2023 年度 卒業論文
走行ルート指定時の取り扱い改善による ロボットの自律走行の向上

奈良女子大学 生活環境学部 情報衣環境学科
生活情報通信科学コース 4 回生 新出研究室
20480215 岸上愛

2024 年 2 月

概要

近年、我々の生活においてロボットの活躍は様々な場面で顕著である。そこで本学並びに各地の大学や研究機関、企業においても自律走行型のロボットの研究や開発が盛んに行われている。特に屋外では、人の往来や物流などの動きが活発であり、ロボットは周囲の環境の変化に応じてスムーズな走行が必須である。我々が実験に使用している自律走行ロボットでは、従来、交通量の多い都会での走行実験において、走行ルートの変更が柔軟に行えないことが課題であった。そこで本研究では、走行ルートを決定する際に必須となるウェイポイントの取り扱いを改善し、ルートの変更を行いやすくした。また、改善したプログラムを実際の実験で使うことによりその有用性を示した。

目次

1　はじめに

自律走行ロボットが走行する際には、正確な自己位置の把握と、経由地の指定である地図上のウェイポイントの的確な配置が必須である。特に、地図の作成後に障害物の位置の変化などがあった場合、ウェイポイントを適切に再配置することが、正確な走行と実験の効率化の両面から必要である。そこで本研究では、地図上でのウェイポイントの指定を従来よりも柔軟に行えるように改善し、実際のロボットチャレンジ実験走行で改善したプログラムを使うことによりその有用性を示した。

2　使用したロボット Arno について

本研究では、北陽電機製作の自律走行型ロボット Arno を用いて実験を行った。前方に 3 次元測域センサを一台、前方・後方に 2 次元測域センサをそれぞれ一台ずつ搭載している。これらにより、ロボットの進行方向の高低差と周囲 360 ° を認識可能である。センサや走行の制御は、オープンソースで提供されているロボット操作のオペレーティングシステムである ROS[1] を用いて行っており、ロボットを手動で動かして地図を作成しウェイポイントを設定することでルートを指定し自律走行を行う。地図とセンサ情報をもとにロボットの走行をコントロールする既存の ROS ノードで動作している。

3　使用したプラットフォームについて

3.1　ROS

ROS(Robot Operating System) とは、ロボット操作のプログラムを開発するプラットフォームである。ROS によるロボット制御プログラムはノードと呼ばれており、それらが互いに通信し、情報やデータの交換を行う。モーターやセンサなどのロボットのハードウェアとの通信を、トピックという通信チャンネルを通したメッセージのやり取りとして、Python や C++ などのプログラミング言語から利用できる機能を提供する。トピックは、配信/購読 (publish/subscribe) 型の通信メカニズムを実装している。

3.2　Rviz

Rviz とは、ROS に付属する 3D ビジュアライザーである。Rviz を使うことで、ロボットやセンサーなどのデータを三次元空間または二次元平面で表示することができる。ロボットのシミュレーションや、実際にロボットを動かした際のデータを可視化することができるので、ロボットの開発に役立てられる。

3.3　ウェイポイント

ロボットが自律走行をする際にどの経路を走行するかという情報をロボットに与える必要があり、ROS におけるウェイポイントとはこの経路を走るための経由地の情報のことである。我々が参加したロボットチャレンジでも、自律走行を行うためにあらかじめウェイポイントを作成する。このウェイポイントによって、ある程度の経路を決定するが、動的な障害物に関しては回避が必要なため、ロボットが経路上を走るとは限らない。例えば、人や自転車やコーンなどの動的な障害物が経路上に現れた場合は経路の変更が必要になる。この

ような場合、従来のシステムではウェイポイントを記録したファイル中において修正したい座標のみを特定することがかなり困難だった ため、改めて出発地点からロボットを走らせ、ウェイポイントを取り直すことが事実上最善であった。しかし、この方法をとると走行実験の効率が低下する。特に、自律ロボットの走行大会である中之島チャレンジなどでは、実験に使える時間が限られるため、このことは大きな問題であった。

そこで経路の変更をより簡単に行うために ROS の可視化ツールである Rviz 上でウェイポイントの座標と番号を表示し、マウス操作で番号を指定するとターミナル上に座標が表示される機能を持つプログラムを作成した。これによって、ウェイポイントファイル中の修正箇所の特定を可能にし、容易に座標を修正出来るようにした。

4　実験による課題の発見

本節では、自律ロボットの走行大会である中之島ロボットチャレンジに我々が参加した経験によって明らかになった課題について述べる。中之島ロボットチャレンジは、大阪市中央公会堂で行われる本大会と、それ以外の会場で行われるエクストラチャレンジがあり、2023 年度は 10 月 14, 15 日に大工大エクストラチャレンジ、11 月 5, 25, 26 日に ATC エクストラチャレンジという日程で開催された。以降では、開催された順に述べる。

4.1　大工大エクストラチャレンジ

大阪工業大学の梅田キャンパスにて行われた。コースは比較的平坦でロボットにとっては易しいコースであった。ウェイポイントの取得自体はスムーズに行えたが、自律走行をさせてみると途中で自己位置を見失い、何度も同じ場所で回転し続ける現象が発生した。緊急停止ボタンを押してその地点から再走行を試みたが成功しなかった。そこで、ロボットが把握していた自己位置と指定していたウェイポイントとの誤差が大きかった点が課題であると考えた。

4.2　ATC エクストラチャレンジ

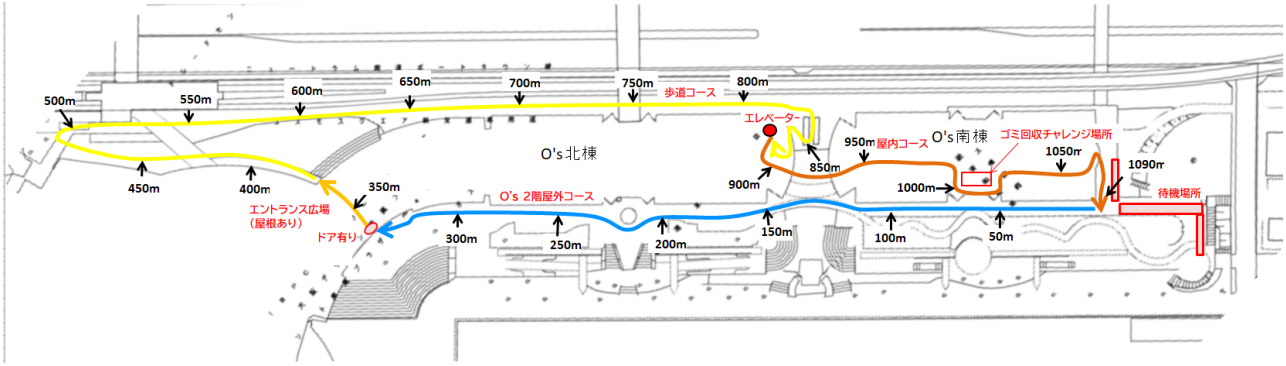

図 1　ATC においてのロボット走行経路

ATC(アジア太平洋トレードセンター) にて行われた。コース (図 1[3]) は障害物が多く坂道や階の変更など

も多くあり、完走するには試行錯誤を強いられる。実験走行日と本走行日の間にコースの途中でガラス扉の開閉の変更があり、ウェイポイントの打ち直しが必要だった。実験走行時には常時開いていたガラス扉のうち一部が本走行日には閉じており、実験走行時にその扉を通るルートでウェイポイントを設定していたため、本走行ににはウェイポイントを、当日開いている扉を経由するようにずらして打ち直す必要があった。その際、出発地点に戻って最初から打たなければならず、多くの時間と手間を要した。このことから、コースの途中のウェイポイントのみ座標の変更ができる方法が必要と考えられる。後の節で、この課題の解決について述べる。

4.3 中之島ロボットチャレンジ

大阪府の大阪市中央公会堂付近にて行われた。コース自体は平坦な道ではあるけれども、センサーにとって目印となる柱や木などの大きな静的な障害物が少ないため、自己位置を見失うことが数回あった。また地図作成時には、実際の土地には障害物はないところでセンサーが障害物判定をしてしまうことが多く、手動で地図上の障害物を消す作業が必要となった。そこで、地図作成後の修正の簡易化が課題であると考えた。

5 実装

本節では、4 節で述べた課題点のうち、本研究では 4.2 で述べた、コース上のウェイポイントの変更を容易に行うことを課題とした。そこでウェイポイントの経由地変更を従来よりも簡単に行うために作成したプログラムについて述べる。

5.1 データ

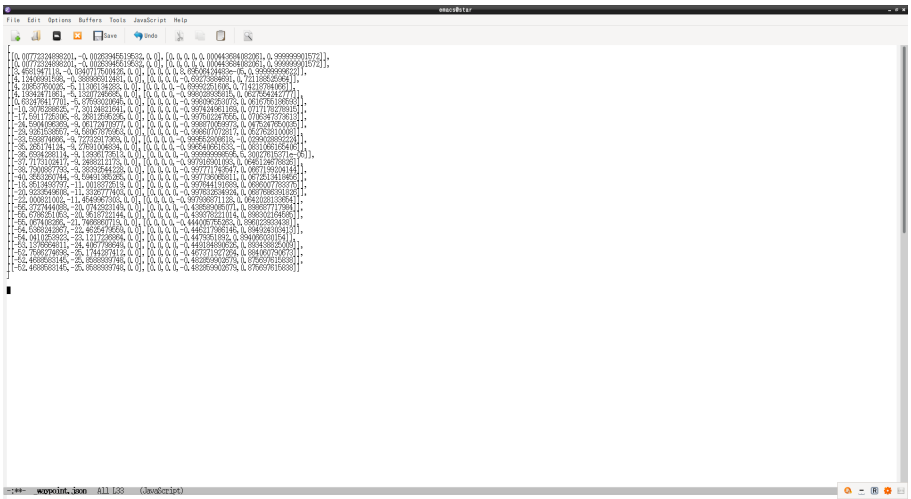

図 2 従来のウェイポイント json ファイル

ウェイポイントの json ファイルは、図 2 のように座標を数値で記載したものである。これではウェイポイントをどこにずらしたり、どこに追加したりすればよいのかが人間にはわからないため、修正が容易にできない問題がある。

5.2 プログラムと処理

5.1 節に述べた問題を解決するため、以下の 2 つの機能を持つプログラムを作成した。

- ウェイポイントの座標を番号を振った上でを Rviz の地図上に表示
- Rviz のマウスで指定した番号の座標をターミナル上に表示

作成したプログラムを以下に示す。開発環境の都合上、Python2 での実装になっている。

```
#!/usr/bin/env python2

import rospy
import json
from visualization_msgs.msg import Marker
from move_base_msgs.msg import MoveBaseActionGoal

def callback(data):
    pos = data.goal.target_pose.pose
    print ("[{0},{1},0.0],[0.0,0.0,{2},{3}]".format(pos.position.x,pos.position.y,
pos.orientation.z,pos.orientation.w))



rospy.init_node("waypoint_manager")

pub = rospy.Publisher("waypoint", Marker, queue_size = 10)
rospy.Subscriber("/move_base/goal",  MoveBaseActionGoal, callback)

rate = rospy.Rate(25)



while not rospy.is_shutdown():
    with open('waypoint.json', 'r') as f:
        counter = 0
        loader = json.load(f)

        for row in loader:
            # Mark arrow
            marker_data = Marker()
            marker_data.header.frame_id = "map"
            marker_data.header.stamp = rospy.Time.now()
```

```
marker_data.ns = "basic_shapes"
marker_data.id = counter

marker_data.action = Marker.ADD

marker_data.pose.position.x = row[0][0]
marker_data.pose.position.y = row[0][1]
marker_data.pose.position.z = row[0][2]

marker_data.pose.orientation.x=row[1][0]
marker_data.pose.orientation.y=row[1][1]
marker_data.pose.orientation.z=row[1][2]
marker_data.pose.orientation.w=row[1][3]

marker_data.color.r = 1.0
marker_data.color.g = 0.0
marker_data.color.b = 0.0
marker_data.color.a = 1.0
marker_data.scale.x = 1
marker_data.scale.y = 0.1
marker_data.scale.z = 0.1

marker_data.lifetime = rospy.Duration()

marker_data.type = 0

pub.publish(marker_data)
counter +=1


# Mark number
marker_data = Marker()
marker_data.header.frame_id = "map"
marker_data.header.stamp = rospy.Time.now()

marker_data.ns = "basic_shapes"
marker_data.id = counter

marker_data.action = Marker.ADD
```

```
            marker_data.pose.position.x = row[0][0]
            marker_data.pose.position.y = row[0][1]
            marker_data.pose.position.z = row[0][2]

            marker_data.pose.orientation.x=row[1][0]
            marker_data.pose.orientation.y=row[1][1]
            marker_data.pose.orientation.z=row[1][2]
            marker_data.pose.orientation.w=row[1][3]

            marker_data.color.r = 0.0
            marker_data.color.g = 0.0
            marker_data.color.b = 0.0
            marker_data.color.a = 1.0
            marker_data.scale.x = 1
            marker_data.scale.y = 0.1
            marker_data.scale.z = 0.1

            marker_data.lifetime = rospy.Duration()

            marker_data.type = Marker.TEXT_VIEW_FACING
            marker_data.text = str(counter)

            pub.publish(marker_data)
            counter +=1

    rate.sleep()

rospy.spin()
```

このプログラムでの処理を述べる。Marker 型のデータ marker_data に方向と座標の値を設定し、ROS トピック waypoint に publish することで Rviz 上に json ファイルの座標と番号が反映される仕組みとなっている。また、callback 関数を用いて ROS トピック/move_base/goal から購読した座標を print させることで、指定した座標をターミナルに表示する。

6 検証

6.1 Rviz

本プログラムと Rviz を起動させると、Rviz の地図上に指定したウェイポイントファイルの座標位置と番号、方向が反映された (図 3)。

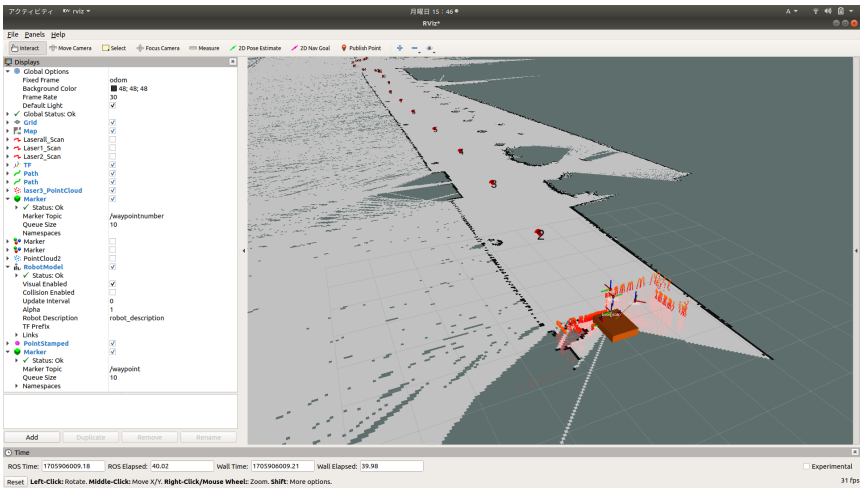

図 3　地図上に反映されたウェイポイント

6.2　ターミナル

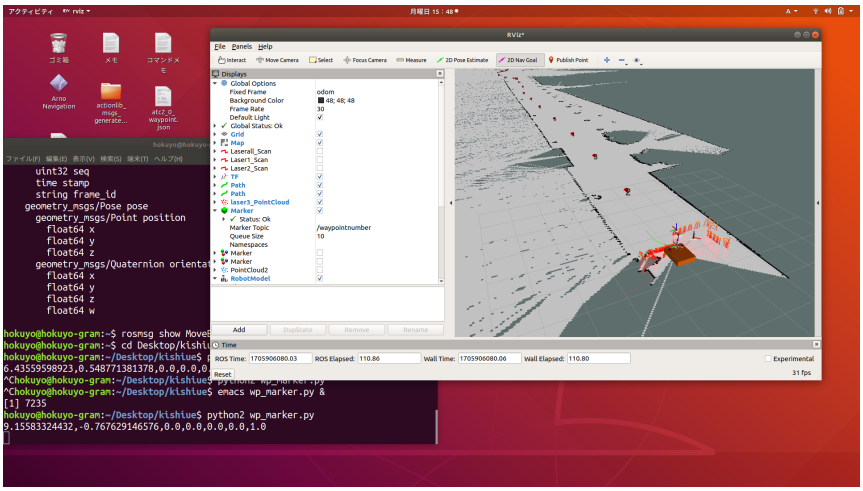

図 4　ターミナル上に表示された座標

マウス (2D Nav Goal[2]) で修正したい箇所や追加したい箇所を指定すると、指定した座標 (図 4) のみがターミナルに表示された。人間はこの出力により、json で記述されたウェイポイントファイルのどこを修正すべきか、またどのような値に修正すべきかを知ることができる。

7　結果

従来の走行システム単体では変更したいウェイポイントの位置が分からなかったが、Rviz 上にウェイポイント毎の座標の番号とロボットの方向を反映するという処理を実装したことにより、ウェイポイントの位置を画面上で判断できるようになった。また、2D Nav Goal で修正したい箇所や追加したい箇所を指定すると指

定した座標のみがターミナルに表示される処理を実装したことにより、waypoint ファイルにターミナルに出てきた座標を追加するだけで修正が終わるので、修正を座標の計算から手動で行っていた従来よりも修正方法を格段に簡易化することができた。

8 まとめ

ロボットの自律走行を行う場合にウェイポイントの座標変更を余儀なくされたとき、従来では全て手動で行っていたのに対して、変更する座標の確定を簡単に行なえるようになった。これにより、ロボットの自律走行が途中で失敗する可能性を大きく減らすことができた。

今後の課題としては 2 点挙げられる。ウェイポイントの変更操作は現在は手動でファイルを修正して行っているが、これを Rviz 上で完結出来るとさらに修正時間を短縮できるので、マウス操作によるウェイポイントの直接修正や加筆修正も可能にしていきたい。また、前述 (4.3 節 実験による課題点の発見) の通りセンサーの障害物誤判定によって出現した地図上の障害物を消去する際、修正方法の自動化ができれば自律走行の準備において効率が大幅に向上するはずである。

9 謝辞

本研究の遂行にあたり、指導教官として終始丁寧なご指導を賜った、新出尚之准教授に深謝致します。並びに中之島ロボットチャレンジにご協力いただいた北陽電機の皆様、また新出研究室の先輩方には多大なご助言、ご協力を頂きました。心より感謝申し上げます。

参考文献

[1] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, 2015. (河田 卓志 (監訳) and others. プログラミング ROS ー Python によるロボットアプリケーション開発, オライリージャパン, 2017).

[2] ロボット理工学科演習. プログラムからナビゲーションを実行する方法 (python 版). `https://robot.isc.chubu.ac.jp/?p=1626`. Robotics laboratory(2024 年 1 月 20 日閲覧).

[3] 中之島ロボットチャレンジ実行委員会. 中之島ロボットチャレンジ 2023. `https://www.nakanoshima-rc.jp`. Nakanoshima Robot challenge (2024 年 1 月 20 日閲覧).