

卒業論文

ROS ベースの自律移動ロボットにおける ナビゲーション調整手順の体系化

Systematization of Navigation Adjustment Procedures
for ROS-Based Autonomous Mobile Robots

2025 年 11 月 16 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
22C1062 塩沢悠星

概要

ROS ベースの自律移動ロボットにおける ナビゲーション調整手順の体系化

本論文では、ROS(Robot Operating System) の Naviagtion stack を対象とした自律移動ロボットにおけるナビゲーション調整手順の体系化について述べる。近年、ROS ベースの自律移動ロボットのナビゲーション技術の活用が進んでいるが、ROS Navigation stack を用いた地図ベースの自己位置推定、経路計画に基づく自律移動を行うためには複数のパラメータを適切に調整する必要がある。しかし、現状ではパラメータ調整に関する明確な指針はほとんど示されておらず、特に屋外環境に関する情報は少ない。本研究では、ナビゲーションの調整手順をドキュメントとしてまとめ、調整方法の一例を示すことを目的とする。作成したドキュメントの有用性を検証するため、ROS Navigation stack を使用し、千葉工業大学津田沼キャンパスで実施される技術チャレンジである津田沼チャレンジのコースで自律走行実験を行った。その結果、事前走行および本走行においてコースを完走することができた。さらに、ROS 初心者の学部 3 年生 3 名が本ドキュメントを用いてナビゲーション調整を行った結果、全員がコースの完走を実現し、本ドキュメントの有用性が確認された。

キーワード：自律移動ロボット、ナビゲーション、パラメータ

abstract

Systematization of Navigation Adjustment Procedures for ROS-Based Autonomous Mobile Robots

This paper describes the systematization of navigation tuning procedures for autonomous mobile robots using the ROS (Robot Operating System) Navigation stack. While the use of ROS-based autonomous mobile robot navigation technology has advanced in recent years, performing map-based self-localization and autonomous navigation based on path planning using the ROS Navigation stack requires the appropriate adjustment of multiple parameters. However, clear guidelines for parameter tuning are currently scarce, with particularly limited information available for outdoor environments. This research aims to document the navigation tuning procedure and provide an example of tuning methods. To verify the usefulness of the created documentation, autonomous navigation experiments were conducted on the course of the Tsudanuma Challenge, a technical challenge held at Chiba Institute of Technology's Tsudanuma Campus, using the ROS Navigation stack. As a result, the vehicle successfully completed the course during both preliminary and main runs. Furthermore, three third-year undergraduate students new to ROS used this document to perform navigation tuning, and all three successfully completed the course, confirming the usefulness of this document.

keywords: Autonomous Mobile Robot, Navigation, Parameter

目次

第 1 章	序論	1
1.1	背景	1
1.2	ROS Navigation Stack のパラメータの調整方法に関する公開情報	2
1.2.1	ROS Navigation Tuning Guide	2
1.2.2	A guide to implement ROS Navigation Stack on any robot	3
1.3	目的	4
1.4	論文の構成	4
第 2 章	要素技術	5
2.1	ROS	5
2.2	オドメトリ	6
2.3	ナビゲーション	6
第 3 章	作成したドキュメント	12
3.1	ドキュメントの構成	12
3.2	ドキュメントの例	13
3.2.1	オドメトリ調整	13
3.2.2	AMCL におけるオドメトリ関連パラメータの調整	14
3.2.3	コストマップのパラメータ調整	17
3.2.4	大規模屋外環境での地図解像度設定	23
第 4 章	津田沼チャレンジによるドキュメントの有用性の検証実験	26
4.1	実験概要	26

4.2	実験結果	28
4.2.1	実験 1	28
4.2.2	実験 2	29
第 5 章	つくばチャレンジによるドキュメントの有用性の検証実験	31
5.1	実験概要	31
5.2	実験結果	32
第 6 章	結論	33
参考文献		34
付録		35
謝辞		36

図目次

1.1	Laser-related parameters(souce: [4])	2
1.2	Odometry and particle filter parameters(souce: [4])	2
1.3	Parameters for TrajectoryPlannerROS(souce: [5])	3
1.4	Parameters for global costmap(souce: [5])	3
1.5	Parameter for local costmap(souce: [5])	4
2.1	ROS topic communication	6
2.2	Localization in Rviz	7
2.3	Created map and aerial photograph	8
2.4	footprint in Rviz	9
2.5	Comparison of local paths at different sim_time	10
2.6	Costmap in Rviz	11
3.1	Scan visualized in Rviz	14
3.2	Example of translational drift	15
3.3	Example of rotational drift	15
3.4	Example of localization failure	16
3.5	Map and scan visualized in Rviz	17
3.6	Local_costmap	18
3.7	Inflated obstacles at cost_scaling_factor=1	19
3.8	Inflated obstacles at cost_scaling_factor=10	19
3.9	Inflated obstacles at cost_scaling_factor=20	20

3.10	Costmap at resolution=0.01	21
3.11	Costmap at resolution=0.1	21
3.12	Costmap at resolution=0.5	22
3.13	Global_costmap	23
3.14	Map before adjustment	25
3.15	Map after adjustment	25
4.1	Course map of the Tsudanuma Challenge 2025(souce: [9])	27
4.2	Course map of the Tsudanuma Challenge 2024(souce: [9])	27
4.3	ORNE-box2	28
4.4	System configuration	28
4.5	Experiment scene	29
5.1	Course map of the Tsukuba Challenge 2025(souce: [8])	32

表目次

3.1	Comparison of characteristics for different map resolutions	24
3.2	Comparison of Map Scale Adjustment in Tsukuba Challenge 2025	24
4.1	Navigation results of the Tsudanuma Challenge 2025	28
4.2	Results of B3 students in the Tsudanuma Challenge 2024	29

第1章

序論

1.1 背景

近年, ROS(Robot Operating System) をベースとした自律移動ロボットのナビゲーション技術が, 警備ロボットや案内ロボットなど, 様々な分野で活用されつつある.[1][2] ただし, ナビゲーションにおいては, ROS Navigation stack[3] のパラメータ, オドメトリ調整のパラメータ, 地図生成のパラメータなど, 複数のパラメータを適切に調整しなければ経路に沿った自律移動を行うことができない.

また, 現状ではこれらのパラメータ調整に関する明確な指針は十分に示されていない. 特に, 後述するインターネット上で公開されている情報の多くは屋内環境を対象したものであり, センサ誤差や環境変動が大きく調整が困難な屋外環境に関する情報は少ない. そのため, 屋外環境を対象としたナビゲーションにおけるパラメータ調整手順の明確化を行えば, 今後 ROS Navigation stack を使用して屋外自律移動を行う方々に対して有益な情報となると考える.

1.2 ROS Navigation Stack のパラメータの調整方法に関する 公開情報

1.2.1 ROS Navigation Tuning Guide

ROS のナビゲーションのパラメータ調整について解説されている資料として, Kaiyu Zheng による ROS Navigation Tuning Guide[4] が挙げられる. このガイドは, ROS Navigation stack の性能を最大化するためのパラメータ調整プロセスを解説するものであり, AMCL や Move Base など, ナビゲーションにおける主要な項目を網羅している.

しかし, Fig. 1.1, Fig. 1.2 のように具体的なパラメータ値が提示され, それらが望ましい設定値であると示している. そのため, 自己位置推定のずれや経路計画の失敗など, 特定の環境や問題に直面した際に, どのパラメータをどのように調整すべきかという手順までは明確に示されていない. 読み手はパラメータに関する知識を得られるものの, 全体としてナビゲーションシステムを最適化していく具体的な調整フローを導き出すのは難しいという課題がある.

```

1 {
2   "laser_z_hit": 0.9,
3   "laser_sigma_hit": 0.1,
4   "laser_z_rand": 0.5,
5   "laser_likelihood_max_dist": 4.0
6 }
```

Fig. 1.1: Laser-related parameters(souce: [4])

```

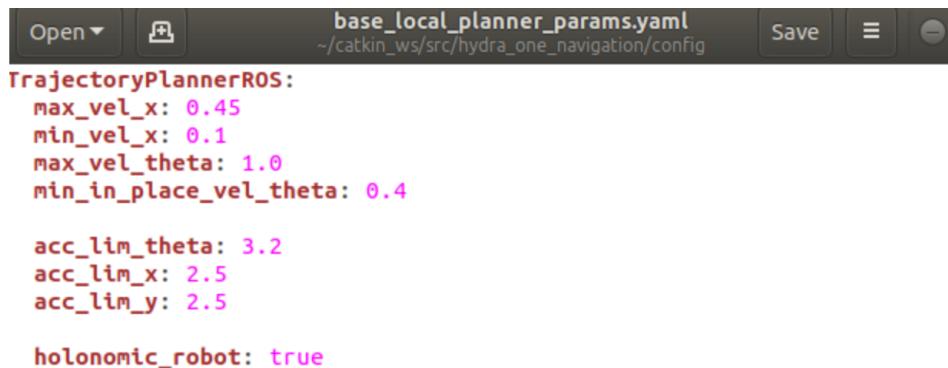
1 {
2   "kld_err": 0.10,
3   "kld_z": 0.5,
4   "odom_alpha1": 0.008,
5   "odom_alpha2": 0.040,
6   "odom_alpha3": 0.004,
7   "odom_alpha4": 0.025
8 }
```

Fig. 1.2: Odometry and particle filter parameters(souce: [4])

1.2.2 A guide to implement ROS Navigation Stack on any robot

ROS のナビゲーションのパラメータ調整について解説されている資料として, A guide to implement ROS Navigation Stack on any robot[5] が挙げられる. このガイドでは, ROS Navigation Stack の主要な構成要素や一部のパラメータ例が紹介されており, シミュレータ上でのナビゲーション実装手順についても解説されている.

しかし, Fig. 1.3, Fig. 1.4, Fig. 1.5 のように具体的な調整方法については明確に示されていない. 読み手はパラメータの設定値の一例は得られるものの, 実際にナビゲーションシステムを向上させるための具体的な調整手順を導き出すことは難しいという課題がある .



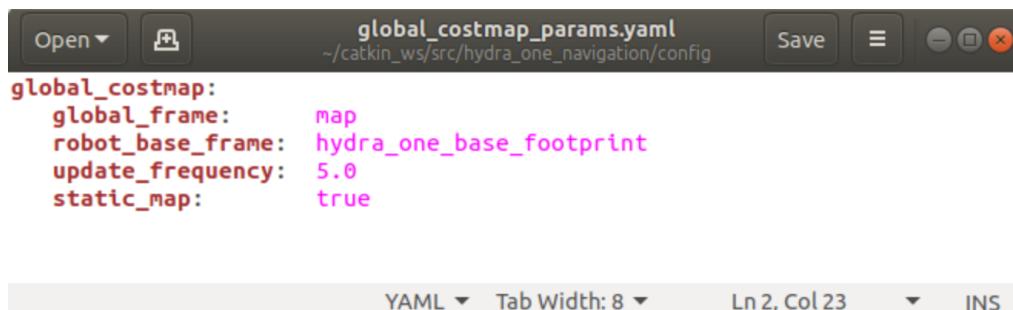
```
base_local_planner_params.yaml
~/catkin_ws/src/hydra_one_navigation/config

TrajectoryPlannerROS:
  max_vel_x: 0.45
  min_vel_x: 0.1
  max_vel_theta: 1.0
  min_in_place_vel_theta: 0.4

  acc_lim_theta: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 2.5

  holonomic_robot: true
```

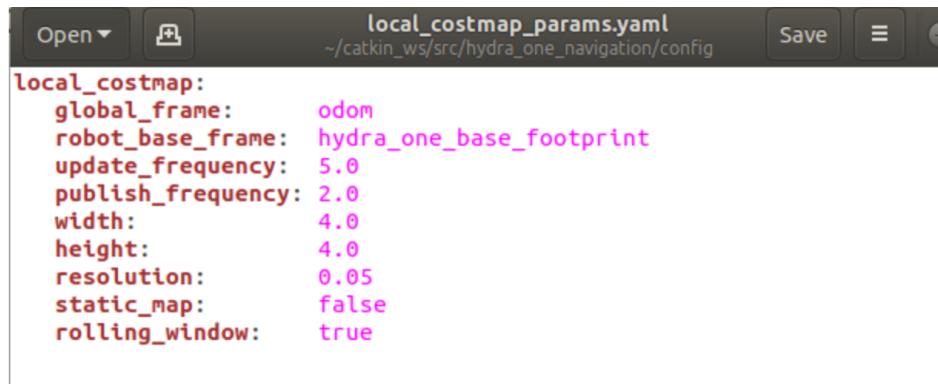
Fig. 1.3: Parameters for TrajectoryPlannerROS(souce: [5])



```
global_costmap_params.yaml
~/catkin_ws/src/hydra_one_navigation/config

global_costmap:
  global_frame: map
  robot_base_frame: hydra_one_base_footprint
  update_frequency: 5.0
  static_map: true
```

Fig. 1.4: Parameters for global costmap(souce: [5])



```
local_costmap:  
  global_frame:      odom  
  robot_base_frame: hydra_one_base_footprint  
  update_frequency: 5.0  
  publish_frequency: 2.0  
  width:            4.0  
  height:           4.0  
  resolution:       0.05  
  static_map:        false  
  rolling_window:   true
```

Fig. 1.5: Parameter for local costmap(souce: [5])

1.3 目的

本論文では、ロボットにおける ROS Navigation stack を用いたナビゲーションの調整手順を体系化し、調整方法の一例を示すことを目的とする。

1.4 論文の構成

本論文では以下のように構成される。

2 章で本研究で使用される要素技術について述べる。

3 章では本研究で作成したドキュメントについて述べる。

4 章では津田沼チャレンジによる実験でドキュメントの有用性を検証する。

5 章ではつくばチャレンジによる実験でドキュメントの有用性を検証する。

6 章では本論文について結論を述べる。

第2章

要素技術

2.1 ROS

ROS(Robot Operating System) は、ロボット開発を効率化するためのオープンソースソフトウェアフレームワークである。複数のプログラミング言語向けのライブラリや、センサや状態を可視化するツール群、ノード間のメッセージ通信機構、およびパッケージによるモジュール管理機能を備えている点が特徴である。Fig. 2.1 に示すように、ROS における基本的な実行単位はノードであり、ノードはトピックやサービスといった仕組みを通じて情報を交換する。これにより、センサデータの取得や制御アルゴリズム、経路計画などを個別のモジュールとして構成し、分散して動作させることが可能となる。さらに、既成のアルゴリズムやデバイス向けソフトウェアをパッケージとして利用することで、開発者は低コストかつ短期間で複雑な機能を実装することができる。

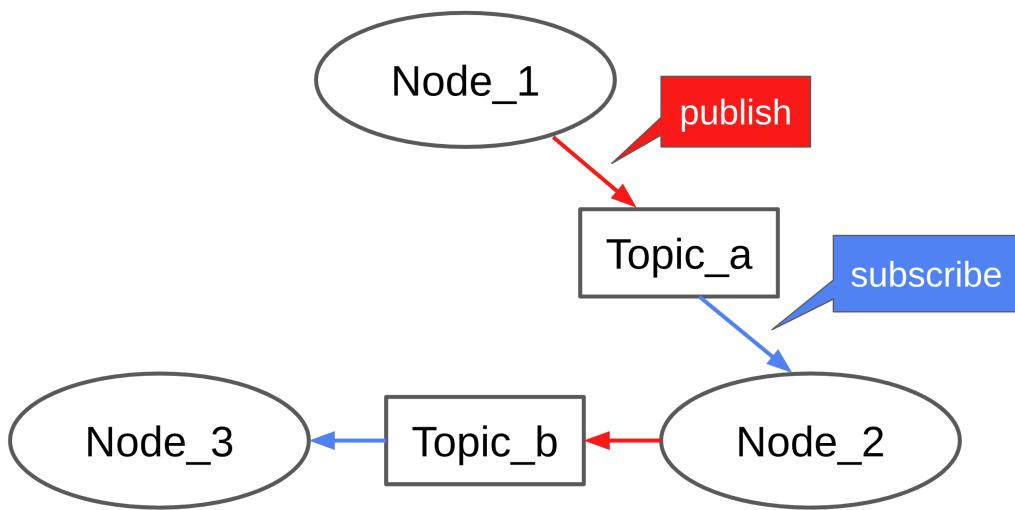


Fig. 2.1: ROS topic communication

2.2 オドメトリ

オドメトリは、ロボットが自己位置を推定するための基礎的な手法であり、車輪の回転量から移動距離および姿勢変化を算出する技術である。具体的には、エンコーダによって取得される各車輪の回転角度から、ロボットの並進移動量および回転角度を求めてることで、時系列的にロボットの位置を更新していく。この手法は、外部装置に依存せずに、連続的な位置推定が可能であるため、自己位置推定や経路追従制御における基本情報として広く用いられている。しかし、車輪の空転やスリップ、床面の凹凸などによって誤差が累積するという問題があり、長時間の走行では位置ずれが大きくなる傾向がある。そのため、ロボットの車輪間距離や各車輪径などのパラメータを事前に調整する必要がある。

2.3 ナビゲーション

ROS Navigation stack[3] は、自律移動ロボットが環境内を自律的に移動するためのソフトウェアフレームワークである。主に以下の要素から構成されている。

- 自己位置推定

ロボットの現在位置を推定する代表的な手法として、AMCL(Adaptive Monte Carlo

Localization) が用いられる。AMCL は ROS Navigation Stack で自己位置推定を行う確率的アルゴリズムであり、LiDAR やオドメトリ情報をもとに多数の仮説（パーティクル）を生成し、それぞれの重みを更新することでロボットの位置を推定する。また、状況に応じてパーティクル数を動的に調整することで精度と計算負荷のバランスを取る特徴をもつ。一方で、その性能はオドメトリ誤差やレーザノイズなど多くのパラメータ設定に依存しており、実環境に応じたチューニングが重要となる。Fig. 2.2 に自己位置推定の様子を示す。

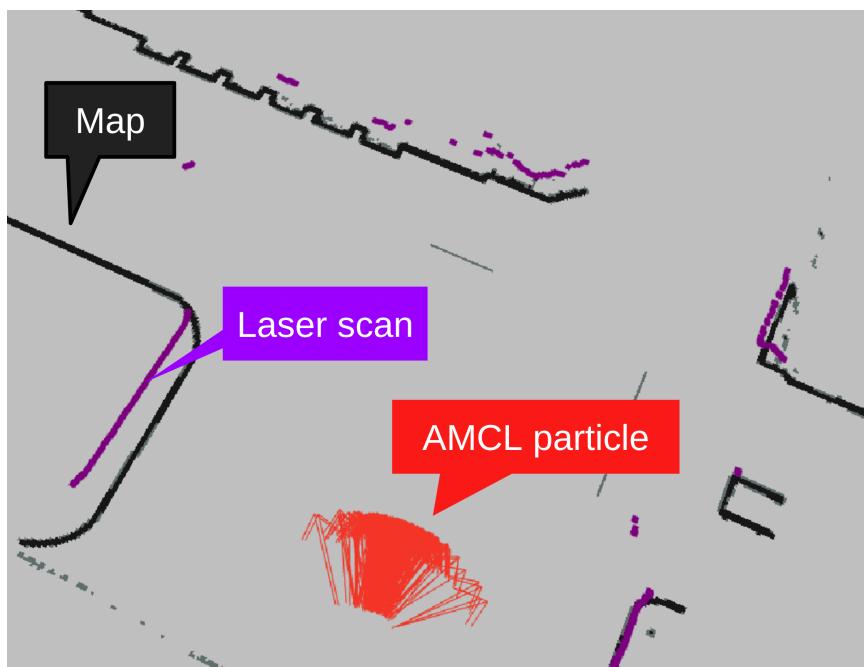


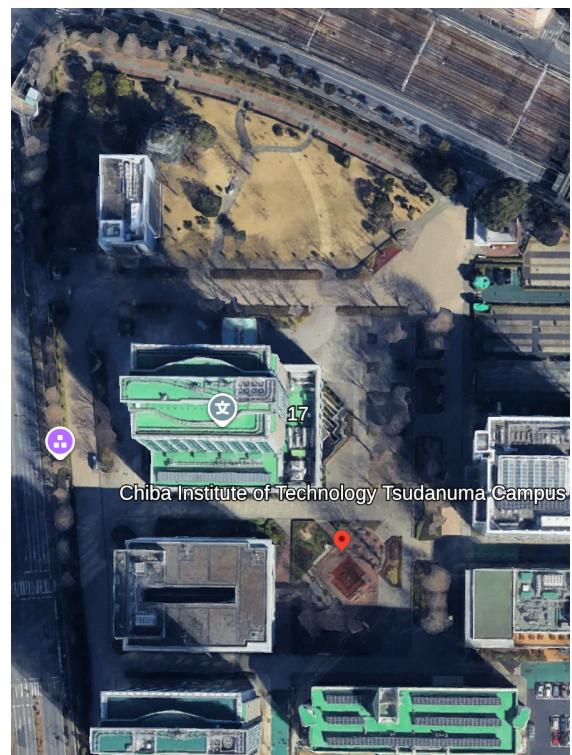
Fig. 2.2: Localization in Rviz

• 地図生成

未知領域においては、環境のマッピングのため、SLAM(Simultaneous Localization and Mapping) が必要となる。SLAM は、ロボットが未知の環境内で自己位置を推定しながら同時に地図を生成するための手法であり、自律移動の基盤技術の一つである。ロボットは LiDAR やカメラなどのセンサ情報を取得し、環境内の特徴点や障害物を検出することで地図を構築すると同時に、自身の位置をその地図上で推定する。この相互依存的な推定を繰り返すことで、外部基準を持たない環境でも自己位置と地図を同時に確立できる。Fig. 2.3 に SLAM で作成された地図とそれに対応する実環境の航空写真を示す。



(a) Map created with SLAM



(b) Aerial photograph(souce: [6])

Fig. 2.3: Created map and aerial photograph

- 経路計画

経路計画は、ロボットが目的地へ到達するための経路を生成、追従するプロセスであり、ナビゲーションシステムの中心的な役割を担う。大域的経路計画は、静的な地図情報をもとに、ロボットの現在位置から目的地までの全体的な経路を計算する段階である。ここでは主に Dijkstra 法や A*アルゴリズムといったグラフ探索手法が用いられ、障害物を避けつつコストマップ上で最短または最適な経路を算出する。この際、ロボットの形状を定義する footprint は経路生成に大きく影響するパラメータである。footprint の設定が小さすぎると、実際には通過不可能な狭い領域を経路として計算してしまう一方、過度に大きく設定すると通行可能な経路を避けてしまうため、実ロボットの外形に基づいた適切な設定が必要となる。Fig. 2.4 に footprint の一例を示す。一方、局所的経路計画は、ロボットが実際に移動する際の動作をリアルタイムに制御する段階であり、センサ情報をもとに動的な障害物を回避しながら経路を追従する。代表的な手法である DWA(Dynamic Window Approach) は、ロボットの運動学的制約

を考慮しつつ、速度空間内で安全かつ効率的な制御コマンドを探索するアルゴリズムである。DWA は目標方向の進行性、障害物との距離、安全性など複数の評価関数を組み合わせて最適な行動を選択することで、滑らかな走行と衝突回避を両立させている。この最適な行動を選択するシミュレーション時間を決定するのが sim_time パラメータであり、値を大きくすると遠い将来までの経路を考慮するが、計算負荷が増加する。Fig. 2.5 に異なる sim_time における経路を示す。また、経路計画の更新頻度を制御するパラメータとして、planner_frequency(グローバルプランナーの再計算周期) および controller_frequency(ローカルプランナーの制御出力周期) がある。これらの値は PC の処理能力や環境に応じて適切に調整する必要がある。

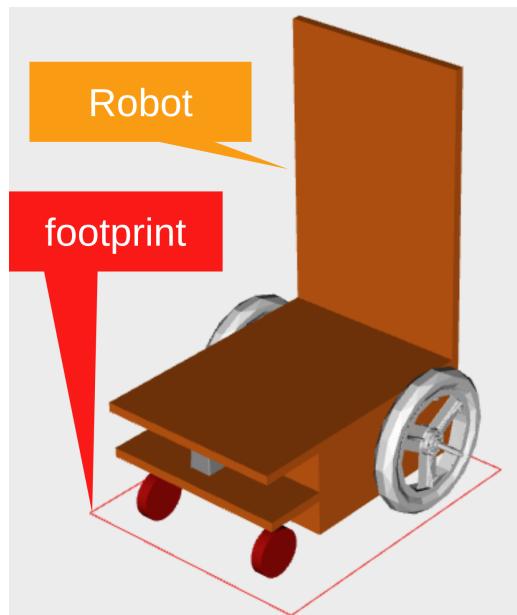


Fig. 2.4: footprint in Rviz

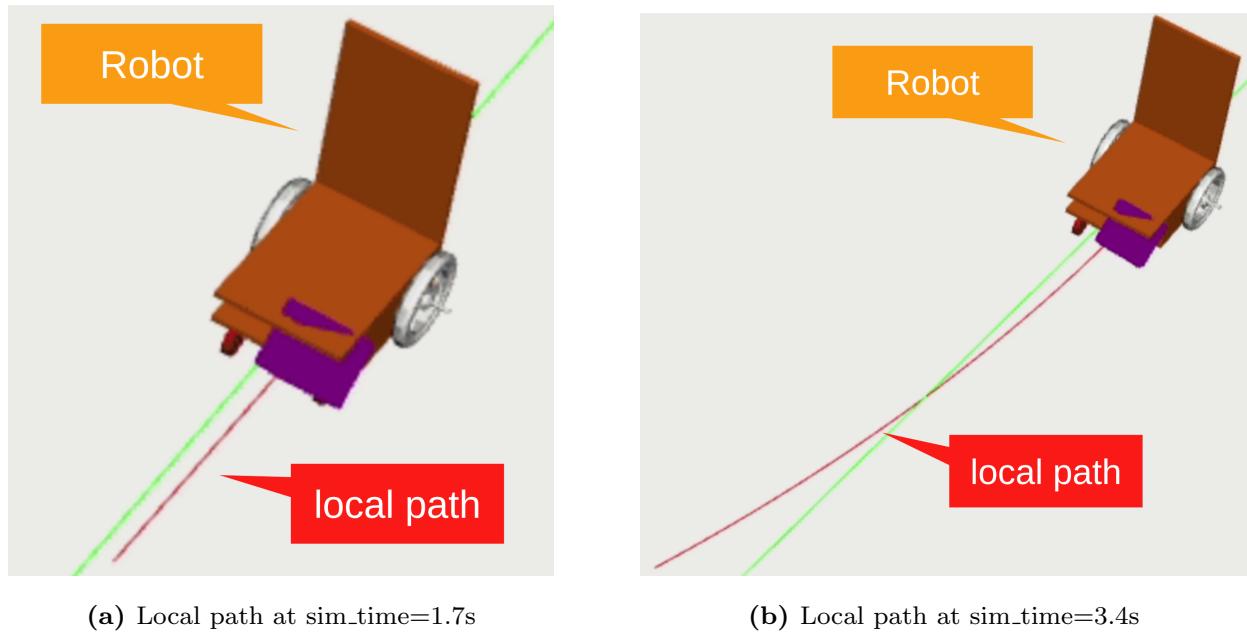


Fig. 2.5: Comparison of local paths at different sim_time

- コストマップ

コストマップは、ロボットの経路計画において基盤となる情報構造であり、環境内の障害物や走行困難な領域を数値化して表現することで、経路計画アルゴリズムが安全かつ効率的に移動できるようにする。ROS Navigation stack では、コストマップは、静的マップと動的マップに分けて管理される。静的マップはあらかじめ生成された地図に基づく固定的な障害物情報を提供し、局所的な経路計画や障害物回避の基盤として利用される。一方、動的マップは LiDAR やカメラなどのセンサ情報をもとにパラメータ設定された値で更新され、移動中に出現する人などの動的障害物を反映する。コストマップ上では、障害物が存在するセルの値が高く、通行可能な領域は低い値で表現される。この数値は経路計画アルゴリズムによって考慮され、ロボットはより安全でコストの低い経路を優先して移動する。また、コストマップでは障害物に過度に接近することを防ぐため、障害物の周囲にコストをパラメータに応じて膨張させる処理が行われる。これにより、経路計画は単に障害物を避けるだけでなく、走行中の安全性を確保しつつ目標地点まで到達できるようになる。Fig. 2.6 にコストマップの一例を示す。

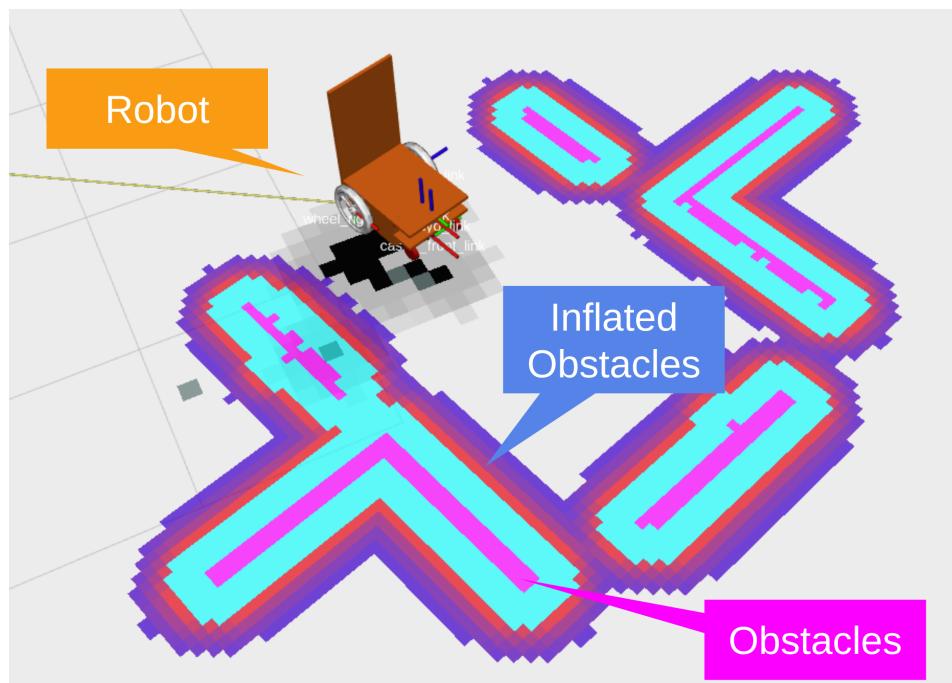


Fig. 2.6: Costmap in Rviz

第3章

作成したドキュメント

3.1 ドキュメントの構成

作成したドキュメント (https://github.com/open-rdc/nav_tuning_guide) の構成は以下の要素から構成されている。また、作成にあたって ROS Wiki の Navigation ページを参考にした。[7]

- 自己位置推定 (AMCL)
 - 必要最小限の設定
 - 主要パラメータの調整
 - その他パラメータの調整
 - **ROS_ERROR** が出たときの問題と対処法
- 経路計画 (Move Base)
 - 必要最小限の設定
 - Move Base の基本となるパラメータ調整
 - リカバリ動作のパラメータ調整
 - コストマップのパラメータ調整
 - ローカルプランナーのパラメータ調整
 - グローバルプランナーのパラメータ調整
 - **ROS_WARN** が出たときの問題と対処法

- 地図
 - 地図作成方法に関して
 - slam_toolbox での地図作成方法
 - glim での地図作成方法
 - 大規模屋外環境での地図解像度設定

- オドメトリ
 - オドメトリの調整手順

3.2 ドキュメントの例

本論文では、作成したドキュメントの中から例として、オドメトリ調整と自己位置推定 (AMCL)、コストマップ、地図の 4 項目を取り上げ、記載内容の一部を示す。

3.2.1 オドメトリ調整

調整対象のパラメータは、車輪半径に関する補正係数である `wheel_radius_multiplier` と、車輪環距離に関する補正係数である `wheel_separation_multiplier` の 2 つである。これらはロボットのオドメトリの正確さに直結するため、自己位置推定を行う上で最初に調整すべき項目である。

調整手順は以下の通りである。まず、Rviz を用いてトピックの可視化の準備を行う。固定座標系を `odom` に設定し、`LaserScan` トピックを表示することで、ロボットの移動に伴うセンサ計測ができるようにする。

次に、ロボットを壁から数メートル離れた位置に配置し、直進させて並進成分の正確さを確認する。このとき、得られるレーザスキャンに厚みが生じる場合は、`wheel_radius_multiplier` を調整する。

さらに、ロボットをその場で回転させ、回転成分の正確さを確認する。スキャンが 1~2 度以上ずれている場合は、`wheel_separation_multiplier` を調整する。最後に再度直進させ、並進と回転の双方でスキャンが一致していることを確認した時点で調整を完了とする。

Fig. 3.1 にオドメトリ調整前後のレーザスキャンを Rviz で可視化した様子を示す。

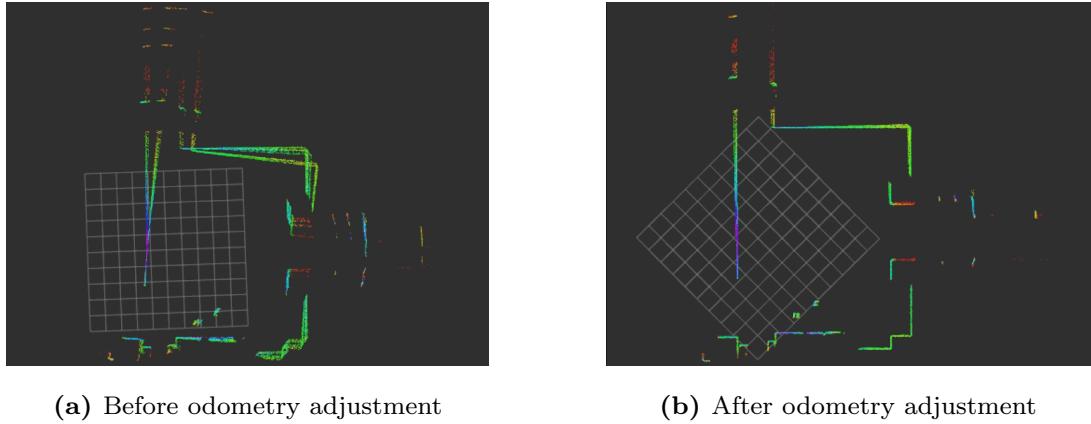


Fig. 3.1: Scan visualized in Rviz

3.2.2 AMCL におけるオドメトリ関連パラメータの調整

自己位置推定に用いる AMCLにおいて、最初に調整すべきパラメータは `odom_alpha1 ~ odom_alpha4` である。これらはオドメトリの信頼度を決定する値であり、大きな値を設定すると、オドメトリに含まれるノイズが大きいとみなされ、オドメトリの影響が小さくなる。一方で、小さな値を設定するとオドメトリを強く信頼するようになる。屋外環境ではオドメトリ誤差が大きくなるため、パラメータを過度に小さくすると誤推定に繋がる危険がある。特に `odom_alpha2` と `odom_alpha3` の調整が有効である。

調整方法は以下の通りである。まず、走行データを `rosbag` で記録し、`Rviz` を用いてパーティクルの散らばりや自己位置のずれ方を可視化する。これにより、どのパラメータが問題に寄与しているかを予測できる。その後、記録したデータを再生し、パラメータを変更しながらオフラインで AMCL を動作させることで、パーティクルの挙動を確認できる。

調整の基準としては、スキャンデータと地図の対応関係を利用する。例えば、Fig. 3.2 に示すように、スキャンが並進方向にずれる場合は `odom_alpha3` を増加させることで改善できる。また、Fig. 3.3 のように、回転方向にずれる場合は `odom_alpha2` を増加させることで修正を試みる。Fig. 3.4 に示すように、自己位置が徐々にずれていく場合には、`odom_alpha1 ~ odom_alpha4` させることで安定化を図る。特に、`alpha2` と `alpha3` の調整が有効である。

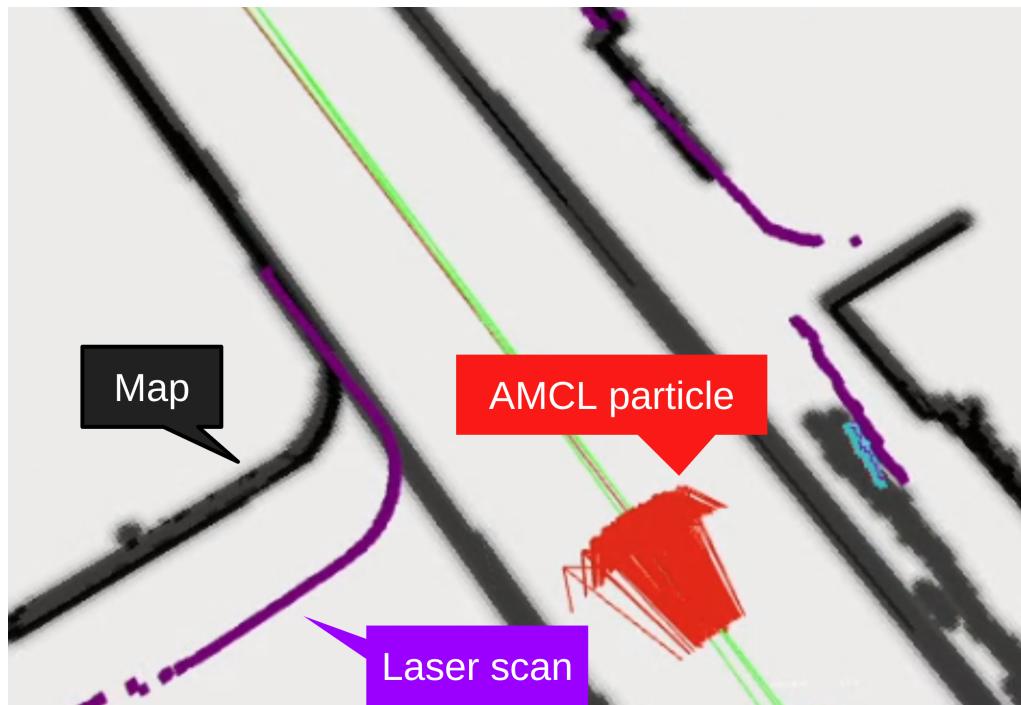


Fig. 3.2: Example of translational drift

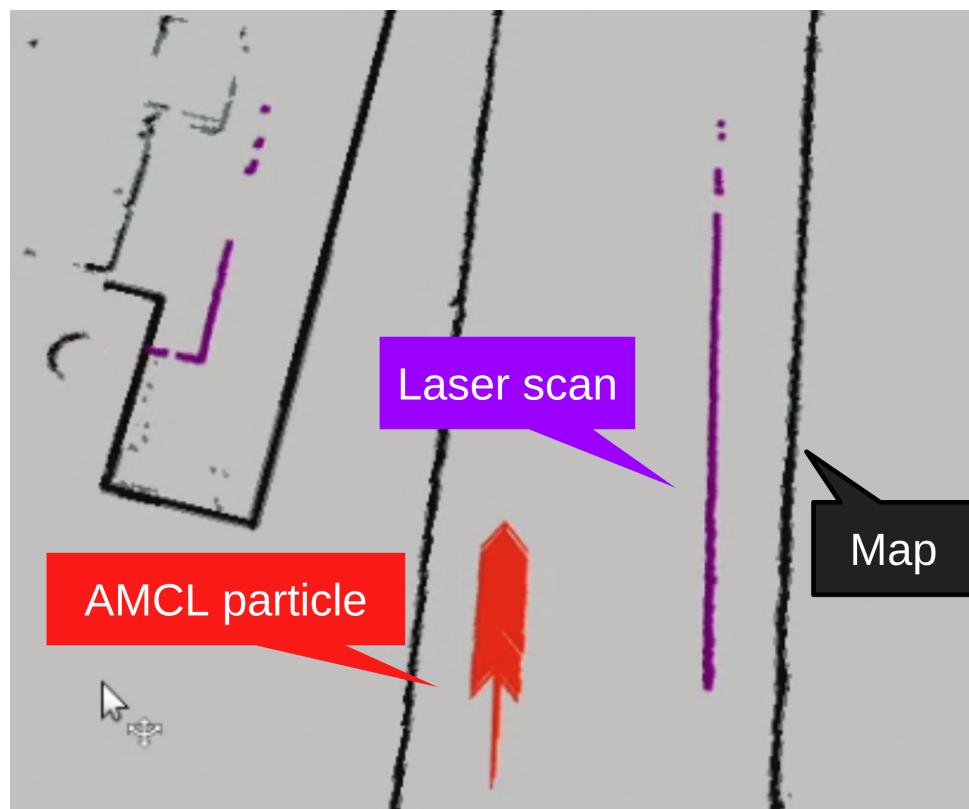


Fig. 3.3: Example of rotational drift

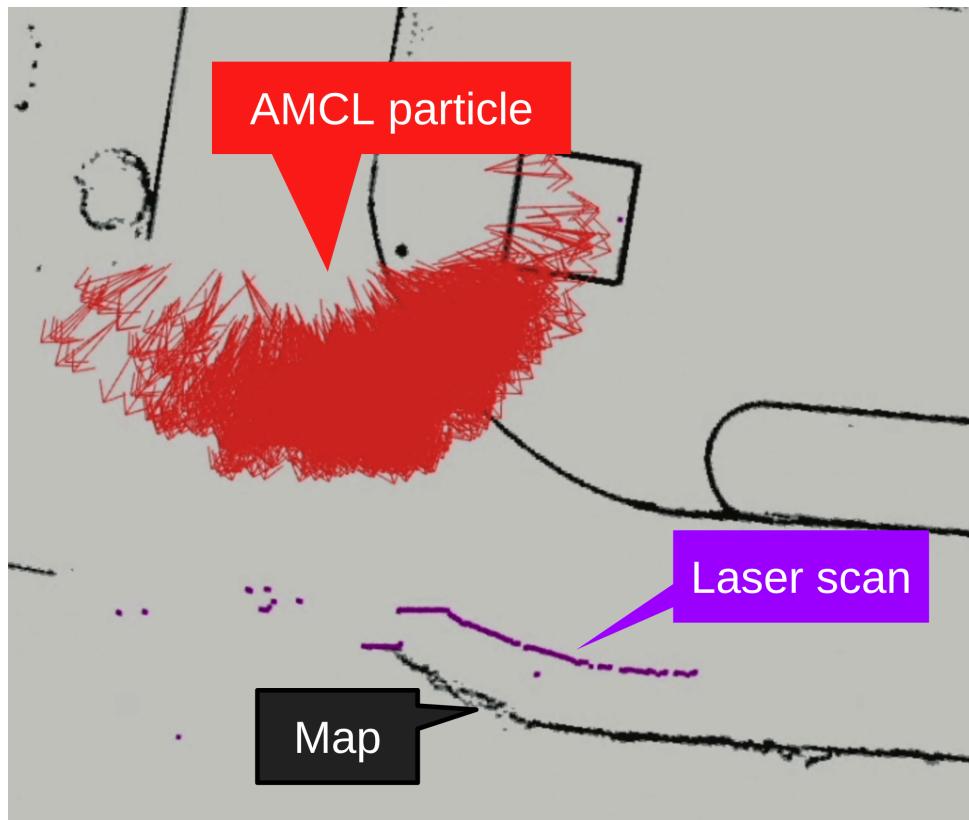


Fig. 3.4: Example of localization failure

調整完了の基準としては、コントローラ操作時の rosbag を再生した際に自己位置の破綻がなく、かつ実ロボットによる自律走行においてもゴールまで破綻なく移動できることである。

さらに、AMCL の調整においては、地図とオドメトリのどちらを信頼するかというトレードオフが存在する。地図が高精度で環境と一致している場合には、スキャンマッチングが有効に働くため、odom_alpha を大きめに設定しても安定した推定が得られる。一方で、地図の精度が低い場合や環境変動が大きい場合には、オドメトリを相対的に信頼する方が安定する。ただし、オドメトリへの依存度を上げすぎると、特に屋外環境では累積誤差によって自己位置が破綻する危険がある。このように、状況に応じてバランスを取ることが、AMCL のパラメータ調整の難しさであるといえる。

Fig. 3.5 に AMCL 調整前後のレーザスキャンと地図を Rviz で可視化した様子を示す。

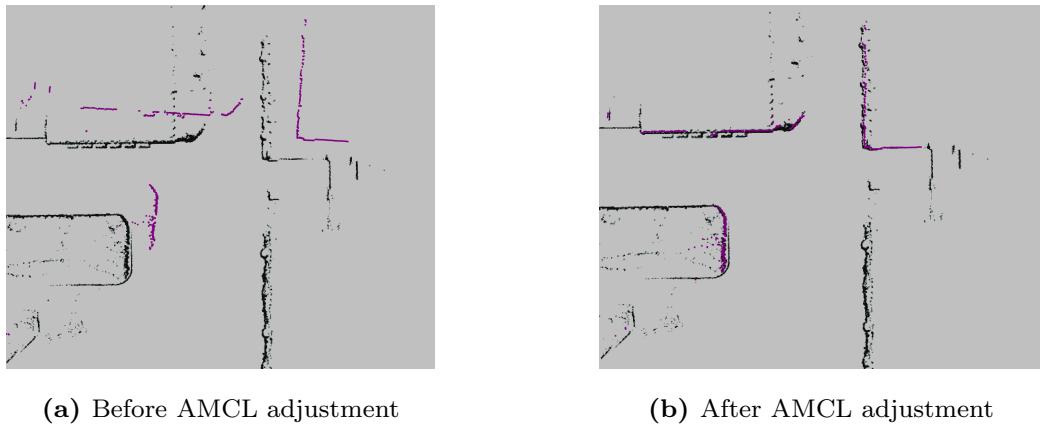


Fig. 3.5: Map and scan visualized in Rviz

3.2.3 コストマップのパラメータ調整

costmap は、ロボットの周囲環境を表現する重要なマップである。デフォルト設定のままで動作は可能であるが、障害物回避性能や応答性を向上させたい場合には、各種パラメータを調整することが有効である。

costmap には local_costmap と global_costmap の 2 種類が存在し、両者で共通するパラメータが多い。しかし、名前空間ごとに独立してパラメータを定義することで、それぞれの役割に適した設定値を与えることが可能である。また、名前空間で個別に値を指定しなかった場合には、そのパラメータ値が local_costmap と global_costmap の両方に適用される。

まず、local_costmap の主なパラメータについて述べる。update_frequency は地図の更新頻度を表し、値を大きくしすぎると動的障害物の変化を適切に反映できなくなる可能性がある。一方で、小さくしすぎると CPU 負荷が増大し、処理が滞る可能性がある。また、この値が適切でない場合には、実行時に ROSWARN が出るため注意が必要である。

local_costmap の幅と高さを設定する width と height は、Fig. 3.6 に示すように、ロボット周辺に生成される局所領域の大きさを規定する。値を大きくするほど広範囲の障害物情報を取り込めるが、その分だけ計算量は増加する。

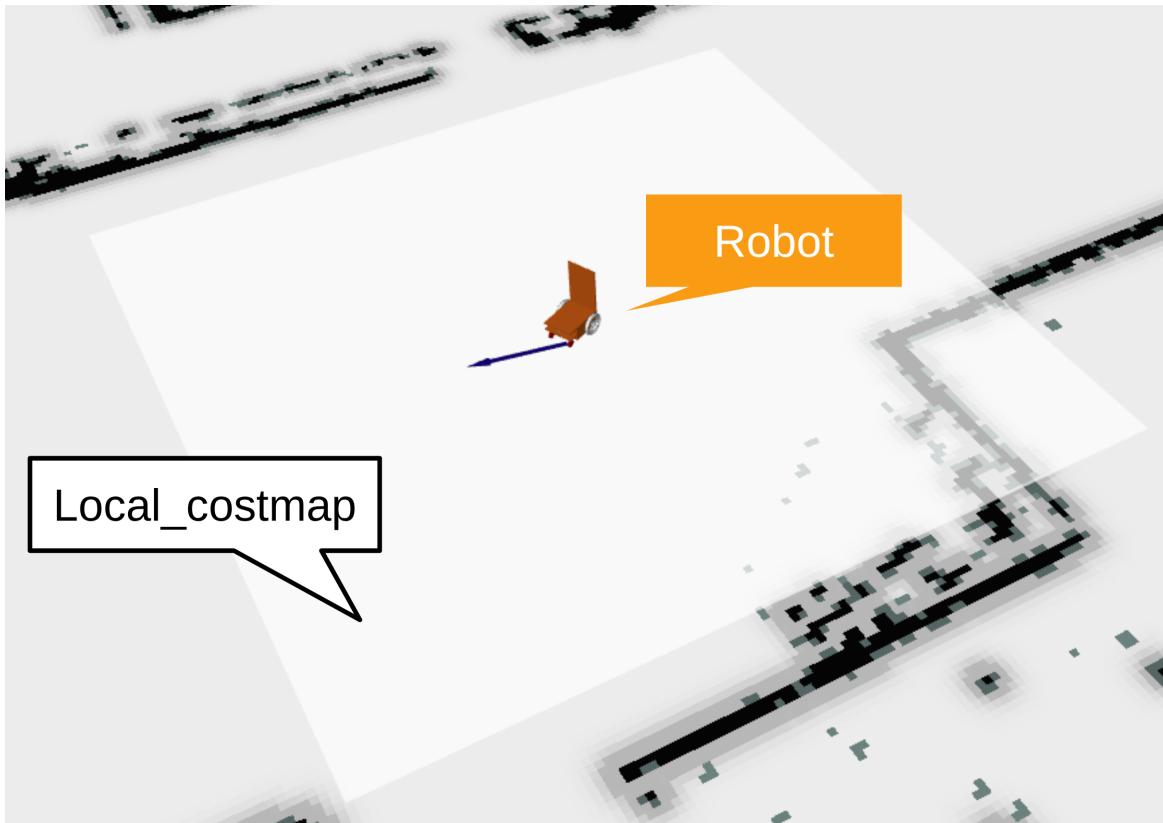


Fig. 3.6: Local_costmap

`inflation_radius` は障害物をどの程度膨張させるかを決定するパラメータである。障害物セルから周囲に、コストを膨張させる値であり、安全な経路を生成する際に重要となる。`inflation_radius` に加えて、`cost_scaling_factor` を用いることで、膨張したコストの減衰具合を調整することができる。`cost_scaling_factor` の値を大きくしすぎると、`inflation_radius` を変更しても、実際にコストが膨張する範囲が極端に狭くなるため、注意が必要である。Fig. 3.7, Fig. 3.8, Fig. 3.9 に `inflation_radius` の値を固定し、`cost_scaling_factor` の値を変更したときのコストの膨張具合を示す。

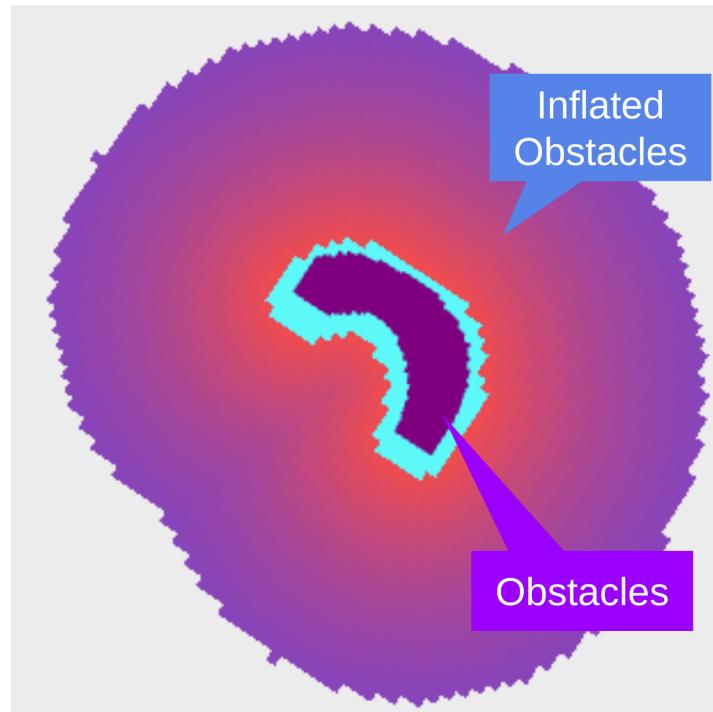


Fig. 3.7: Inflated obstacles at cost_scaling_factor=1

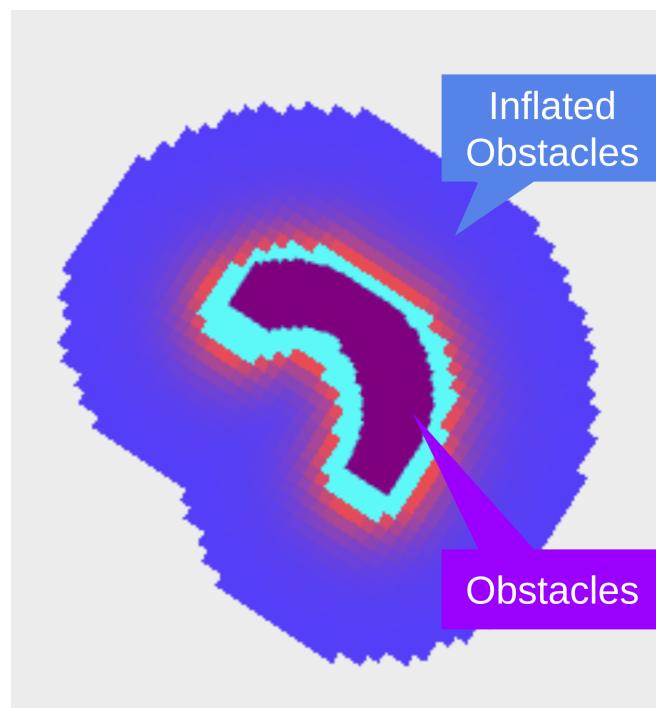


Fig. 3.8: Inflated obstacles at cost_scaling_factor=10

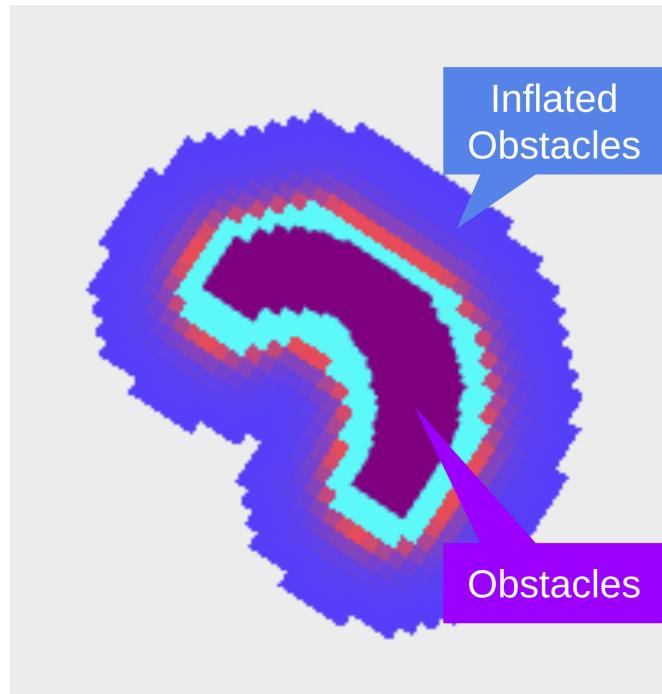


Fig. 3.9: Inflated obstacles at cost_scaling_factor=20

resolution は costmap を構成する 1 セルにおける大きさを示す値であり、小さくするほど高精細な地図となる一方、計算量は増加する。逆に、resolution を大きくすると粗い地図となり計算量を削減できるが、障害物の形状が不正確になる可能性がある。Fig. 3.10, Fig. 3.11, Fig. 3.12 に異なる resolution におけるコストマップを示す。

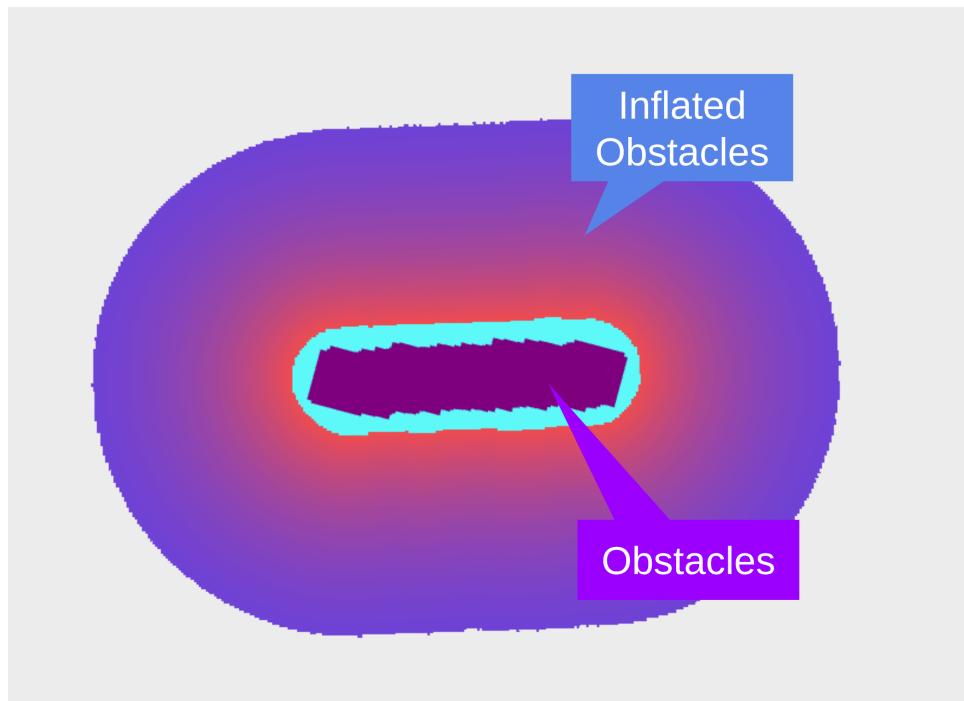


Fig. 3.10: Costmap at resolution=0.01

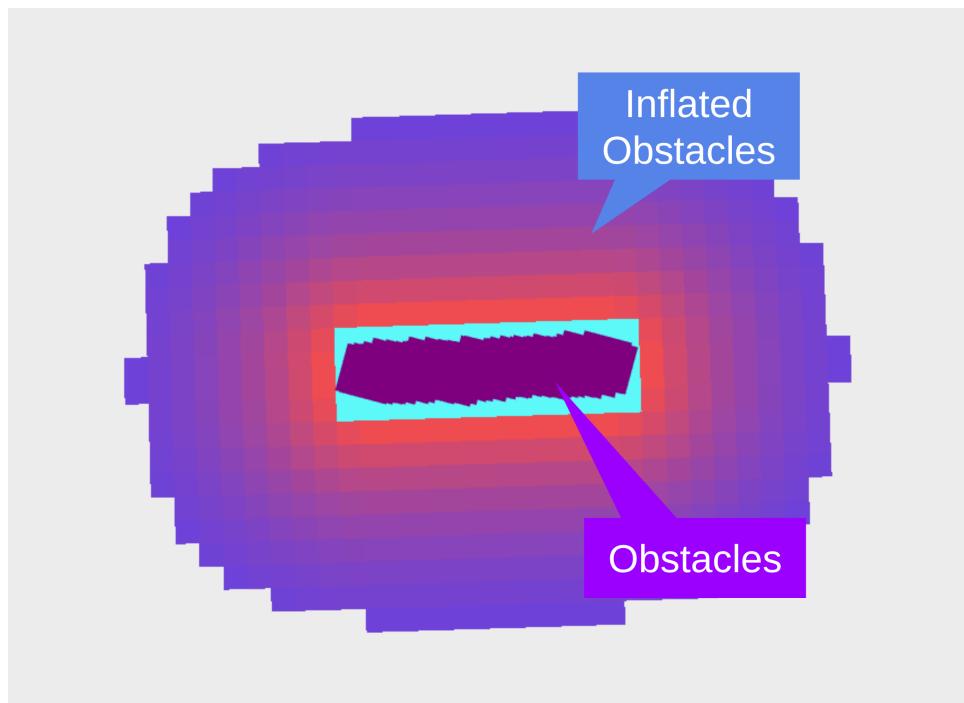


Fig. 3.11: Costmap at resolution=0.1

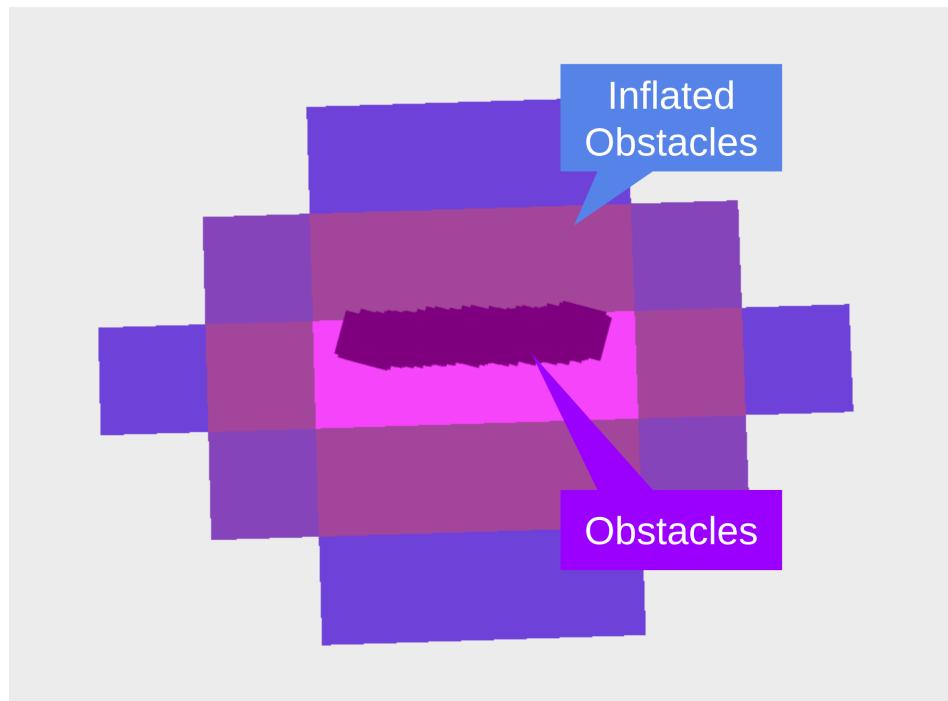


Fig. 3.12: Costmap at resolution=0.5

次に, global_costmapについて述べる. 上記で述べたパラメータは local_costmap と同様の意味を持つ.しかし, inflation_radius に関しては, local_costmap と異なり, static_layer によって読み込まれた静的マップ全体に対して膨張処理が適用される点に特徴がある. Fig. 3.13 に global_costmap の一例を示す.

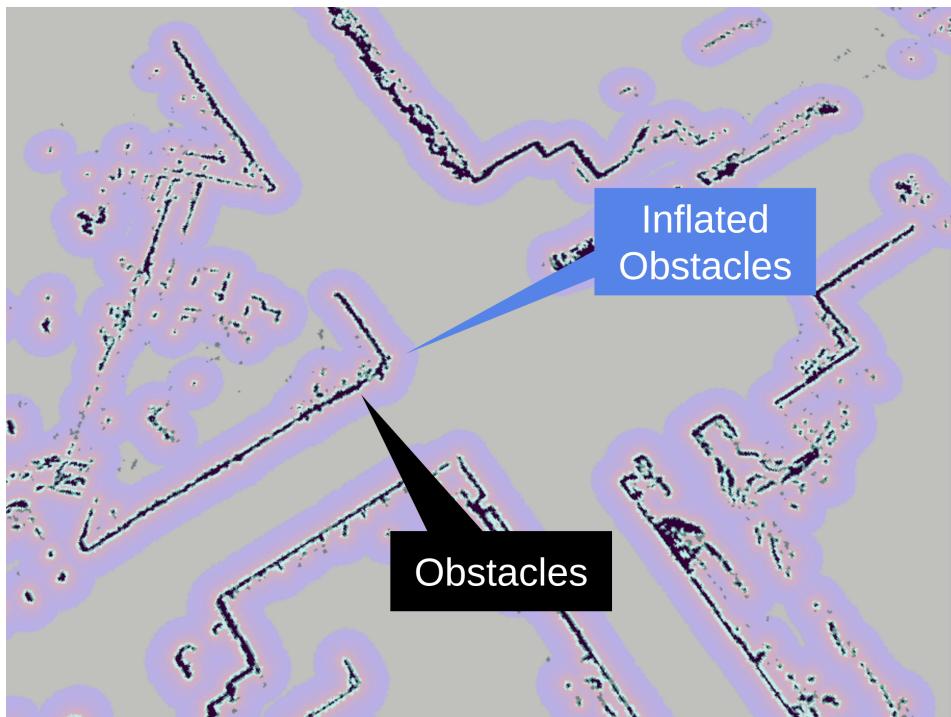


Fig. 3.13: Global_costmap

以上のように, costmap のパラメータはロボットの経路計画と障害物回避性能に直接影響するため, 環境に応じた調整が必要となる.

3.2.4 大規模屋外環境での地図解像度設定

つくばチャレンジ [8] のような大規模な屋外環境における自律移動では, 使用する地図の総セル数が非常に大きくなる. このとき, 経路計画で用いられる Dijkstra 法の計算量は, 地図の総セル数に強く依存する. そのため, セル数が増大すると計算負荷が高まり, ロボットが停止を繰り返すなどの問題が生じることがある. 実際に, つくばチャレンジ 2025 の初回実験走行では, 計算領域が過大であったために, ロボットが断続的に停止する様子が確認された.

ROS の map_server が扱う地図は格子状のセルで構成されており, 1 セルが表す物理距離は解像度 (resolution) によって定義される. 解像度を大きく設定すると, 1 セルあたりの距離が広がり, 結果として地図全体のセル数が減少する. 例えば, resolution を 0.10 から 0.20 に変更した場合, 格子は荒くなるが, 総セル数は大きく減少し, 経路計画の計算速度は向上する. Table 3.1 に異なる解像度における特性の比較を示す.

つくばチャレンジ 2025においては, 地図のスケールを調整した. その結果を, Table 3.2 に示

Table 3.1: Comparison of characteristics for different map resolutions

	Low resolution	High resolution
Grid granularity	Fine	Coarse
Number of total cells	Large	Small
Computational cost	High	Low

す。総計算領域を約 65% 削減することができた。この削減により、Dijkstra 法による経路探索の計算量が大幅に低減し、ロボットが停止を繰り返す問題を解消することができた。Fig. 3.14, Fig. 3.15 にスケールを統一した調整前、調整後の地図を示す。

ただし、解像度を過度に大きくすると、地図が粗くなり、自己位置推定が不安定になったり、狭い通路がセルの粗さによって潰され、通行不可能と判断される場合がある。そのため、解像度は単に大きくすればよいというものではないことを注意する必要がある。

Table 3.2: Comparison of Map Scale Adjustment in Tsukuba Challenge 2025

	Image size [pixel]	Total number of cells	resolution
Before adjustment	7077 × 3773	Approx. 26.69 million	0.10
After adjustment	4163 × 2219	Approx. 9.24 million	0.17

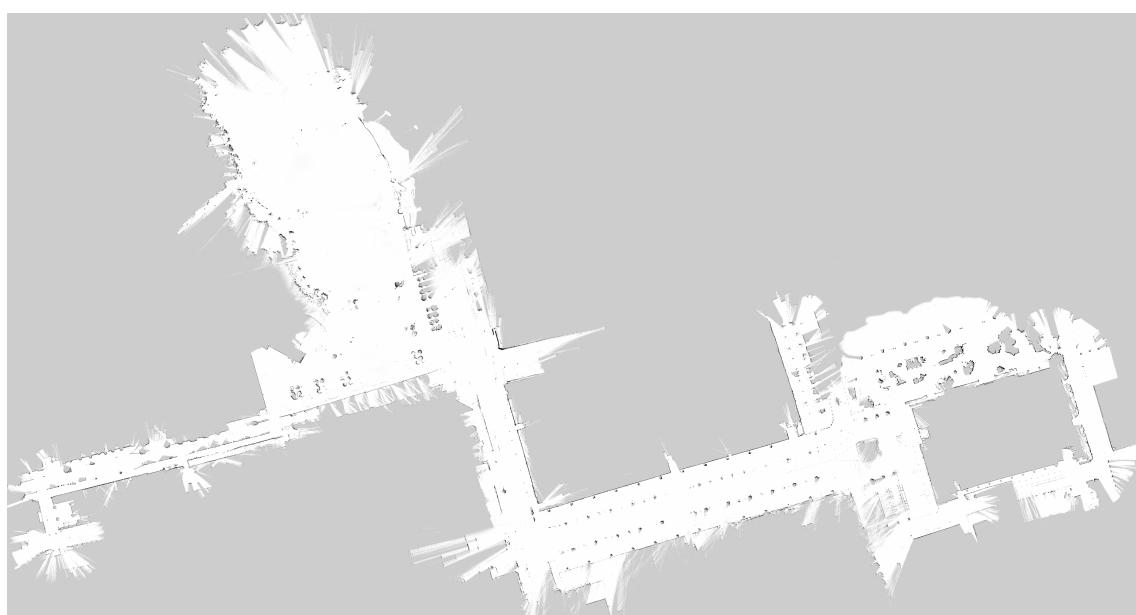


Fig. 3.14: Map before adjustment

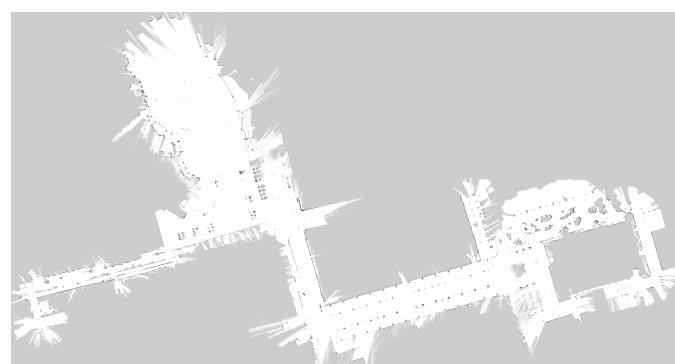


Fig. 3.15: Map after adjustment

第4章

津田沼チャレンジによるドキュメントの有用性の検証実験

4.1 実験概要

本研究で作成したドキュメントの有用性を確認するため、津田沼チャレンジ [9] のコースを用いて自律移動実験を行った。実験では、ドキュメントによる手順通りにナビゲーションのパラメータを調整されたロボットが、自律的に設定されたコースを完走できるかで、ドキュメントの有用性を確認することを目的とした。

検証のため以下の2つの実験を行った。

実験1 ドキュメント作成者が2025年度版コースを対象として行った有用性の検証

実験2 ROS初心者の学部3年生が2024年度版コースを対象として行った有用性の検証

それぞれのコースをFig. 4.1, Fig. 4.2に示す。実験1では、走行地図の作成から自己位置推定および経路計画に関する各種パラメータの調整までを全て行った。一方、実験2では、作成者がまとめたドキュメントを基に、作成者が作成した地図を使用してナビゲーションのパラメータ調整を行った。また、パラメータ調整に要した時間についても記録し、調整の負担を把握する参考とした。

実験には、本研究室で開発されているロボット ORNE-box2[10] を使用した。その外観を Fig. 4.3 に示す。また、ナビゲーションには ROS Navigation stack を用いた。ORNE-box2 を

含めたそのシステム構成を Fig. 4.4 に示す。



Fig. 4.1: Course map of the Tsudanuma Challenge 2025(souce: [9])



Fig. 4.2: Course map of the Tsudanuma Challenge 2024(souce: [9])



Fig. 4.3: ORNE-box2

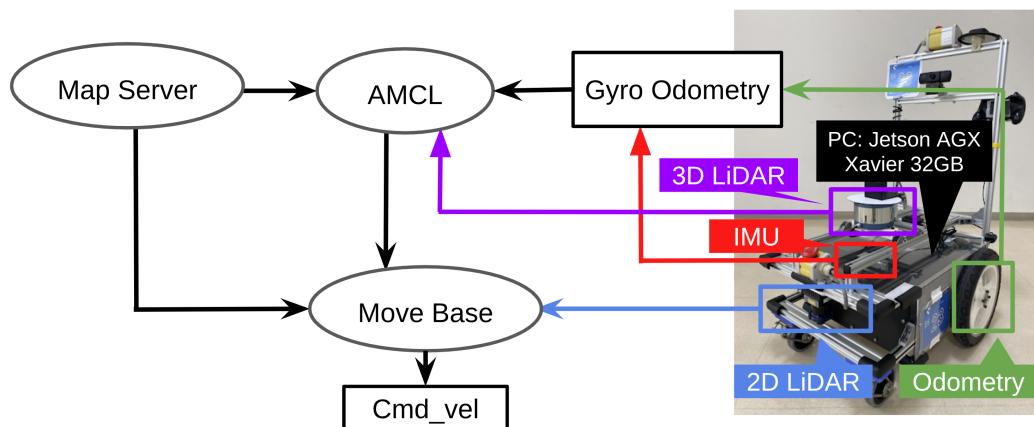


Fig. 4.4: System configuration

4.2 実験結果

4.2.1 実験 1

Table 4.1: Navigation results of the Tsudanuma Challenge 2025

Run type	Number of trials	Number of success
Preliminary run	10	10
Official run	1	1

Table 4.1 に示すように、調整後のシステムを用いて事前走行を 10 回実施した結果、すべての試行でゴールまでの完走を確認した。走行コースの全長は約 1799m、走行時間はおよそ 40 分であり、すべての走行で同区間を完走した。

さらに、2025 年 8 月 27 日に行われた津田沼チャレンジの本走行においてもコースを完走し、自己位置の破綻や経路追従の失敗は見られなかった。本走行での実験風景を Fig. 4.5 に示す。



Fig. 4.5: Experiment scene

4.2.2 実験 2

Table 4.2: Results of B3 students in the Tsudanuma Challenge 2024

	No.1	No.2	No.3
Completion	Yes	Yes	Yes
Total time to completion [h]	5.0	3.5	4.5

Table 4.2 に示すように、ドキュメントの手順に沿ってパラメータ調整を行った結果、3名ともコースの完走を達成することができた。走行コースの全長は約 1661m、走行時間はおよそ 35 分であり、すべての走行で同区間を完走した。いずれの実験においても、ロボットは経路から大きく逸脱することなく目標地点まで走行でき、安定したナビゲーション動作が確認された。また、パラメータ調整に要した時間は 3.5~5.0 時間であり、いずれの参加者も作業を完遂することができた。これにより、ドキュメントの有用性を数例ではあるが確認された。

第5章

つくばチャレンジによるドキュメントの有用性の検証実験

5.1 実験概要

本研究で作成したドキュメントの有効性を確認するため、つくばチャレンジ 2025[8] のコースを用いて自律移動実験を行った。実験では、ドキュメントによる手順通りにナビゲーションのパラメータを調整されたロボットが、自律的に設定されたコースを完走できるかで、ドキュメントの有用性を確認することを目的とした。そのコースを Fig. 5.1 に示す。実験では、作成者が走行地図の作成から自己位置推定および経路計画に関する各種パラメータの調整までを全て行った。

実験には、本研究室で開発されているロボット ORNE-box2 を使用した。また、ナビゲーションには ROS Navigation stack を用いた。

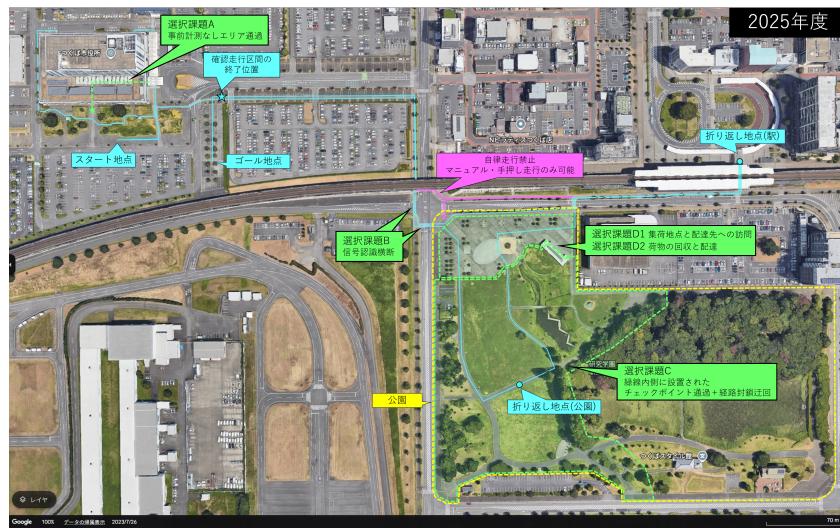


Fig. 5.1: Course map of the Tsukuba Challenge 2025(souce: [8])

5.2 実験結果

Table ??に示すように、

第6章

結論

本研究では、ROS ベースの自律移動ロボットのナビゲーションにおけるパラメータ調整手順を整理したドキュメントを作成し、その有効性を検証した。ドキュメントは ROS Navigation stack のパラメータを中心に構成されている。津田沼チャレンジの実験では、作成者自身および学部3年生のいずれもがそのコースを完走し、ドキュメントの有用性が確認された。また、つくばチャレンジ 2025において、

参考文献

- [1] Zheng Chen. Ros-based navigation and obstacle avoidance: A study of architectures, methods, and trends. <https://www.mdpi.com/3397394>. (Accessed on 10/25/2025).
- [2] ロボットオペレーティングシステム市場規模. <https://www.gminsights.com/ja/industry-analysis/robot-operating-system-market/>. (Accessed on 10/25/2025).
- [3] ros planning. Github - ros-planning/navigation: Ros navigation stack. code for finding where the robot is and how it can get somewhere else. <https://github.com/ros-planning/navigation>. (Accessed on 10/13/2025).
- [4] Kaiyu Zheng. Ros navigation tuning guide, 2017. 19 pages, 21 figures. Written in September 2016.
- [5] A guide to implement ros navigation stack on any robot. <https://prabhjotkaurgosal.com/>. (Accessed on 10/25/2025).
- [6] Google earth. <https://earth.google.com/web>. (Accessed on 11/12/2025).
- [7] navigation. <https://wiki.ros.org/navigation>. (Accessed on 9/22/2025).
- [8] つくばチャレンジ 2025. <https://tsukubachallenge.jp/2025/regulations/tasks>. (Accessed on 10/13/2025).
- [9] Tsudanuma challenge. <https://sites.google.com/p.chibakoudai.jp/rdc-lab/development/tsudanuma-challenge>. (Accessed on 10/13/2025).
- [10] 井口颯人, 樋高聖人, 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.

付録

本研究で作成したドキュメントは GitHub で公開している。

- ドキュメントのリンク: https://github.com/open-rdc/nav_tuning_guide

謝辞

本研究を進めるにあたり，1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。