。このルートは、屋外環境における自律移動性能を評価するための実環境を想定したものである。

実験では、Navigation2 における各種パラメータを個別に変化させた場合のロボットの挙動の変化を調査することを目的とした。各走行実験においては、対象とするパラメータのみを変更し、それ以外のパラメータはすべて一定に保った。

また、パラメータの変更に際しては、変更前の設定値を基準値とし、基準値から増減させた場合のロボットの走行挙動を比較・分析した。これにより、各パラメータがロボットの走行安定性や挙動に与える影響を明確にすることを試みた。



Fig. 4.1 Course map of the Tsudanuma Challenge 2025(source: [5])



Fig. 4.2 ORNE-box3(source: [4])

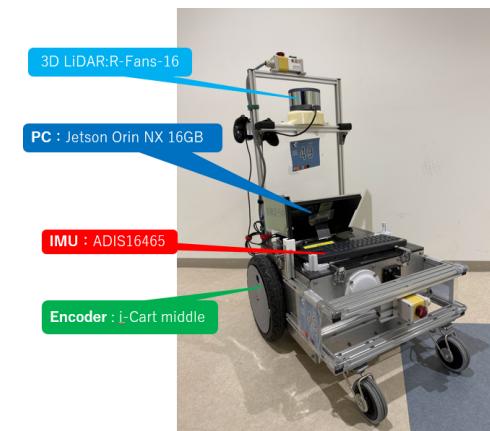


Fig. 4.3 sensor configuration

4.2 実験結果

4.2.1 実験結果 (emcl2)

実験結果 (num_particles)

Table 4.1 パーティクル数の違いによるゴール到達可否

number of times	Number of particles		
	200	500 (base)	1000
1	×		
2			×
3	×	×	×
4			
5			×
6	×	×	×
7			
8			
9			
10			

Table4.1 に、パーティクル数の違いによるゴール到達の可否を示す。

Table 4.2 パーティクル数ごとのゴール到達成功率

パーティクル数	成功率 [%]
200	60
500 (base)	80
1000	60

Table4.2 に、パーティクル数を変化させた際のゴール到達成功率を示す。

Table4.1, Table4.2 が示すように、パーティクル数を 200, 500, 1000 と変化させて自己位

置推定の安定性を評価した。その結果、パーティクル数 500 の場合に最も高いゴール到達率が得られた。一方で、200 および 1000 の場合は、到達率がやや低下する傾向が確認された。パーティクル数が少ないと自己位置推定が破綻しやすく、成功率が低下した。また、パーティクル数が多すぎても計算量の増加によって、遅延が発生して安定性が低下した。以上より、パーティクル数は多ければ良いわけではなく、自己位置推定の精度と計算負荷のバランスを考慮した適切な設定が重要であることが分かる。本実験環境においては、パーティクル数 500 が最も安定した性能を示した。

次からの実験では、Fig. 4.4, Fig. 4.5 のグラフが基準の値の時のロボットの挙動である。Fig. 4.4 はロボットの向き、Fig. 4.5 はロボットの位置のグラフである。

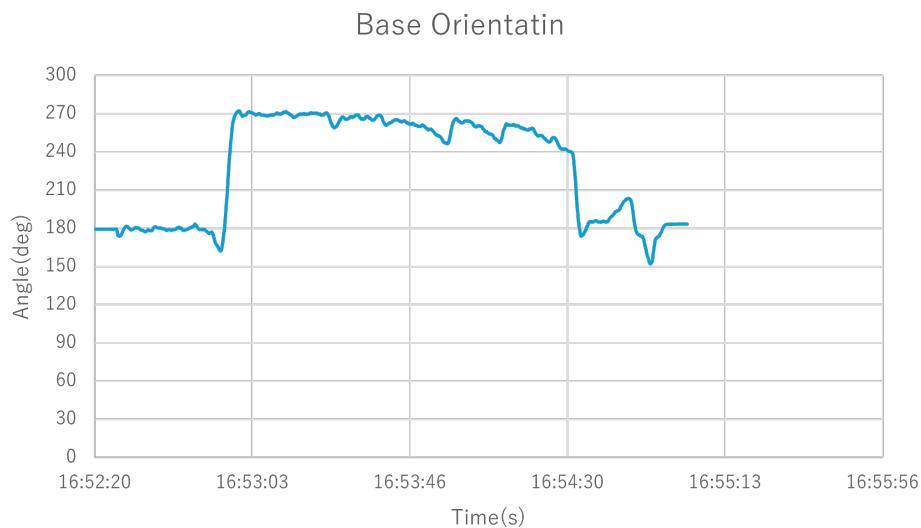


Fig. 4.4 base orientation

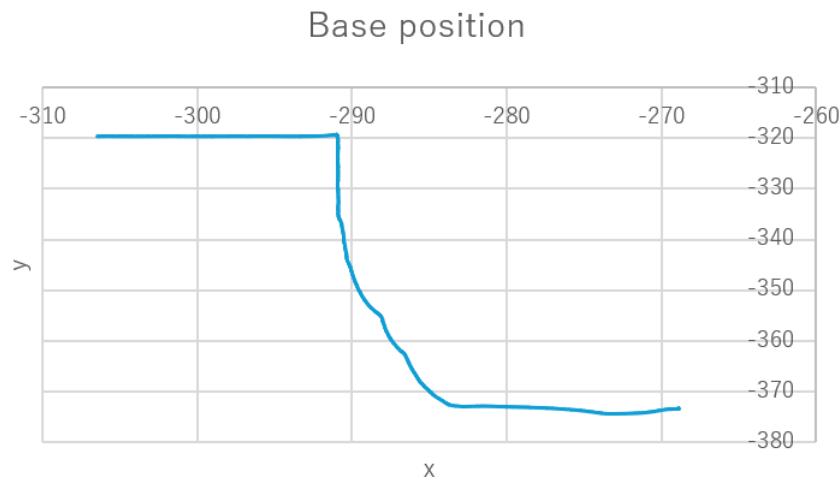


Fig. 4.5 base position

実験結果 (odom_fw_dev_per_fw)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.19



Fig. 4.6 Robot orientation with fw_dev_per_fw = 0.01

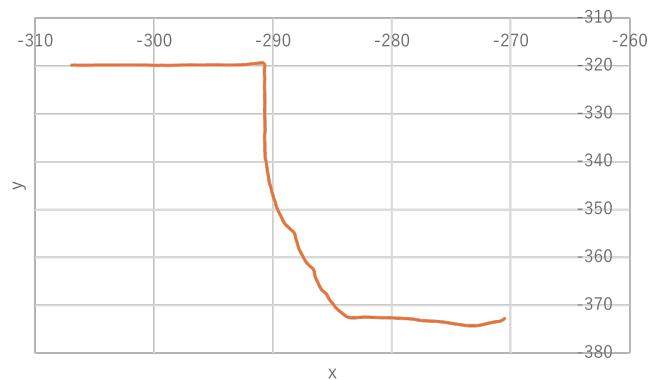


Fig. 4.7 Robot position with $fw_dev_per_fw = 0.01$



Fig. 4.8 Robot orientation with $fw_dev_per_fw = 0.05$

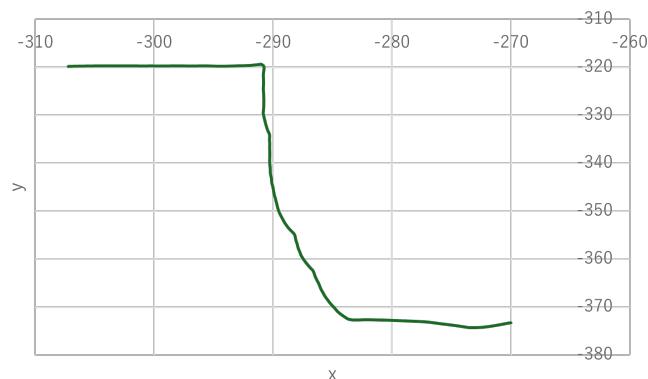


Fig. 4.9 Robot position with $fw_dev_per_fw = 0.05$



Fig. 4.10 Robot orientation with fw_dev_per_fw = 0.1

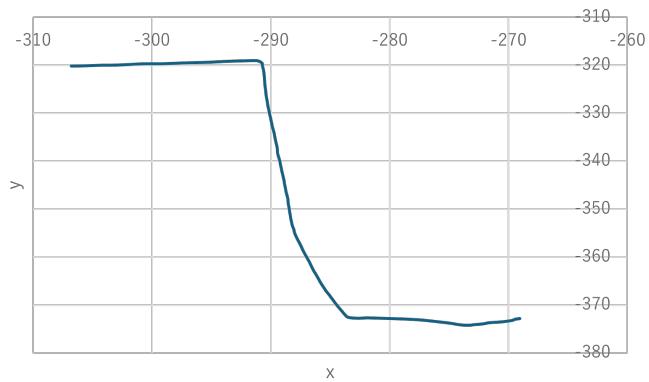


Fig. 4.11 Robot position with fw_dev_per_fw = 0.1



Fig. 4.12 Robot orientation with fw_dev_per_fw = 0.3

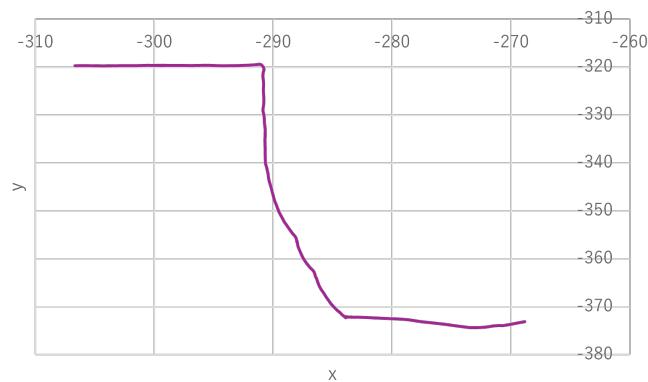
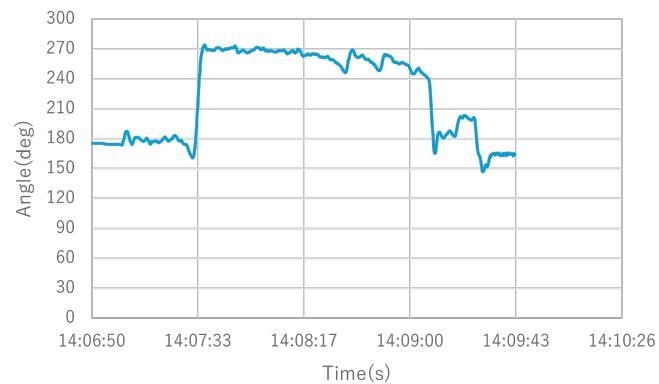
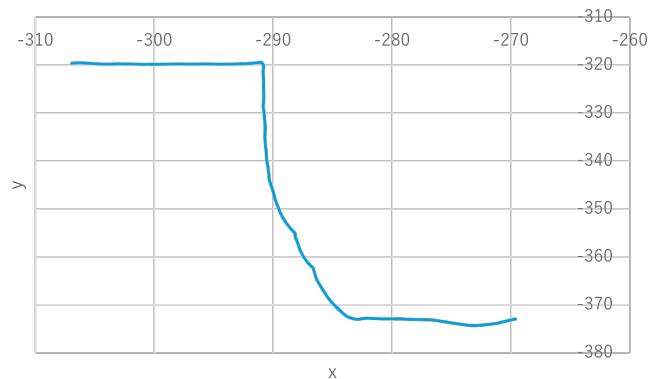
Fig. 4.13 Robot position with $\text{fw_dev_per_fw} = 0.3$ Fig. 4.14 Robot orientation with $\text{fw_dev_per_fw} = 0.4$ Fig. 4.15 Robot position with $\text{fw_dev_per_fw} = 0.4$

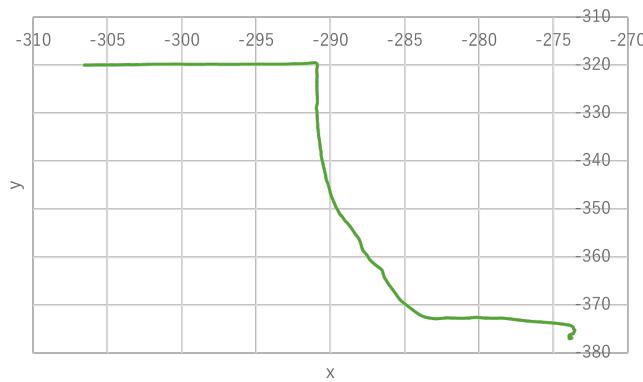
Fig. 4.16 Robot orientation with $fw_dev_per_fw = 0.5$ Fig. 4.17 Robot position with $fw_dev_per_fw = 0.5$

Fig. 4.16 で示すように、`odom_fw_dev_per_fw` を 0.5 に設定した場合、ロボットは走行を続けるにつれて自己位置推定の誤差が累積し、最終的にゴールに到達することができなかった。

一方で、0.4 以下の値に設定した場合には、ロボットはゴールまで到達することができ確認された。

また、直線走行時の挙動を基準条件と比較すると、`odom_fw_dev_per_fw` を大きく設定した場合には、推定されたロボットの位置が徐々にずれしていく様子が確認された。このことから、`odom_fw_dev_per_fw` の値を過度に大きくすると、前進移動に対するオドメトリ誤差が強調され、自己位置推定の精度が低下することが示唆される。

実験結果 (`odom_fw_dev_per_rot`)

Fig. 4.4, Fig. 4.5 で示すように、基準の値は、0.0001



Fig. 4.18 Robot orientation with fw_dev_per_rot = 0.001

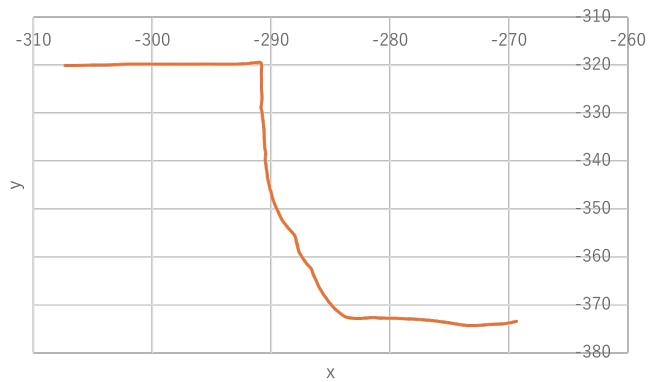


Fig. 4.19 Robot position with fw_dev_per_rot = 0.001



Fig. 4.20 Robot orientation with fw_dev_per_rot = 0.01

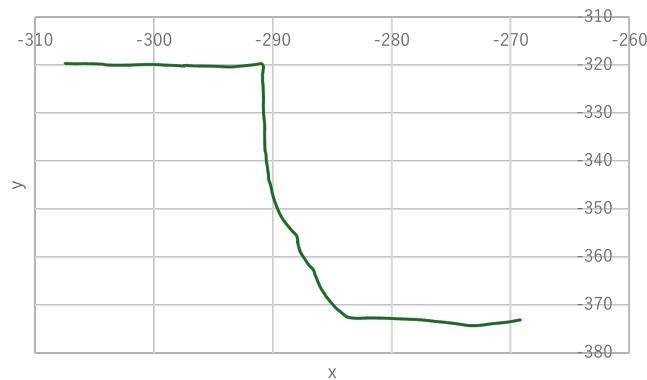


Fig. 4.21 Robot position with $fw_dev_per_rot = 0.01$



Fig. 4.22 Robot orientation with $fw_dev_per_rot = 0.1$

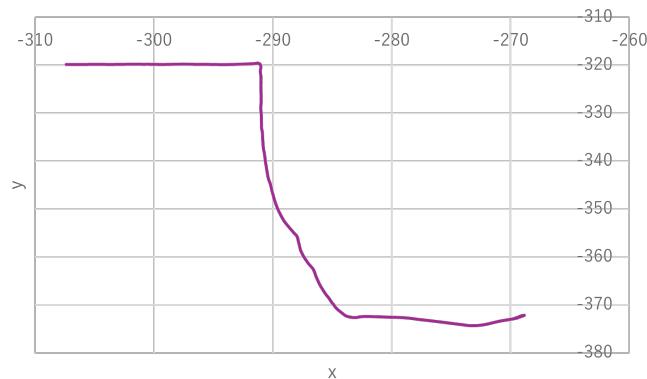


Fig. 4.23 Robot position with $fw_dev_per_rot = 0.1$



Fig. 4.24 Robot orientation with fw_dev_per_rot = 0.5

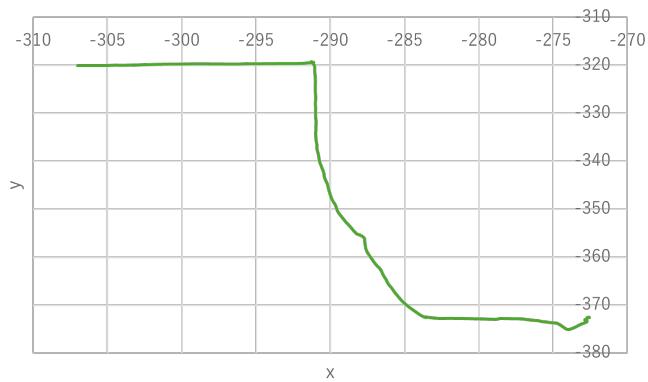


Fig. 4.25 Robot position with fw_dev_per_rot = 0.5

`odom_fw_dev_per_rot` を変化させた実験では、すべての設定値においてロボットはゴールに到達することができた。特に、`odom_fw_dev_per_rot` を 0.5 に設定した場合には、走行中に自己位置推定のずれが一部確認されたものの、走行不能となるような大きな影響は見られず、最終的にゴールまで到達することが確認された。

この結果から、`odom_fw_dev_per_rot` は自己位置推定に一定の影響を与えるものの、fw 方向の移動誤差を表す `odom_fw_dev_per_fw` と比較すると、ゴール到達性に与える影響は相対的に小さいと考えられる。

実験結果 (`odom_rot_dev_per_fw`)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.13



Fig. 4.26 Robot orientation with $\text{rot_dev_per_fw} = 0.01$

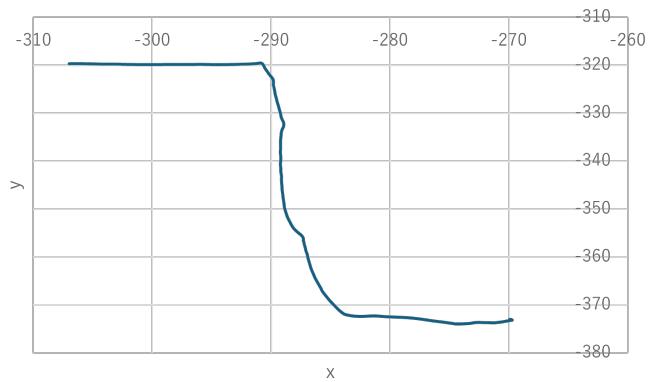


Fig. 4.27 Robot position with $\text{rot_dev_per_fw} = 0.01$



Fig. 4.28 Robot orientation with $\text{rot_dev_per_fw} = 0.2$

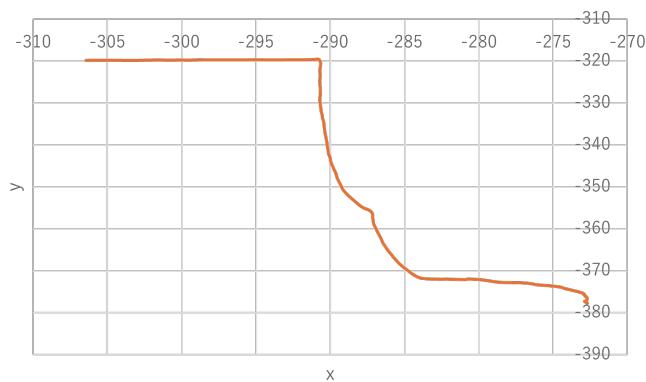


Fig. 4.29 Robot position with $\text{rot_dev_per_fw} = 0.2$



Fig. 4.30 Robot orientation with $\text{rot_dev_per_fw} = 0.3$

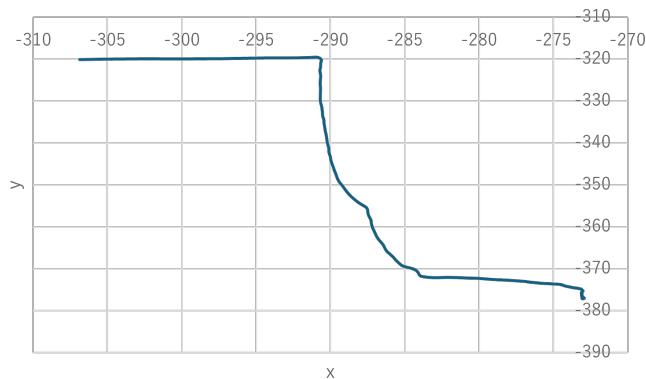


Fig. 4.31 Robot position with $\text{rot_dev_per_fw} = 0.3$



Fig. 4.32 Robot orientation with $\text{rot_dev_per_fw} = 0.4$

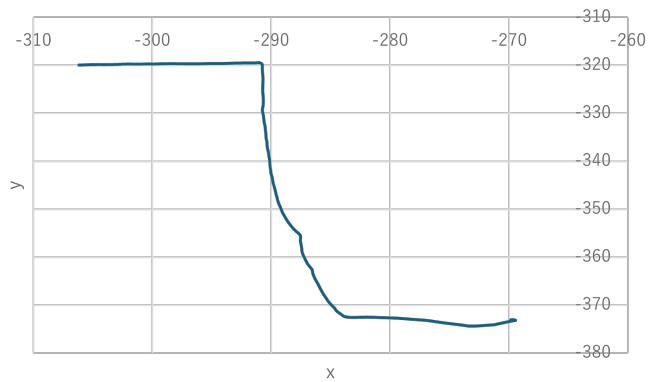


Fig. 4.33 Robot position with $\text{rot_dev_per_fw} = 0.4$



Fig. 4.34 Robot orientation with $\text{rot_dev_per_fw} = 0.5$

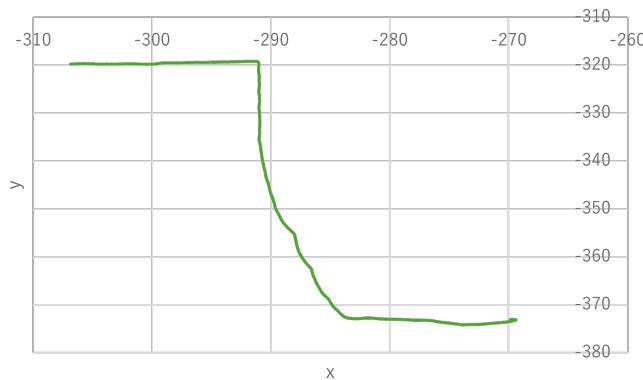


Fig. 4.35 Robot position with $\text{rot_dev_per_fw} = 0.5$

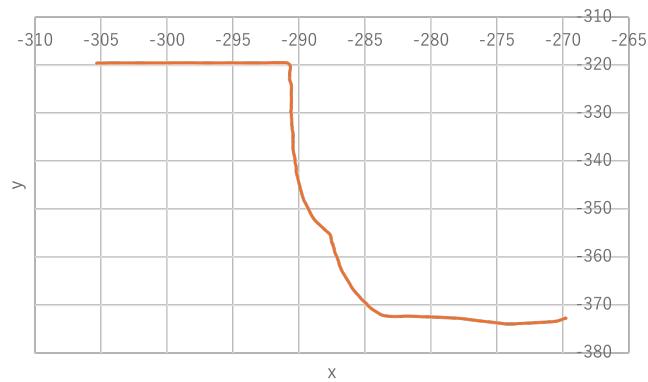
rot_dev_per_fw での実験では、 rot_dev_per_fw の値を大きくした場合においても、ロボットは直進走行中に自己位置および姿勢のわずかなずれが生じるのみであり、走行不能となることはなく、最終的にゴールまで問題なく到達することが確認された。

直線走行区間（目標方位 270° ）でのロボットの向きに着目すると、 rot_dev_per_fw を 0.01 に設定した Fig. 4.26 の場合には、ロボットは目標方位をほぼ維持したまま高い精度で直進していることが確認できた。一方で、 rot_dev_per_fw を 0.4 以上に設定した Fig. 4.32, Fig. 4.34 で示した場合には、走行中のロボットの方位が 270° を下回る傾向が見られ、進行に伴って姿勢が徐々にずれていく様子が確認された。

しかしながら、この方位のずれはナビゲーション全体に致命的な影響を与えるほどではなく、ゴール到達性に大きな影響は見られなかった。

実験結果 ($\text{odom_rot_dev_per_rot}$)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.19

Fig. 4.36 Robot orientation with $\text{rot_dev_per_rot} = 0.1$ Fig. 4.37 Robot position with $\text{rot_dev_per_rot} = 0.1$ Fig. 4.38 Robot orientation with $\text{rot_dev_per_rot} = 0.3$

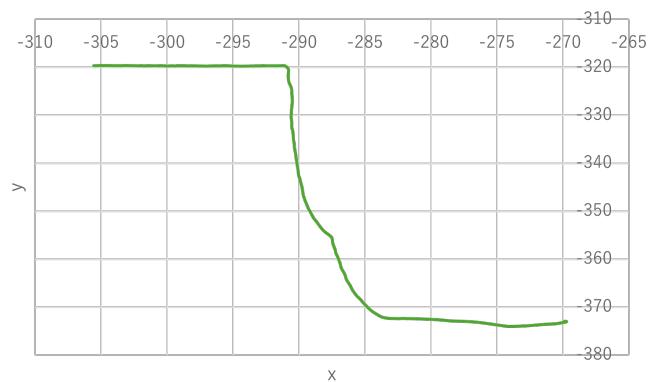


Fig. 4.39 Robot position with $\text{rot_dev_per_rot} = 0.3$



Fig. 4.40 Robot orientation with $\text{rot_dev_per_rot} = 0.4$

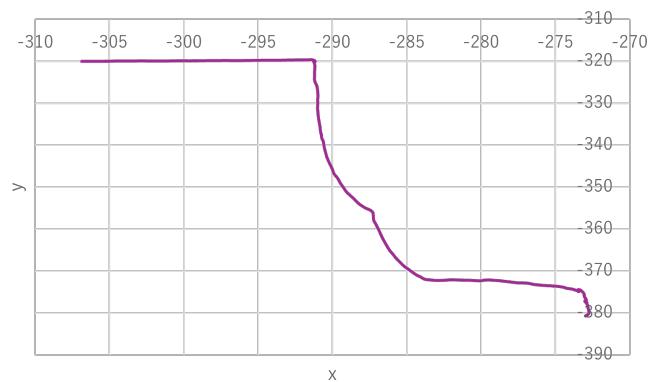
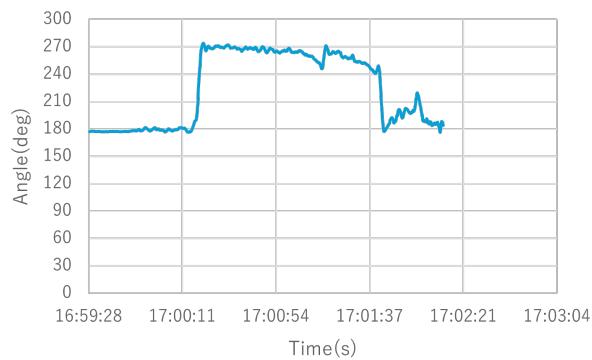
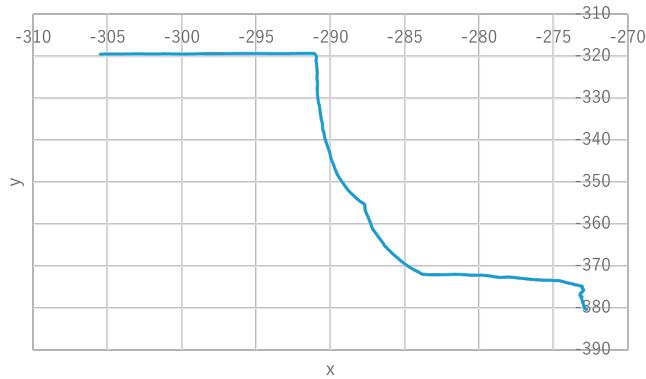


Fig. 4.41 Robot position with $\text{rot_dev_per_rot} = 0.4$

Fig. 4.42 Robot orientation with $\text{rot_dev_per_rot} = 0.5$ Fig. 4.43 Robot position with $\text{rot_dev_per_rot} = 0.5$

rot_dev_per_rot は、emcl において回転量に比例して付加される回転誤差を表すパラメータであり、値が大きいほどセンサ情報を重視し、値が小さいほどオドメトリ情報を重視する設定となる。

本実験の結果、 rot_dev_per_rot の値を大きく設定した場合、自己位置推定においてセンサの影響が強くなり、ロボットの姿勢推定が不安定となることで、走行中の挙動がふらつく傾向が確認された。

特に旋回動作時においてセンサ重視の設定とした場合、ゴール付近で自己位置推定が不安定となり、 rot_dev_per_rot を 0.5 および 0.4 に設定した条件では、ロボットがゴール端付近で停止し、正常にゴールへ到達できない事例が確認された。

以上より、 rot_dev_per_rot を過度に大きく設定すると、回転時の姿勢推定においてセンサノイズの影響を受けやすくなり、ナビゲーション性能が低下する可能性があることが示唆さ

れる。

実験結果 (laser_likelihood_max_dist)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.2



Fig. 4.44 Robot orientation with $\text{laser_likelihood_max_dist} = 0.5$

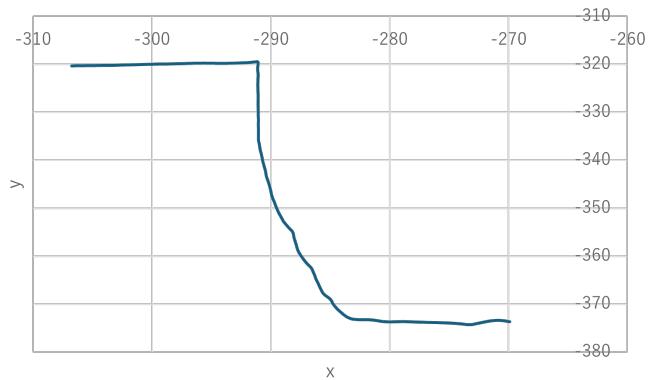
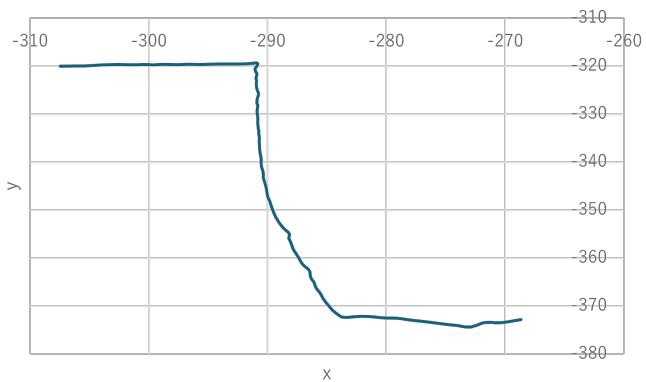


Fig. 4.45 Robot position with $\text{laser_likelihood_max_dist} = 0.5$

Fig. 4.46 Robot orientation with `laser_likelihood_max_dist = 1.0`Fig. 4.47 Robot position with `laser_likelihood_max_dist = 1.0`

本実験では、基準値から `laser_likelihood_max_dist` の値を増加させた場合、レーザによって評価される情報量が増加し、特に障害物や構造物が多く存在する環境において、自己位置推定の更新が頻繁に行われることが確認された。

その結果、レーザスキャンによる評価負荷が高い箇所では、ロボットが一時的に停止する挙動が観測された。これは、自己位置推定の更新に伴い、ナビゲーションが安全側に制御されたためであると考えられる。

一方で、`laser_likelihood_max_dist` を約 0.5 程度まで増加させた場合でも、ロボットは全体として安定した走行が可能であり、大きなナビゲーション性能の低下は確認されなかった。

実験結果 (range_threshold)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.3



Fig. 4.48 Robot orientation with range_threshold = 0.5

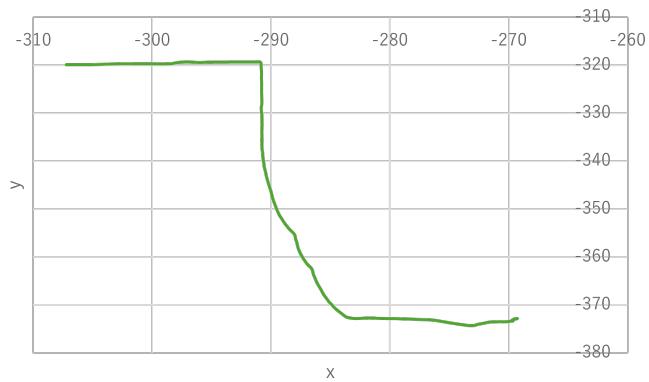


Fig. 4.49 Robot position with range_threshold = 0.5



Fig. 4.50 Robot orientation with range_threshold = 0.9

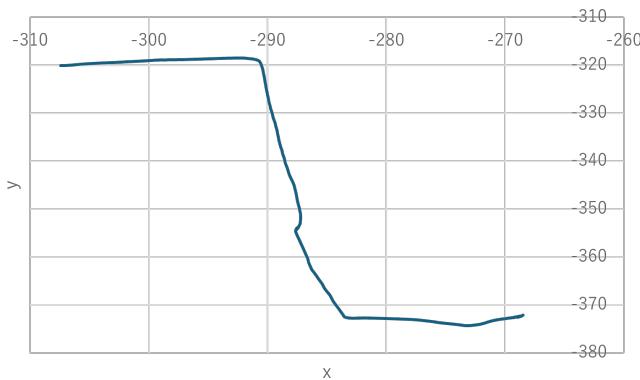


Fig. 4.51 Robot position with range_threshold = 0.9

本実験では、基準値である $\text{range_threshold} = 0.3$ の場合、ロボットは主に比較的近距離のレーザスキャン情報を用いて自己位置推定を行っており、推定された位置および姿勢は安定していた。

一方、 range_threshold を 0.9 に設定した場合、より遠方のレーザスキャン情報も自己位置推定に使用されるため、基準値の場合と比較して、推定されたロボットの位置に明らかなずれが生じることや、姿勢が異なる方向を示す場合が確認された。しかしながら、これらのずれが生じた場合でも、ロボットは最終的にゴール地点へ到達することができた。

また、 range_threshold を 0.5 に設定した場合には、自己位置推定および走行挙動は安定しており、基準値と比較して大きな性能低下は確認されなかった。

以上の結果より、 range_threshold は過度に大きな値を設定すると自己位置推定の精度低下を招く可能性がある一方で、0.5 程度までであれば、安定したナビゲーションを維持したまま設定値を引き上げることが可能であると考えられる。

以上の結果より、emcl2 における各パラメータは、オドメトリ情報とレーザセンサ情報の信頼度のバランスを調整する役割を持ち、設定値によって自己位置推定の特性およびロボットの走行挙動が大きく変化することが明らかとなった。そのため、実環境におけるナビゲーションでは、各パラメータの意味を理解した上で、環境特性に応じた適切な調整が重要であるといえる。

4.2.2 実験結果 (Nav2_controller)

実験結果 (max_vel_x)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.6

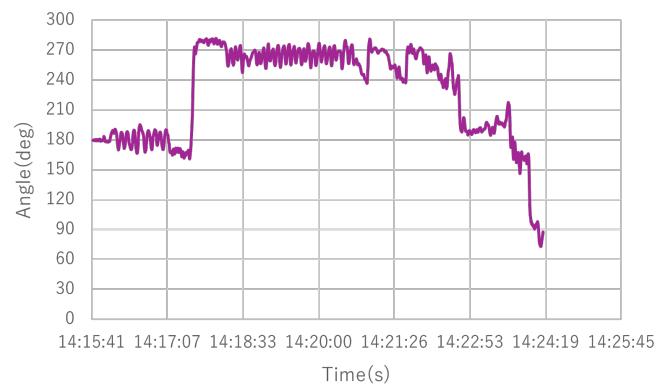


Fig. 4.52 Robot orientation with $\text{max_vel_x} = 0.2$

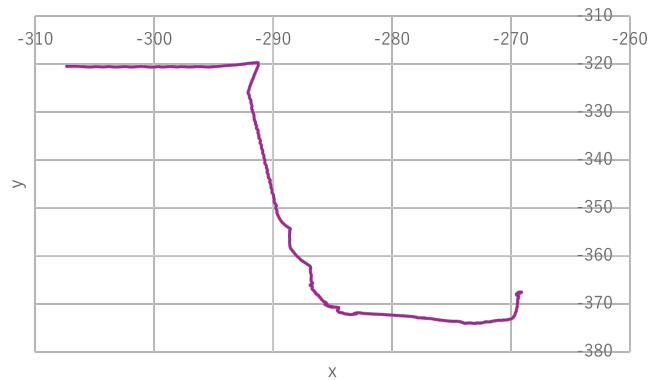
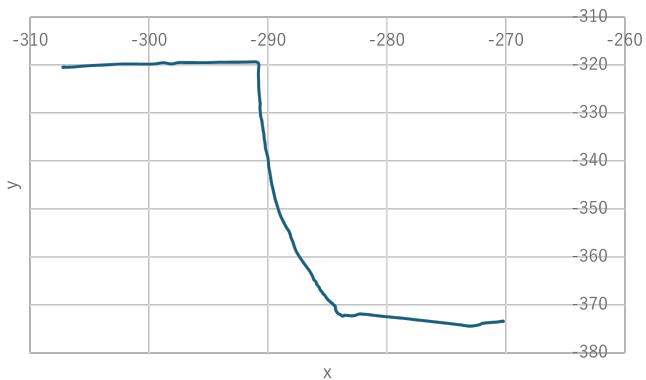


Fig. 4.53 Robot position with $\text{max_vel_x} = 0.2$

Fig. 4.54 Robot orientation with $\text{max_vel_x} = 1.0$ Fig. 4.55 Robot position with $\text{max_vel_x} = 1.0$

まず、 max_vel_x の値を小さく設定した場合、ロボットの走行速度が低下し、走行中の挙動が不安定になる傾向が確認された。しかしながら、速度が低下した状態であっても、ロボットはゴールまで到達することができ、ナビゲーション自体が破綻することはなかった。

一方で、 max_vel_x の値を基準値より大きく設定した場合、走行挙動に大きな変化は見られず、ロボットの実際の走行速度も基準値である 0.6 [m/s] 以上には上昇しなかった。この結果から、 max_vel_x を増加させても、他のパラメータやシステム制約によってロボットの最大速度が制限されている可能性が考えられる。

以上のことから、 max_vel_x は走行速度の上限を設定する重要なパラメータである一方で、ロボットの実際の最高速度は、Controller 以外のパラメータやハードウェア設定の影響も受けることが示唆された。この速度制限の要因については、後述する実験において詳細に検証

する。

実験結果 (max_vel_theta)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.7



Fig. 4.56 Robot orientation with $\text{max_vel_theta} = 0.4$

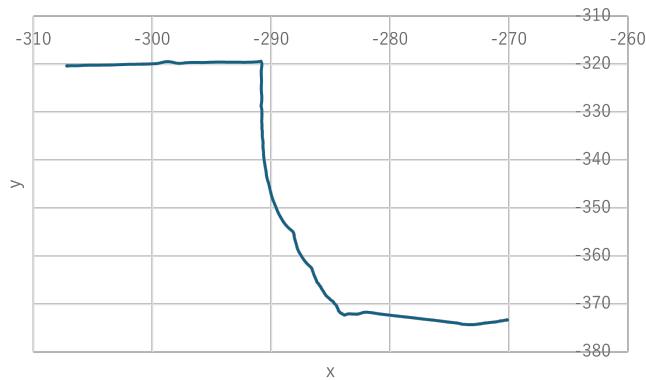
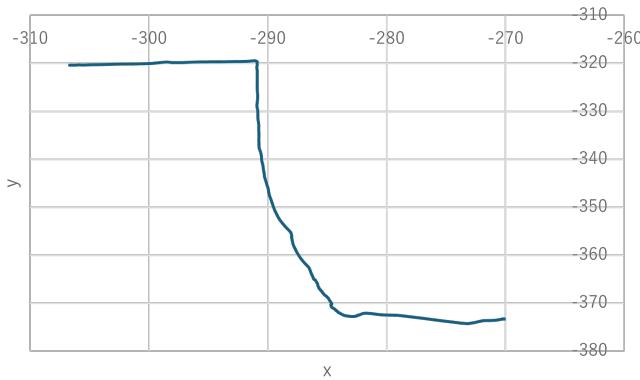


Fig. 4.57 Robot position with $\text{max_vel_theta} = 0.4$

Fig. 4.58 Robot orientation with $\text{max_vel_theta} = 1.0$ Fig. 4.59 Robot position with $\text{max_vel_theta} = 1.0$

本実験で、 max_vel_theta の値を基準値より大きく設定した場合、ロボットの走行挙動に顕著な変化は見られなかった。この結果から、基準値の時点でロボットの旋回性能は十分に確保されており、それ以上角速度の上限を引き上げても、実際の挙動には大きく反映されないことが示唆される。

一方で、 max_vel_theta の値を基準値より小さく設定した場合、ロボットの挙動が不安定になる様子が確認された。特に、旋回が必要となる箇所において、ロボットが一時的に停止し、旋回完了までに時間を要する場面が多く見られた。さらに、旋回後の直進動作においても、進行方向が安定せず、ふらついた挙動を示す傾向が確認された。

以上の結果より、 max_vel_theta を過度に小さく設定すると、旋回動作およびその後の直進動作に悪影響を及ぼし、ナビゲーション全体の安定性が低下することが明らかとなった。その

ため，本パラメータは，旋回性能を確保できる十分な値に設定することが重要である．

実験結果 (max_speed_xy)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.9

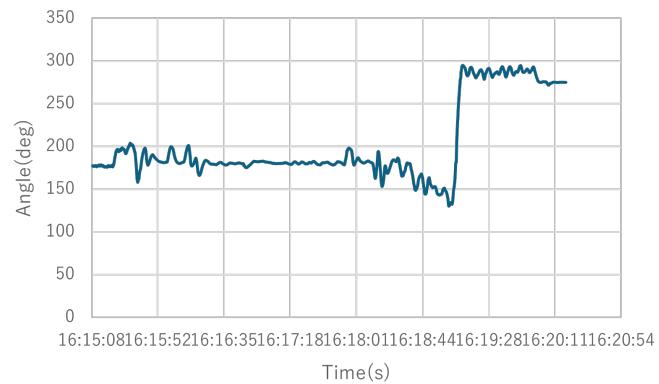


Fig. 4.60 Robot orientation with $\text{max_speed_xy} = 0.1$

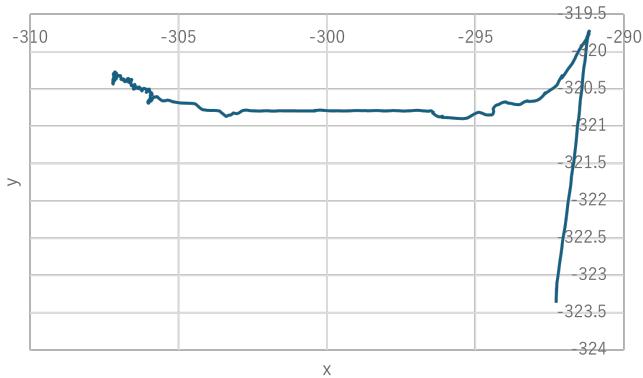


Fig. 4.61 Robot position with $\text{max_speed_xy} = 0.1$



Fig. 4.62 Robot orientation with $\text{max_speed_xy} = 0.5$

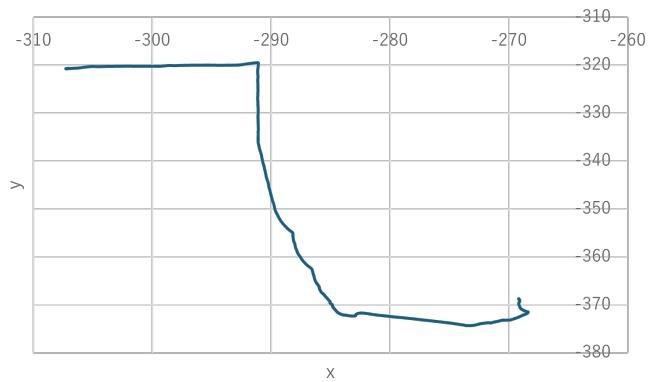


Fig. 4.63 Robot position with $\text{max_speed_xy} = 0.5$



Fig. 4.64 Robot orientation with $\text{max_speed_xy} = 1.5$

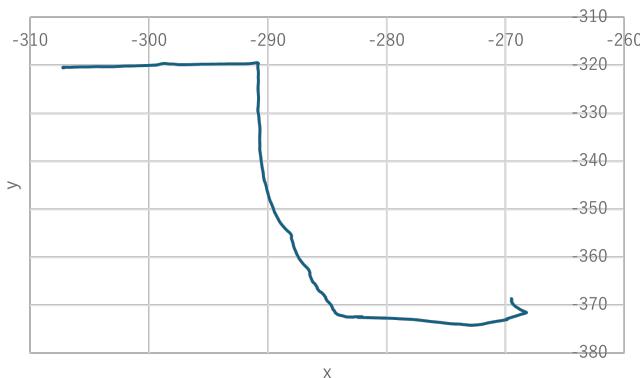


Fig. 4.65 Robot position with $\text{max_speed_xy} = 1.5$

本実験では、 max_speed_xy を 0.5 に設定した場合、ロボットは安定した走行を維持したままゴールへ到達することができた。また、 max_speed_xy を 1.5 に設定した場合においても、ロボットの実際の走行速度に大きな変化は見られなかった。この挙動は、前節で述べた max_vel_x の実験結果と同様であり、他のパラメータやシステム制約によってロボットの最大速度が制限されている可能性が考えられる。この点については、後述する実験において検証を行う。

一方で、 max_speed_xy を 0.1 に設定した場合、ロボットの走行速度が極端に低下し、ゴールまで到達することができなかった。特に、走行開始後の初期段階において、最初の曲がり角付近の段差に接触する挙動が確認された。また、この設定値では、曲がり角に進入する際に左右の車輪を交互に動かすような挙動が見られ、滑らかな旋回動作が行われていなかった。

以上の結果より、 max_speed_xy を過度に小さく設定すると、旋回動作が不安定となり、ナビゲーションの遂行が困難になることが明らかとなった。一方で、本パラメータを大きく設定した場合でも、他の速度制限要因によって実際の走行速度が制約される可能性があるため、他の速度関連パラメータとの関係を考慮した調整が重要である。

実験結果 ($\text{min_theta_velocity_threshold}$)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.001



Fig. 4.66 Robot orientation with `min_theta_velocity_threshold = 0.01`

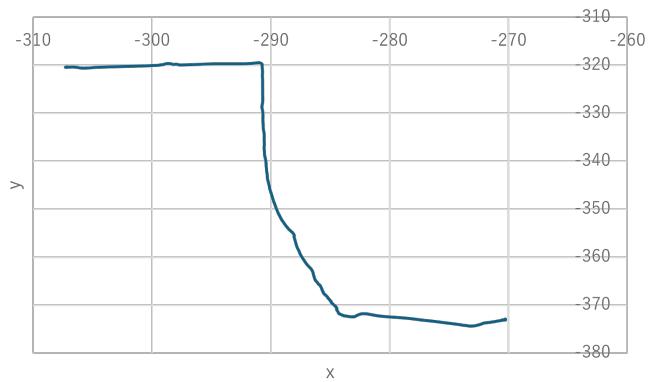


Fig. 4.67 Robot position with `min_theta_velocity_threshold = 0.01`



Fig. 4.68 Robot orientation with `min_theta_velocity_threshold = 0.1`

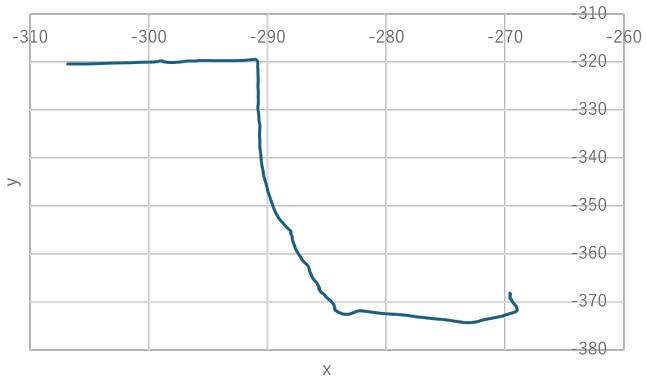


Fig. 4.69 Robot position with `min_theta_velocity_threshold = 0.1`



Fig. 4.70 Robot orientation with `min_theta_velocity_threshold = 1.0`

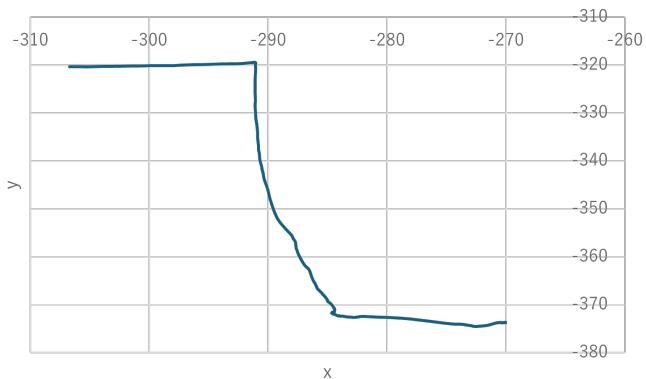


Fig. 4.71 Robot position with `min_theta_velocity_threshold = 1.0`

本実験では、基準値である `min_theta_velocity_threshold = 0.001` に対し、本実験では 0.01，

0.1, 1.0 の各値について走行実験を行った。その結果、基準値から値を段階的に増加させた場合においても、ロボットの走行挙動に大きな変化は確認されなかった。

特に、`min_theta_velocity_threshold` を 1.0 に設定した場合、旋回動作時にロボットが停止する挙動が生じることが予想されたが、実際には旋回および直進動作ともに問題なく行われ、ロボットはゴール地点まで到達することができた。

以上の結果より、`min_theta_velocity_threshold` は、今回の実験環境および走行条件においては、一定範囲内で値を変化させてもロボットの挙動に与える影響が比較的小さいパラメータであることが示唆された。

実験結果 (`acc_lim_x`)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、1.0



Fig. 4.72 Robot orientation with `acc_lim_x` = 0.1

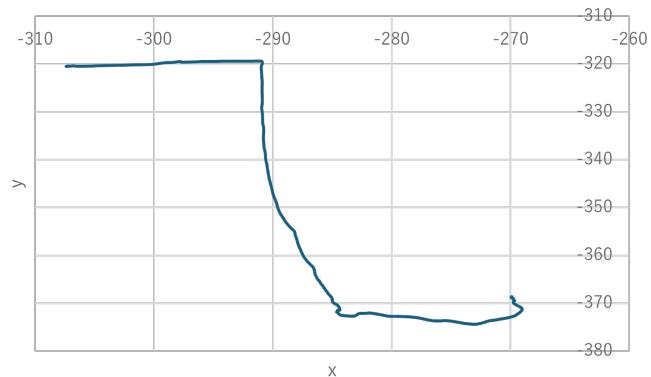
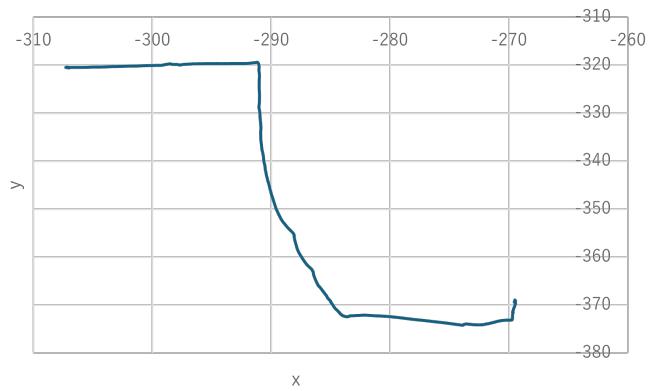
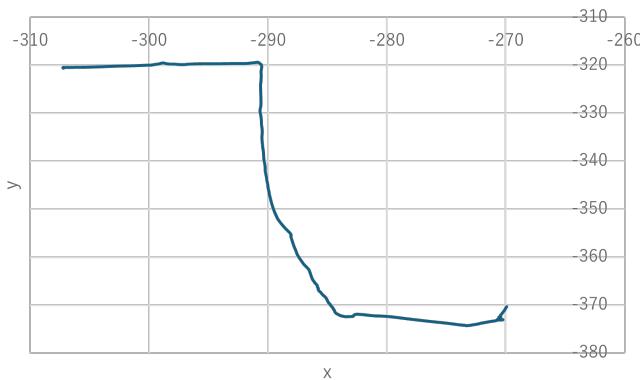


Fig. 4.73 Robot position with `acc_lim_x` = 0.1

Fig. 4.74 Robot orientation with $\text{acc_lim_x} = 0.5$ Fig. 4.75 Robot position with $\text{acc_lim_x} = 0.5$ Fig. 4.76 Robot orientation with $\text{acc_lim_x} = 1.5$

Fig. 4.77 Robot position with $\text{acc_lim_x} = 1.5$

本実験では、 acc_lim_x の値を変化させた場合、曲がり角通過後の直進動作において、ロボットの加速がやや速く感じられる場面が確認された。しかしながら、それ以外の走行挙動においては顕著な変化は見られず、いずれの設定値においてもロボットは安定した走行を維持したままゴール地点へ到達することができた。

以上の結果より、 acc_lim_x はロボットの加速特性を決定する上で必要なパラメータであるものの、本実験条件においては、走行経路の追従性能やナビゲーション全体の安定性に与える影響は比較的小さいことが示唆された。

実験結果 (acc_lim_theta)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、3.2

Fig. 4.78 Robot orientation with $\text{acc_lim_theta} = 0.1$

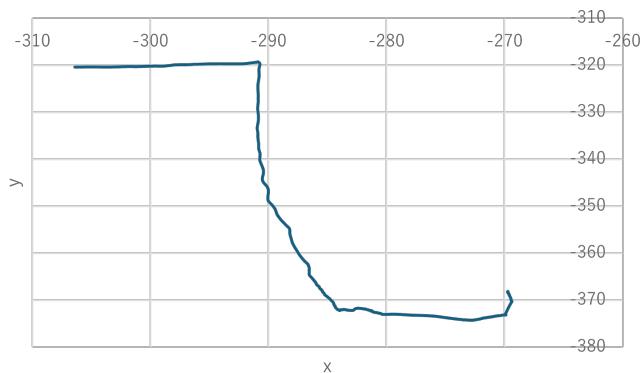


Fig. 4.79 Robot position with $\text{acc_lim_theta} = 0.1$



Fig. 4.80 Robot orientation with $\text{acc_lim_theta} = 1.0$

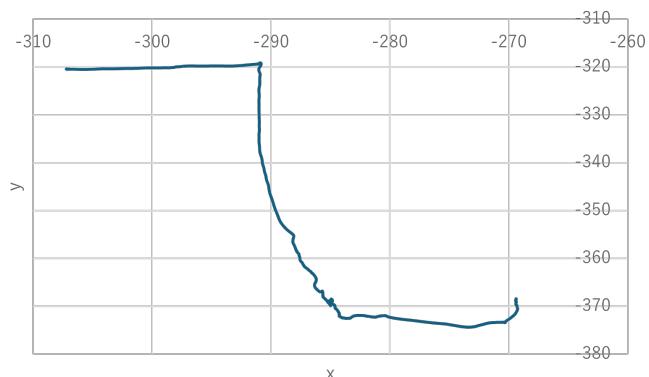
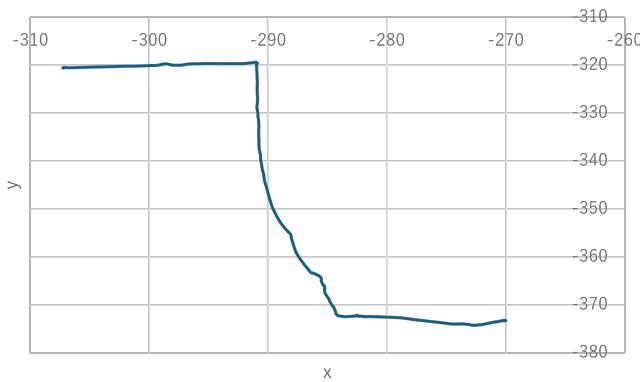


Fig. 4.81 Robot position with $\text{acc_lim_theta} = 1.0$

Fig. 4.82 Robot orientation with $\text{acc_lim_theta} = 4.0$ Fig. 4.83 Robot position with $\text{acc_lim_theta} = 4.0$

本実験では、 acc_lim_theta を 0.1 に設定した場合、最終的な旋回動作後の直進区間において、ロボットの進行方向が細かく変化する、いわゆるくねくねとした挙動が確認された。その後、姿勢が安定し、最終的にはゴール地点へ到達することができた。

また、 acc_lim_theta を 1 に設定した場合においても、走行中の進行方向が細かく変化する挙動が確認され、これは取得したグラフからも明確に観察された。しかしながら、この設定値においても、ナビゲーション自体に支障はなく、ロボットは問題なくゴールへ到達した。

一方で、 acc_lim_theta を 4 に設定した場合、走行挙動に大きな変化は見られず、旋回および直進動作ともに安定した走行が確認された。

以上の結果より、 acc_lim_theta を過度に小さく設定すると、旋回後の姿勢制御が不安定になりやすい一方で、一定以上の値に設定することで、安定した旋回挙動を維持できることが示

唆された。

以上の結果より、Controller に関するパラメータは、ロボットの走行速度や旋回特性といった運動特性を調整する上で重要な役割を果たす一方で、一部のパラメータについては、他の設定値やシステム制約との相互作用によって挙動への影響が限定されることが分かった。そのため、Controller パラメータの調整においては、単一のパラメータのみを変更するのではなく、関連するパラメータとの関係を考慮しながら総合的に調整を行うことが重要である。

4.2.3 実験結果 (Nav2_Costmap)

実験結果 (global_resolution)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.1

0.05, 0.5, 1.0,

本実験では、global_resolution の値を 0.05, 0.5, 1.0 に変化させて走行実験を行ったが、いずれの設定においても、ロボットの挙動に大きな変化は見られなかった。また、経路生成や走行中の挙動は安定しており、ゴールまで問題なく到達することが確認された。

この結果から、今回使用した環境およびタスク条件においては、global_resolution を変更しても、経路計画やロボットの走行挙動への影響は小さいと考えられる。そのため、本実験環境では、過度に解像度を高く設定する必要はなく、計算負荷とのバランスを考慮した標準的な値を用いることが有効であるといえる。

実験結果 (global_cost_scaling_factor)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、3.0

1.0, 5.0, 10

本実験では、基準値に対して値を下げる場合(1.0)および、値を上げた場合(5, 10)について走行実験を行った。その結果、いずれの設定においてもロボットは無事にゴールへ到達し、ナビゲーション全体や走行中の基本的な挙動に大きな差異は見られなかった。

一方で、ゴール地点付近に設置されたコーンを回避する際の挙動に違いが確認された。具体的には、global_cost_scaling_factor の値が大きい場合、障害物に対する回避動作の開始が早く、より余裕を持って障害物を避ける挙動が見られた。これに対し、値が小さい場合は、障害物に比較的近づいてから回避動作を行う傾向が確認された。

以上の結果から，global_cost_scaling_factor は，経路生成の成否やゴール到達自体には大きな影響を与えないものの，障害物回避のタイミングや挙動の余裕度に影響を及ぼすパラメータであると考えられる。そのため，環境中の障害物配置や安全マージンの要求に応じて，適切な値を選択することが重要である。

実験結果 (global_inflation_radius)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.5

0.1, 1.0, 2.0

本実験では，global_inflation_radius の値を基準値から大きくした場合と小さくした場合について走行実験を行った。値を大きく設定した 1.0, 2.0 の場合，建物や障害物の周囲に広い膨張領域が生成され，ロボットがその範囲に進入しないような経路が生成された。その結果，道幅が狭い最初の直線区間において，ロボットが通行可能な領域が制限され，進行開始がやや遅れる挙動が確認された。しかしながら，走行中の挙動は安定しており，最終的には問題なくゴールに到達した。

一方，global_inflation_radius の値を小さく設定した 0.5 の場合においても，ロボットは大きな問題なくゴールへ到達した。障害物との距離は近くなるものの，今回の環境では衝突や著しい不安定挙動は確認されなかった。

以上の結果から，global_inflation_radius は，ゴール到達の可否よりも，狭い通路や細い道を通過する際の経路の取り方やロボットの走行余裕に影響を与えるパラメータであると考えられる。そのため，通路幅が限られた環境や，障害物間を通過する必要がある環境においては，ロボットのサイズや走行特性に応じて適切に調整することが重要である。

実験結果 (local_resolution)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.1

0.05, 0.5, 1.0

本実験では，基準値である 0.1 を中心に，値を小さくした場合および大きくした場合の挙動を比較した。値を 0.05 に設定した場合，ロボットは経路を正確に追従し，全体を通して安定した走行を行い，問題なくゴールに到達した。基準値より高い解像度で環境を認識できていることから，障害物回避や経路追従が適切に行われたと考えられる。

一方，local_resolution を 0.5 に設定した場合，ロボットは長い直進区間において，右側の段差付近を通過する挙動が確認された．走行位置はやや危険な箇所であったものの，最終的にはゴールまで到達することができた．このことから，解像度の低下により環境表現が粗くなり，障害物との距離評価が不十分になっている可能性が考えられる．

さらに，local_resolution を 1.0 に設定した場合，ロボットは走行開始直後にその場でゆっくり回転し，すぐには移動を開始しなかった．その後も，停止とわずかな旋回を繰り返す挙動が見られ，最終的には最初の曲がり角付近で停止し，ゴールに到達することができなかった．これは，解像度が極端に低下したこと，周囲環境を正しく認識できず，適切な経路生成や障害物回避が行えなかつたためであると考えられる．

以上の結果から，local_resolution はロボットの走行安定性や障害物回避性能に大きく影響を与えるパラメータであり，値を大きくしそぎると，ナビゲーションの破綻を引き起こす可能性があることが確認された．本実験環境においては，基準値である 0.1 もしくは，それ以下の値を用いることで，安定した自律走行が実現できると考えられる．

実験結果 (local_cost_scaling_factor)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、3

1.0, 5.0, 10

実験結果 (local_inflation_radius)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.3

0.1, 0.5, 1.0, 2.0

4.2.4 実験結果 (Nav2_Velocity Smoother)

実験結果 (smoothing_frequency)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、20

5, 10, 40, 60

実験結果 (max_velocity_x)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.6

0.3,0.9,1.2

実験結果 (max_velocity_theta)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、1.7

0.6,2.5

実験結果 (max_accel_x)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.5

0.2,1.0

実験結果 (max_accel_theta)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、1.0

0.5,2.0

4.2.5 実験結果 (Nav2_planner goal)

実験結果 (linear_granularity)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.05



Fig. 4.84 Robot orientation with linear_granularity = 0.01

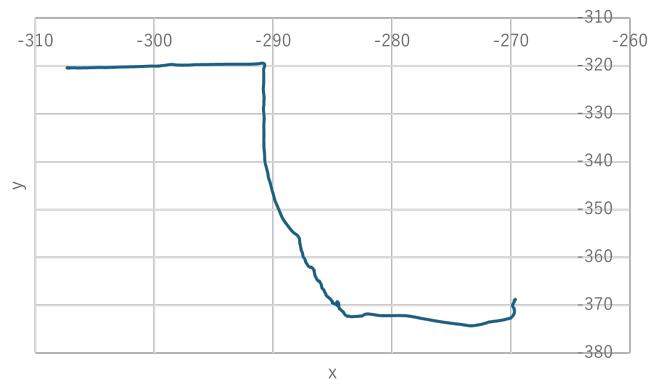


Fig. 4.85 Robot position with linear_granularity = 0.01



Fig. 4.86 Robot orientation with linear_granularity = 0.1

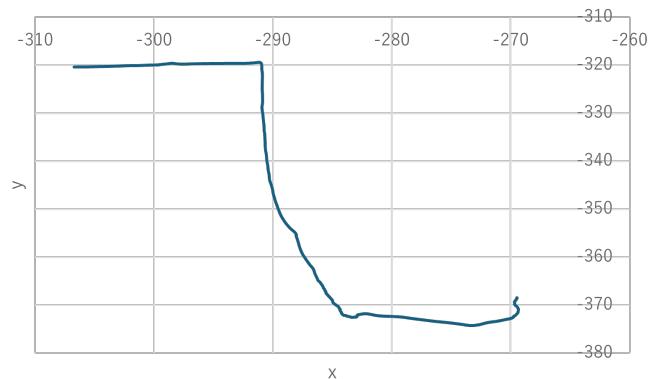


Fig. 4.87 Robot position with linear_granularity = 0.1



Fig. 4.88 Robot orientation with $\text{linear_granularity} = 0.5$

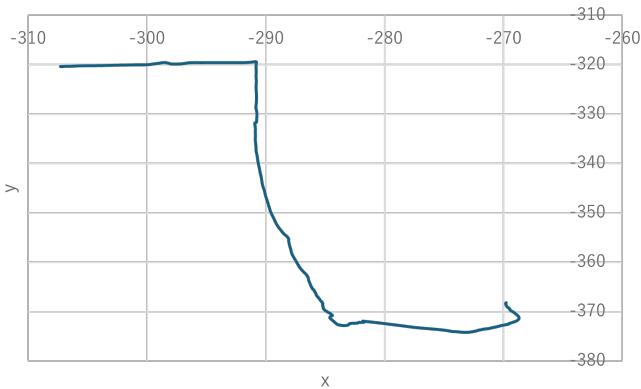


Fig. 4.89 Robot position with $\text{linear_granularity} = 0.5$

本実験では、 $\text{linear_granularity}$ を 0.5 に設定した場合、走行開始後の最初の旋回箇所においてロボットが一時的に停止する挙動が確認され、その後、再び動き出すまでに時間を要した。この挙動は、経路の分解能が粗くなつたことにより、滑らかな経路生成が困難になつたためであると考えられる。

一方で、 $\text{linear_granularity}$ を 0.1 に設定した場合、走行挙動は安定しており、旋回動作を含めた経路においてもスムーズな走行が確認された。この結果から、 $\text{linear_granularity}$ を基準値よりやや大きく設定した場合でも、安定したナビゲーションが可能であることが示唆された。

以上の結果より、 $\text{linear_granularity}$ を過度に大きく設定すると、経路生成の精度低下により走行挙動が不安定になる可能性がある一方で、0.1 程度の設定値であれば、計算負荷と走行

安定性の両立が可能であると考えられる。

実験結果 (angular_granularity)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.025



Fig. 4.90 Robot orientation with angular_granularity = 0.001

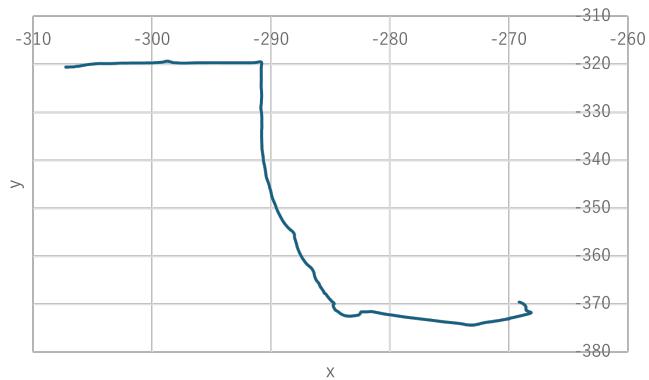


Fig. 4.91 Robot position with angular_granularity = 0.001



Fig. 4.92 Robot orientation with angular_granularity = 0.01

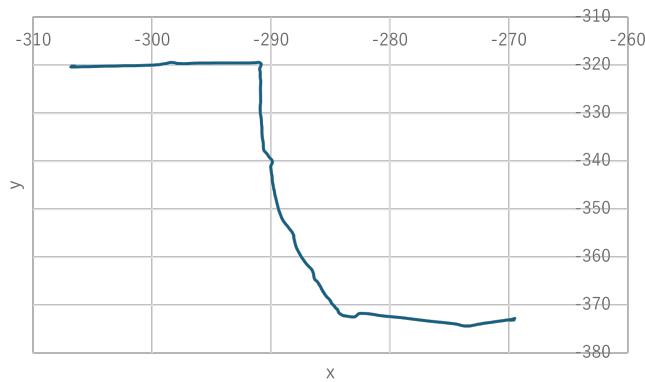


Fig. 4.93 Robot position with angular_granularity = 0.01



Fig. 4.94 Robot orientation with angular_granularity = 0.04

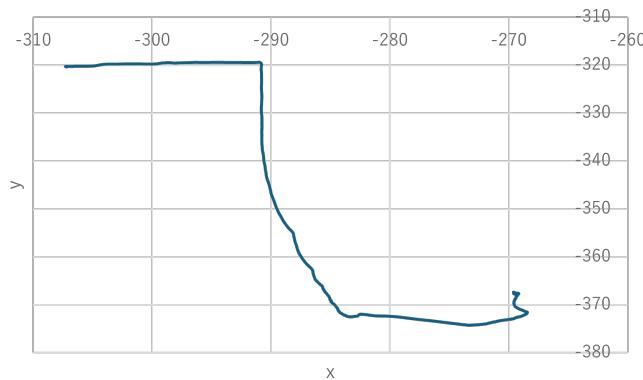


Fig. 4.95 Robot position with angular_granularity = 0.04

本実験では、angular_granularity の値を基準値より小さく設定した場合、ロボットの走行挙動に大きな変化は見られず、旋回および直進動作ともに安定した走行が確認された。この結果から、角度分解能を高めても、本実験環境においては挙動への影響は限定的であることが示唆される。

一方で、angular_granularity を 4.0 に設定した場合、最終的な旋回区間において、ロボットの進行方向にわずかな変化が生じていることが確認された。この挙動は、angular_granularity = 4.0 の設定におけるロボットの姿勢変化を示したグラフからも確認でき、角度分解能が粗くなつたことにより、旋回動作の精度が低下した可能性が考えられる。

以上の結果より、angular_granularity は、一定範囲内で値を小さく設定しても走行挙動に大きな影響を与えない一方で、過度に大きな値を設定した場合には、旋回動作時の姿勢精度に影響を及ぼす可能性があることが示された。

実験結果 (xy_goal_tolerance)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.25



Fig. 4.96 Robot orientation with $xy_goal_tolerance = 0.01$

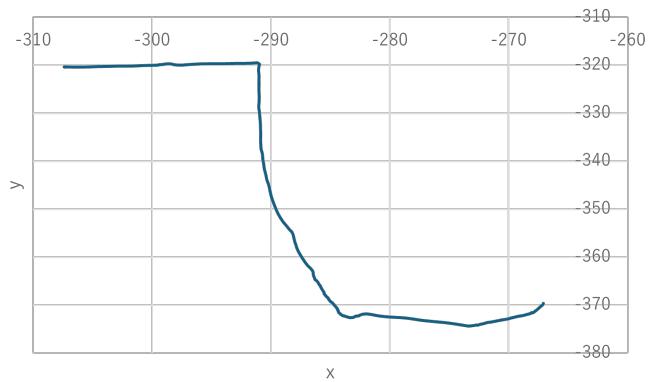
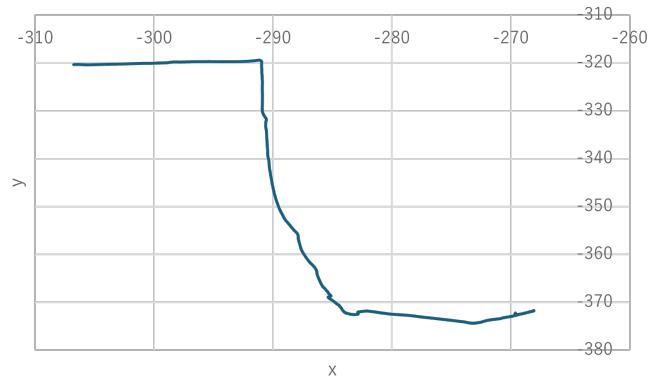
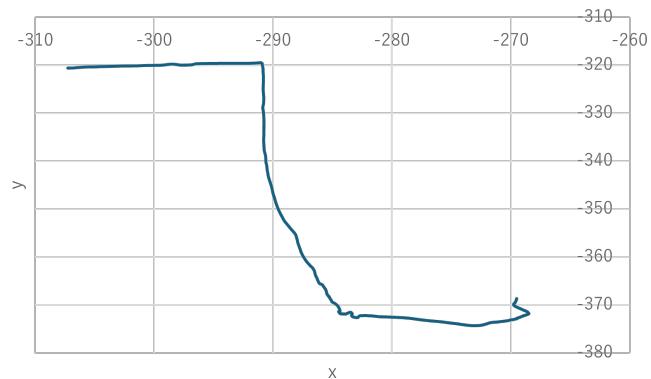


Fig. 4.97 Robot position with $xy_goal_tolerance = 0.01$



Fig. 4.98 Robot orientation with $xy_goal_tolerance = 0.1$

Fig. 4.99 Robot position with $xy_goal_tolerance = 0.1$ Fig. 4.100 Robot orientation with $xy_goal_tolerance = 0.4$ Fig. 4.101 Robot position with $xy_goal_tolerance = 0.4$

本実験では、 $xy_goal_tolerance$ の値を 0.01, 0.1, 0.4 に設定して実験を行った結果、いずれの

値においても、ロボットの走行挙動に顕著な変化は確認されなかった。また、ゴール付近での振動や停止位置の不安定化といった現象も見られず、すべての条件において安定したゴール到達が達成された。

この結果から、本実験環境および使用したロボットモデルにおいては、`xy_goal_tolerance` の設定値がロボットの走行挙動に与える影響は小さいことが示唆される。したがって、本パラメータは、厳密な位置精度よりもゴール到達判定の柔軟性を調整する目的で用いられ、挙動そのものへの影響は限定的であると考えられる。

実験結果 (`trans_stopped_velocity`)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.25



Fig. 4.102 Robot orientation with `trans_stopped_velocity` = 0.01

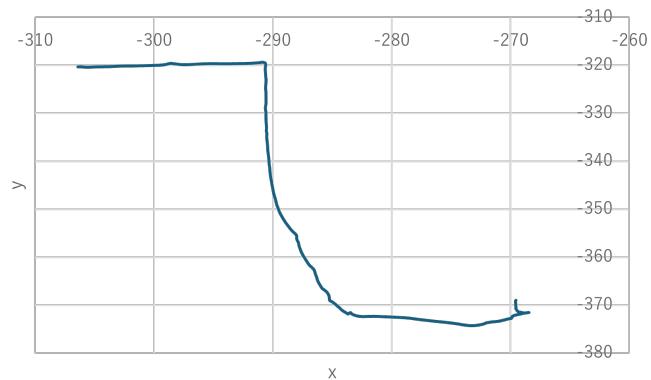


Fig. 4.103 Robot position with `trans_stopped_velocity` = 0.01



Fig. 4.104 Robot orientation with trans_stopped_velocity = 0.1

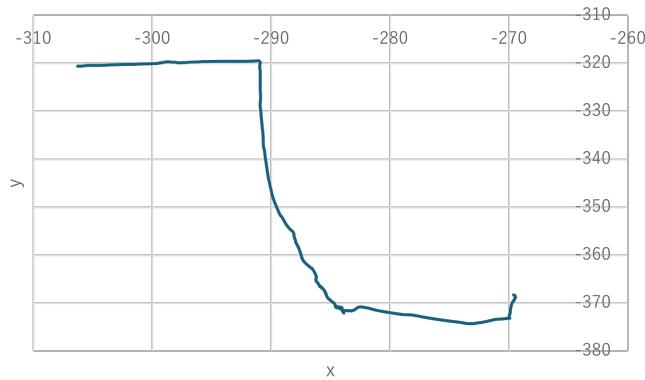


Fig. 4.105 Robot position with trans_stopped_velocity = 0.1



Fig. 4.106 Robot orientation with trans_stopped_velocity = 0.4

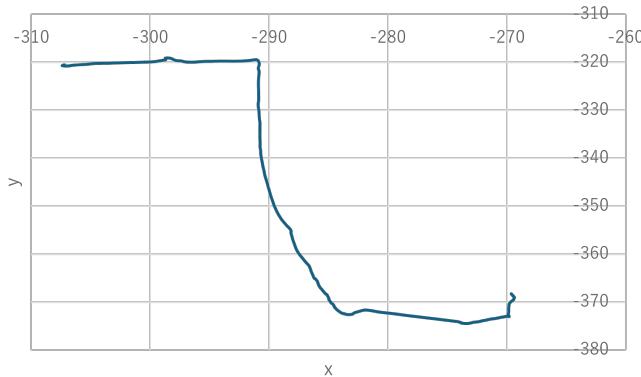


Fig. 4.107 Robot position with trans_stopped_velocity = 0.4

実験の結果、`trans_stopped_velocity` の値を大きく設定した場合、ロボットが完全に減速する前の段階で停止状態と判定されやすくなり、Waypoint を通過する際のゴール判定位置が Waypoint からやや離れた位置で行われる傾向が確認された。そのため、Waypoint 付近での軌道修正が少なくなり、通過精度が低下する挙動が見られた。

一方で、`trans_stopped_velocity` の値を小さく設定した場合、ロボットはより低速になるまで移動を継続するため、Waypoint 近傍で正確に曲がる、あるいは直進する挙動が観測された。これにより、Waypoint 周辺での軌道精度が向上し、より意図した経路に沿った走行が実現できていた。

なお、いずれの設定値においても、最終的には安定してゴールに到達しており、ゴール失敗や停止不能といった問題は発生しなかった。以上の結果から、`trans_stopped_velocity` はロボットの到達可否そのものに影響を与えるパラメータではなく、Waypoint 通過時の判定精度や挙動の細かさを調整するための補助的なパラメータであると考えられる。

本実験環境においては、より正確な Waypoint 通過を実現するため、`trans_stopped_velocity` の値は現行設定よりもやや小さな値に設定することが有効である可能性が示唆された。

実験結果 (planner_tolerance)

Fig. 4.4, Fig. 4.5 で示すように基準の値は、0.5



Fig. 4.108 Robot orientation with planner_tolerance = 0.1

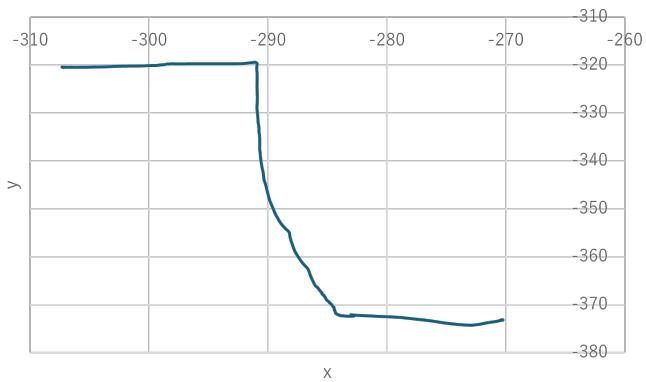


Fig. 4.109 Robot position with planner_tolerance = 0.1



Fig. 4.110 Robot orientation with planner_tolerance = 1.0

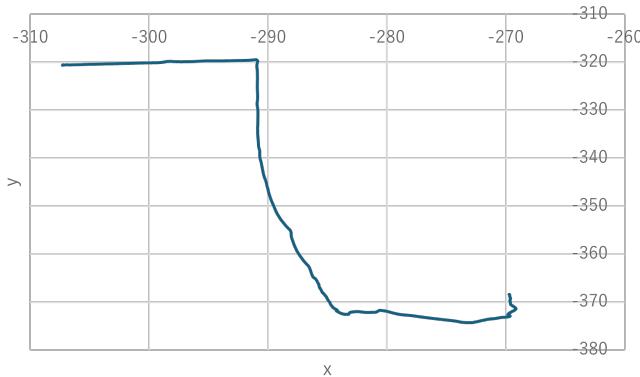


Fig. 4.111 Robot position with planner_tolerance = 1.0

実験では，planner_tolerance を 0.1 および 1.0 に設定して比較した。その結果，planner_tolerance を 1.0 とした場合，経路上で曲がり始めるタイミングが 0.1 の場合と比べてやや早くなる傾向が確認された。これは，ゴールに対する許容範囲が広がったことで，より早い段階で経路生成が完了し，経路の形状がわずかに変化したためであると考えられる。

しかしながら，この違いはロボットの走行挙動に大きな影響を与えるものではなく，速度変化や安定性，ゴール到達の可否といった点において，顕著な差は見られなかった。また，その他の区間においても，planner_tolerance の値を変更したことによる明確な挙動の変化は観測されなかった。

以上の結果から，planner_tolerance は経路生成の細かな形状やタイミングに影響を与えるものの，本実験環境においてはロボットの走行安定性や到達性能に対する影響は小さいパラメータであると考えられる。

以上の結果より，Planner に関するパラメータは，「どのような経路を生成するか」という大域的な計画に影響を与える一方で，ロボットの実際の運動挙動や安定性に対しては，Controller や Localization に比べて影響が小さいことが明らかとなった。そのため，Planner のパラメータ調整は，経路の形状や計画のタイミングを整える目的で行い，ロボットの走行安定性や挙動の改善については，Controller や自己位置推定に関するパラメータ調整を優先すべきであると考えられる。

実験結果（速度関係）

sokudo

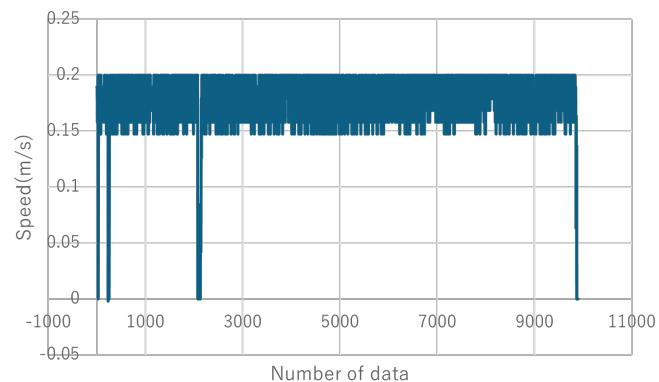


Fig. 4.112 Robot speed with $\text{max_vel_x} = 0.2$

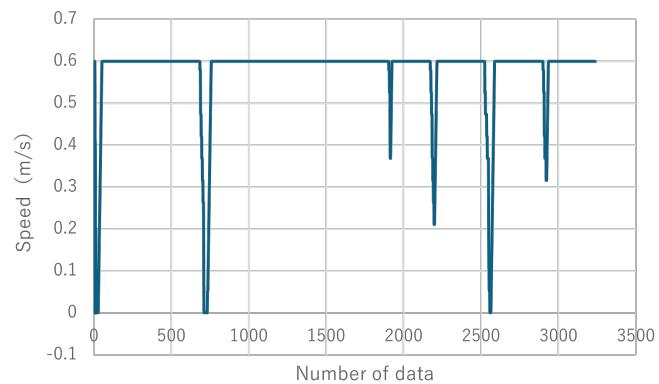


Fig. 4.113 Robot speed with $\text{max_vel_x} = 1.0$

0.6 速度,maxvelx,maxspeedxy,maxvelsmoother,

第5章

結論

5.1 まとめ

5.1.1 emcl2 パラメータ調整に関する総括

本研究では、emcl2 における自己位置推定性能がロボットの走行挙動に与える影響を明らかにするため、オドメトリ誤差モデルおよびレーザ尤度モデルに関する複数のパラメータを変更し、その挙動を比較・評価した。

まず、前進量に比例するオドメトリ誤差を表す fw_dev_per_fw の実験では、値を大きくするにつれてロボットの位置推定誤差が累積し、直進走行時に進行方向から徐々にずれていく挙動が確認された。特に 0.5 の場合にはゴールに到達できず、自己位置推定においてオドメトリ誤差の影響が支配的になることが示された。一方で、0.4 以下ではゴール可能であり、適切な範囲での設定が重要であることが分かった。

fw_dev_per_rot に関する実験では、すべての設定値においてゴール到達が可能であった。値を大きくした場合には位置のずれが確認されたものの、ナビゲーション全体への致命的な影響は見られず、回転時の前進誤差は比較的ロバストであることが示唆された。

次に、前進量に比例する回転誤差を表す rot_dev_per_fw の実験では、値を大きくしてもロボットはゴールまで到達可能であったが、直進区間ににおいてロボットの姿勢が徐々にずれていく傾向が確認された。特に、高い設定値では進行方向が基準値よりも小さくなり、姿勢推定精

度の低下が見られた。

回転量に比例する回転誤差を表す `rot_dev_per_rot` の実験では、値を大きくすることでレーザセンサを重視した自己位置推定となり、ロボットの挙動が不安定になる傾向が確認された。特に曲線走行時には、ゴール付近で停止する事例が観測され、過度なセンサ重視はナビゲーション性能を低下させる可能性が示された。

レーザ尤度モデルに関する実験では、`laser_likelihood_max_dist` を大きく設定すると、レーザスキャンの評価が必要な場面や、スキャン対象物が多い環境において、ロボットが一時的に停止する挙動が確認された。一方で、0.5程度までの設定では、安定した走行が可能であることが分かった。

また、`range_threshold` の実験では、基準値では近距離のスキャンを用いた安定した自己位置推定が行われていたが、値を大きくすると遠距離のスキャンを用いるため、位置および姿勢推定にずれが生じる場合があった。しかし、0.5程度までであれば、安定性を保ったままナビゲーションを行うことが可能であった。

以上の結果より、emcl2における各パラメータは、オドメトリ情報とレーザセンサ情報の信頼度のバランスを調整する役割を持ち、設定値によって自己位置推定の特性およびロボットの走行挙動が大きく変化することが明らかとなった。そのため、実環境におけるナビゲーションでは、各パラメータの意味を理解した上で、環境特性に応じた適切な調整が重要であるといえる。

etc...

参考文献

- [1] Ros 2 documentation — ros2 documentation:rolling. <https://docs.ros.org/en/rolling/index.html>. (Accessed on 12/18/2025).
- [2] Nav2 — nav2 1.0.0 documentation. <https://docs.nav2.org/>. (Accessed on 11/27/2025).
- [3] emcl2/ryuichiueda — 拡張リセット付きの mcl(バージョン 2). <https://github.com/ryuichiueda/emcl2>. (Accessed on 12/18/2025).
- [4] 井口颯人, 橋高聖人, 野村駿斗, 村林孝太郎, 上田隆一, 林原靖男. 屋外自律移動ロボットプラットフォーム orne-box の開発 orne-box の検証・改良 . ロボティクス・メカトロニクス講演会講演概要集 2023, pp. 1P1–I06. 一般社団法人 日本機械学会, 2023.
- [5] Tsudanuma challenge. <https://sites.google.com/p.chibakoudai.jp/rdc-lab/development/tsudanuma-challenge>. (Accessed on 12/22/2025).

付録

謝辞

本研究を進めるにあたり，2年に渡り，熱心にご指導を頂いた林原靖男教授に深く感謝いたします。