2021 年度　卒業論文
ロボットによる自律走行の安定性向上を支援する技術開発と走行環境の違いによる成功条件の比較

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
18480261 木下美咲

2022 年 2 月 9 日

概要

我々はロボットに自律走行をさせる際に、従来より容易な操作で、安定した走行を得るためのプログラムを開発し、これを用いて自律走行の性能向上を示した。

ロボットの自律走行には地図、ウェイポイント（ゴールを含む通過点）、ナビゲーションシステムの 3 要素が必要である。さらに、それらの要素を揃えた上で、事前トレーニングを繰り返すことで自律走行が可能となる。本研究で使用したロボットでは、自律走行を行う際に、すでに 3 要素の構築が可能であったが、事前トレーニングを何度も繰り返す必要があり、多くの手間がかかっていた。また、事前トレーニングの操作の大半は人間によって行われ、複雑な作業も多かった。その結果、操作ミスが発生し、自律走行の失敗の原因となっていた。そこで、いくつかの技術を開発し、同じ作業を何度も繰り返すことなく、安定した走行を実現できる環境を目指した。

また、開発した技術による自律走行を試すために、様々な環境で実験を行った。奈良女子大学における実験では、屋内と屋外という 2 つの環境での比較を行い、安定した自律走行に必要な条件が異なることを実証した。さらに、公道や公園など、より複雑な環境で開催される、ロボットコンテストに参加し、参加日程のみのトレーニングで、参加チームの中でも好成績を残すことができた。

目次

1　はじめに

現在、我々の暮らしには自動化された機械が必要不可欠である。例えば、自動ドアや信号、また家庭内ではエアコンや食洗機などである。さらに近年では、自動化を発展させた「自律化」が登場している。

「自動化」から「自律化」へとステージが移る背景には、自動化の限界があげられる。そもそも、自動化では、製造業において人間が行っていた単純作業の効率化を目的としていた。そこで用いられるロボットは人間の経験によって得られたルールを元に定義され、そのルールの範囲内のみで稼働する。よって予め定義されていない事態に対応できない。例えば、人間によって定義されていない予期せぬ出来事が起こったときにロボットがどのように動作するかはわからない。ましてや、うまくその出来事に対処するというのは「自動化」の段階では不可能なのである。これこそが自動化の限界である。そこで登場したのが、人間の定義したルールの範囲外でも、ロボット自身が経験や周りの環境から情報を得て、どのように行動するべきかを思考し、対処することができる「自律化」である。

「自動化」が現在の私達の生活において、必要不可欠な技術となっているように、「自律化」も私達の生活に必要不可欠な技術となると予想される。本研究では、ロボットが活動する上で基本となる移動の自律化、「自律走行」に焦点をあて、技術開発を行った。その技術を用いた走行実験を学内で行い、さらなる改良を加えた。さらに、自律走行をメインとしたロボットコンテストに参加し、改良を行った技術がより現実に近い環境でも有効であるかを検証した。その結果、本研究で使用したロボットが昨年参加した時に課題となっていた、横断歩道の通過に成功した。また、ロボットコンテストの発展チャレンジとして開催されたコンテストにも参加した。こちらのコンテストは、公園内で目印が少なく、全チームが初めての環境という条件で実験が行われたが、我々のチームは全参加チームの中でも好成績を残すことができた。

2　ロボットについて

2.1　Arno(アルノー)

本研究では北陽電機が開発した自律走行ロボット「Arno」(図 1) を使用する。

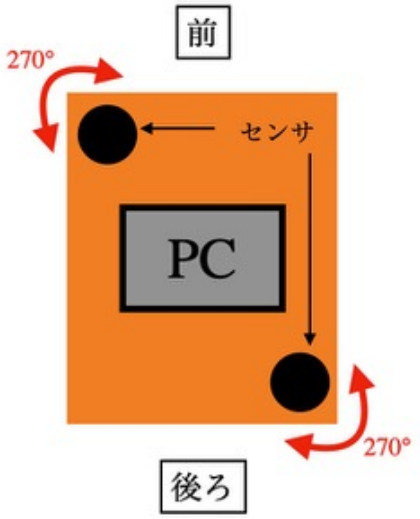
19V のバッテリーを 2 つ搭載し、PC、モーター、センサの動力としている。センサについては 2.2「センサ」で詳しく述べる。Arno では ROS を用いて、センサからの情報を処理し、モータの制御を行うことで、自律走行を実現している。ROS については 2.3 章「ROS」、処理を行っているプログラムについては 3 章「実装」に詳細を記述する。

2.2　センサ

北陽電機の測域センサ URM-40LC/LCN-EW(図 2(a)) を使用する。レーザースキャナであり、距離 40m(本研究では 20m に設定)、270 ° の範囲が計測可能である。Arno には 2 台のセンサを機体の左前と右後ろにそれぞれ 1 つずつ設置している (図 2(b))。これにより、機体の全方位の環境をスキャン可能である。

図 1: Arno

(a) 測域センサ URM-40LC/LCN-EW



(b) センサの取り付け位置

図 2: Arno に取り付けられたセンサ

2.3 ROS

ROS(Robot Operating System)[1] はロボット用プラットフォームやソフトウェア開発を支援するライブラリツールを備えたソフトウェアフレームワークである。フリーソフトウェアであり、共有と配布が自由かつ容易に行えるため、世界中の開発者が作成したソフトウェアを自身のロボットに利用することができる。ま

た、自身の作成したプログラムを公開することで、さらなる改善の提案などを受けることができる。

2.3.1 ノード

ROS では、ノードと呼ばれる実行ファイルを用いる。ノード間で情報を購読、配信することで、センサの情報やモータの制御信号などの伝達を行い、これによってロボットの動作を制御している。図 3 に、Arno で用いられている各ノードとその間の情報のやり取りを示す。

本研究では、従来手動で行っていたウェイポイント指定と地図編集を自動で行う新たなノードを作成した。さらに、これまで使用していたナビゲーションを行うノードに、

- 地図を自動で切り替える機能の追加
- ロボットの自己位置が異なる場所に飛んでしまう問題が起こらないように、ノード内におけるウェイポイントの扱いに対する工夫

の 2 点の改良を加えた、新たなノードを用いて走行実験を行った。

2.3.2 パラメータ

パラメータはロボットを起動させるためにノード内で使用される様々な値である。ノードとは別にパラメータファイルを作成することで、ノード内を書き換えることなく、条件を変更してロボットを起動させることが可能である。従来のプログラムでは、自律走行時の速度やウェイポイントに到達したとみなすトレランスといったパラメータに加え、ノード内で用いられるロボットセンサの位置やタイヤの大きさといったロボットに関する細かな値は、パラメータファイルで設定していた。パラメータファイルは、このような変更が容易にでき便利なものであるが、書き換えには注意が必要である。例えば、自律走行時の速度を変更する際に、単に速度の項目を書き換えるだけでは自律走行は上手くいかない。加速度など速度に関連する値も一緒に調整しなければならない。そして、速度が設定した値まで上がりきっているか、走行時に真っ直ぐに走行できているかなど、不安定な走行でないかを実際に動かして確認する。そして、必要であればさらに調整するという作業を繰り返さなければなければならない。この繰り返しの作業をなくすために、新しく作成したノードで速度を制御できるようにした。このノードによる速度制御方法は、3.2「ウェイポイント指定ノード」に詳しく記載する。

2.4 自律走行

2.4.1 自律走行とは

自律走行とは、ロボットが人間の操作を受けずに走行を行うことである。ロボット自身が、地図と自己位置をマッチングして経路を作成し、周囲の環境を認識することで障害物を避けながら、ゴール到達を目指す。つまり、自律走行には、地図、経路生成のための目印（ウェイポイント）、ナビゲーションシステムの 3 要素が必要である。

本研究における自律走行では、SLAM（Simultaneous Localization and Mapping）[2] という手法を用いる。SLAM は自己位置推定、地図とのマッチング、経路生成による自律走行を行っている。

Arno での自律走行の詳しい手順は 2.4.2 章に記す。

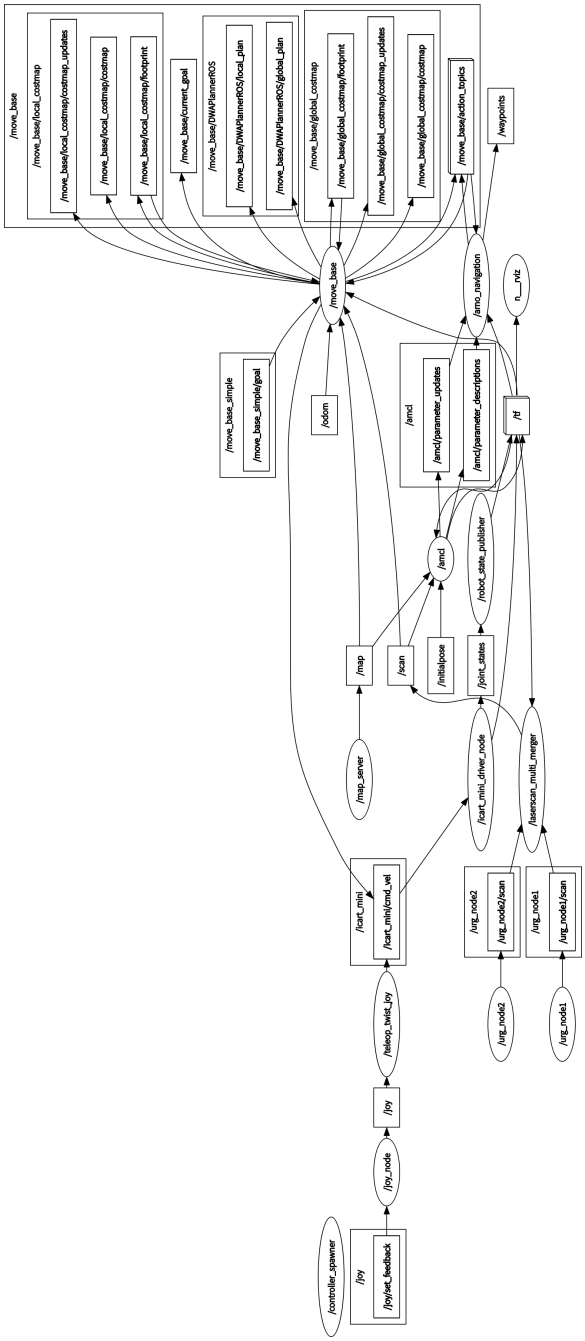


図 3: ノード間で情報が行き来している様子
（丸がノード、四角形が情報を表す）

2.4.2 自律走行の手順

1. 地図作成

   自律走行をスタートさせたい地点で地図作成ノードを起動し、Arno に有線接続されたゲームコントローラで操作し、環境内を移動させる。PC の画面に地図が表示されていくので、作成したい範囲の地

図が出来上がっているのを確認したら、ノードを終了させる。

2. ウェイポイント指定
   作成した地図を、ROS で付属の可視化ツール rviz で表示し、スタートからゴールまでの間の Arno に通過させたい地点をウェイポイントとして指定する。最後に指定したウェイポイントがゴールとなる。

3. ナビゲーション
   ナビゲーションを行うノードに、地図とウェイポイントの座標が書かれたウェイポイントファイルを与えることで、地図と自己位置のマッチングが行われる。ロボットはウェイポイントファイルに書かれたウェイポイントを経路の順にたどり、最終的にゴールに到達する。

以上、3 過程がそれぞれ自律走行のための要素となる。

3 実装

本節では、2.4.2 で述べた 3 要素ごとに従来システムの問題点を挙げ、それらを解決するために行った新たなノードの作成と従来ノードの改良について述べる。

まず、地図では自律走行で使用する前に編集を行うノードを、ウェイポイント指定では、ウェイポイントの指定を自動で行うノードを作成した。次に、ナビゲーションシステムでは従来ノードに改良を加えたノードを用意した。

3.1 地図編集ノード

自律走行で地図を使用するときに起こる問題として、地図上で道以外の場所（実際の環境ではロボットが走行できない場所）にウェイポイントを指定してしまうことが挙げられる。ロボットは指定されたウェイポイントの座標をたどることでゴールまで到達するので、ウェイポイントの指定を誤ると、道から外れてしまったり、障害物に衝突し、自律走行が失敗する原因となる。この問題は、センサの取り付け位置と人間の操作ミスによって引き起こされる。センサの取り付け位置より低い位置にある障害物（花壇や下りの階段など）は地図に反映されない。このため、人間がウェイポイントを指定するときに、障害物があることが、地図を見ただけではわからない。また、地図は白、黒、灰色で構成されており、かつ、二次元の地図なので、人間が本来の環境と比較して、道の上に正確にウェイポイントを指定するのは困難である。

さらに、他の問題として地図作成時と自律走行時の環境の変化への対応が挙げられる。たとえば、ロボットのオペレーターの軌跡や、地図作成時に偶然存在していた車など、自律走行時には存在しない可能性が高いものが地図上に障害物として記録され、ロボットはそれらを避けるような走行ルートを作成してしまう。その結果、無駄な動作が増えたり、自己位置を推定する際に本来とは異なる場所と認識してしまうこともある。この問題は、従来は画像編集ソフトを用いて手書きで地図を修正し、解決していた。しかし、地図のサイズが大きくなるにつれ、修正箇所が増加するうえに、小さな障害物を見落としてしまうなどの問題も発生する。

そこで、地図編集ノードで地図を選択すると、不要な障害物の削除など、地図の手直しを自動で行えるようにした。

- 概要
  地図編集ノードは、自律走行時にロボットが使用する地図を改造するプログラムである。
  編集したい地図を rviz で開きながら地図編集ノードを動かすと、地図のデータをノードが受け取り、そのデータに手を加えて新しい地図を作成する。このノードでは、地図のコピー、障害物部分の強調、人間の軌跡の削除の 3 点を行うことができる。さらに編集前の地図に上書きを行うのではなく、新たに地図を作成するため、編集前と後の地図を比較することが可能である。

- アルゴリズム
  地図編集ノードは地図から以下の情報を受け取る。
  - width(地図の幅)
  - height(地図の高さ)
  - data(地図グリッド 1 マスの占有情報）[free(障害物なし):0, occupancy(障害物あり):100, unknown(不明):−1])

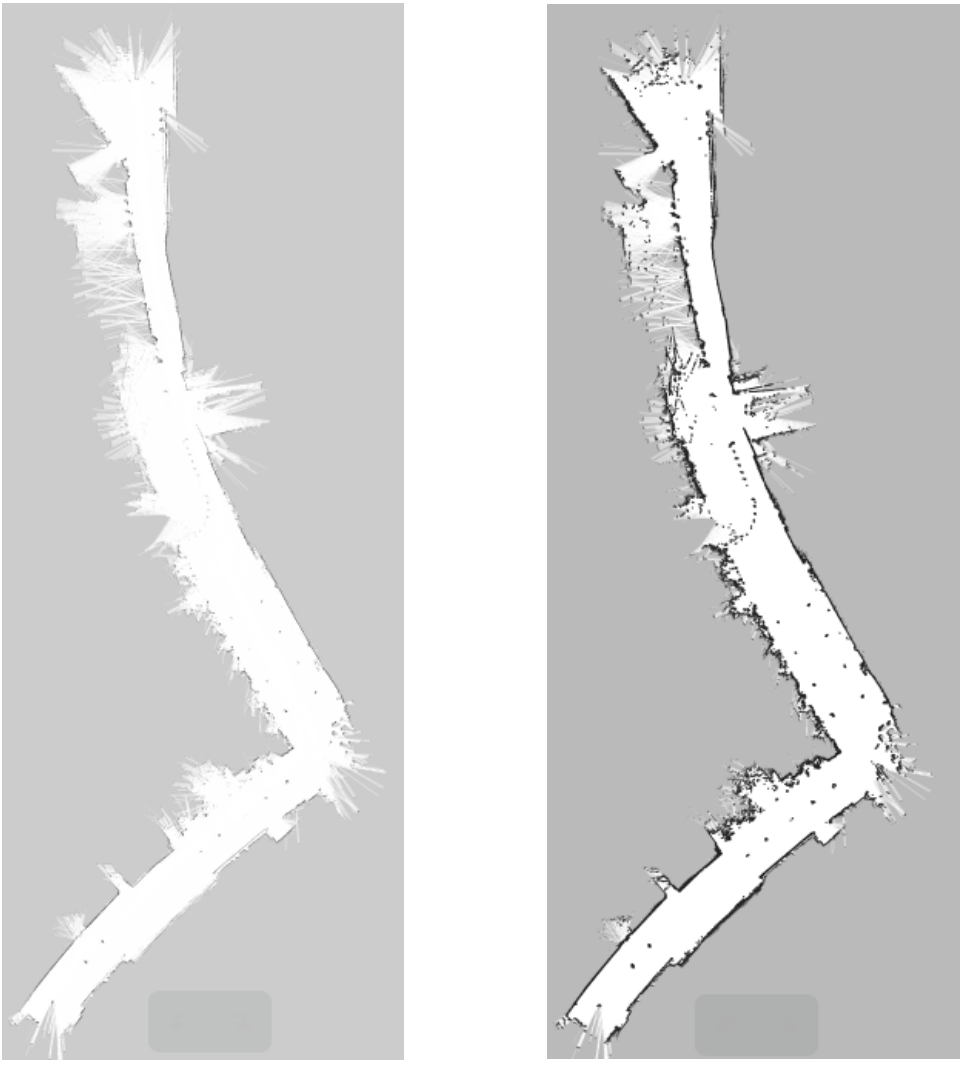
  以上の 3 つの情報から、現在の地図の状態を取得し、指定された地図編集の作業を行った新しい地図の情報を作成する。そして、新たな地図の情報から pgm ファイルと yaml ファイルを作成することで、新しい地図が完成する (図 4)。

3.2 ウェイポイント指定ノード

ウェイポイントとは、スタートからゴールまでの経路をロボットが作成する際に参考とする複数の地点であり、座標で表される。従来のウェイポイント指定は、rviz で地図を開き、人間が地図上のウェイポイントを作成したい地点をクリックするというように手動で行っていた。しかし、道ではない地点をウェイポイントと指定してしまったり、指定するウェイポイントが少なすぎたりなど、人間が指定を行うことで様々な問題が起こり、自律走行が上手くいくまで何度もウェイポイントの指定を繰り返す必要があった。さらに、従来のノードは PC をロボットに接続して使用することを前提として作られていた。なので、ロボットなしでウェイポイントの指定を行った場合、すべての座標がずれるという問題点があった。そこで、それらの問題を解決する 2 つのノードを開発した。1 つ目をウェイポイント指定ノード A、2 つ目を同 B と呼ぶ。まず、ノード A はロボットが通過した地点にウェイポイントの指定を自動で行うノードである。また、ノード B も自動でウェイポイント指定を行うが、その際にロボットがなくても使用することが可能である。しかし、ノード B は通過した地点ではなく、ロボットが通過できると推定される地点にウェイポイントを指定するのでノード A より自律走行の成功率は低いと考えられる。そこで、4 章走行実験では、2 つのウェイポイント指定ノードの比較を行う。以下、各ノードについて説明し、また、図 5 にノード B によるウェイポイント指定の様子を示す。

3.2.1 ウェイポイント指定ノード A

- 概要
  ウェイポイント指定ノード A は作成した地図を rviz で開きながら、自律走行させたいルートを人間がロボットを操縦し、もう一度たどることでウェイポイントの指定を行う、教示再生走行方式を採用した。スタート、ゴールとその間の複数の地点がウェイポイントとしてファイルに自動で記録される。



(a) 編集前の地図　　**(b)** 編集後の地図

図 4: 地図編集ノードで障害物部分を強調
(地図の白が free、黒が occupancy、灰色が unknown である)

- アルゴリズム
  ノードを起動させた地点をスタート、ノードを終了させた地点をゴールとする。一定間隔でのウェイポイント指定を行うために、ウェイポイントの座標間の距離とロボットの向き（角度）を用いる。ロボットを走らせると、スタート地点から現在地までの距離か角度があらかじめ定めておいた値以上になった場合に、自動でその地点の座標をウェイポイントとして記録する。そして、その地点を次のウェイポイント指定のスタート地点とし、次のウェイポイント指定を行う。

- オプション

  様々な環境で自律走行を行った中で、屋外や屋内など、環境の違いによってウェイポイント作成の間隔を変更した方が、成功する可能性が高くなることがわかった。そこで、ノードを起動した際に、ウェイポイント指定の間隔を屋外環境 (out)、屋内環境 (in)、オリジナル設定 (self) の 3 つから選択できるようにした。

  また、先述の通り、自律走行時のロボットの速度指定はパラメータファイルで行っているが、環境によって自律走行が安定する速度が異なるので、従来は環境が変わるごとにパラメータファイルの書き換えを行っていた。その際に、単に速度を書き換えるのではなく、加速度などその他要素も書き換え、走行させてみる、という一連の流れを安定した走行となるまで繰り返していた。

  しかし、ウェイポイント指定ノード A を使用することでパラメータファイルの数値の調整を繰り返し行うことなく、安定した走行が得られる。例えば、屋内環境で速度を抑えたいときはウェイポイントの間隔を狭くする。ロボットはウェイポイントに到達した時に一時停止するので、速度が上がり切る前にウェイポイントに到達し、次のウェイポイントに向かう際にはまた速度は 0 からとなる。このようにして、パラメータファイルに書かれた速度まで上がらないようにする。反対に屋外環境ではウェイポイント指定の間隔を広くすることで速度を上げることができる。

3.2.2 ウェイポイント指定ノード B

- 概要

  ウェイポイント指定ノード B は、地図上で通れると判断できる地点から自動でウェイポイントを選び、それをつないでスタートからゴールまでの経路を構成するというもので、予め地図を用意しておけば、再度ロボットを動かすことなく自動でウェイポイント指定を行うことができる。また、ノードを起動する際にロボットと接続していても、接続していなくても使用することができる。ウェイポイント指定ノード A と同様に作成した地図を rviz で開き、スタートとゴールを指定するとその間のウェイポイント指定が自動で行われる。さらに、スタートとゴールの間にさらにポイントを指定すると、そのポイント (以下、パスポイント) を経由するような経路となるようにウェイポイント指定が行われる。

- アルゴリズム

  ロボットから一定の間隔の複数の地点をウェイポイントの候補地とする。スタートから 1 番初めのパスポイント（又は、ゴール）までの距離が最短となるように、候補地の中からウェイポイントを選択していくが、その際に候補地の状態も考慮する。候補地の状態とは、その候補地がロボットが通ることができる地点であるかを表しているものである。これは、3.1 章「地図編集ノード」で取得する 3 つの情報のうちの 1 つである data(地図グリッド 1 マスの占有情報) から取得する。この情報によって、ウェイポイントの候補地が地図上に存在し、かつ、障害物がない場合にのみ、ウェイポイントを指定している。しかし、候補地の 1 地点のみの状態を考慮するだけでは、範囲が狭すぎたり、地図作成時の失敗でたまたま 0 という数値になっている可能性があるので、候補地を中心とした一定の範囲をグリッドとし、グリッドの範囲内の地図の状態が占める割合によってウェイポイントの指定を行うかを判断する。そして、指定したウェイポイントの座標と 1 番初めのパスポイントの座標の距離が予め定めておいた値よりも小さくなった場合に、1 番初めのパスポイントを新たなスタートとして設定し、次のパスポイントまでのウェイポイント指定を行う。以上の流れを最後に指定したウェイポイントの座標とゴールの座

標が、予め定めておいた値より、小さくなったと判断されるまで行う。

- オプション
  ウェイポイント指定ノード A と同様に環境の違いに合わせて、様々な値の設定が可能である。スタート、ゴール、パスポイントの指定は先述の通り、rviz 上で行うが、事前に複数のポイントの座標が書かれたファイルを用意しておき、そのファイルを読み見込ませることで、設定を行うこともできる。その場合、ファイルに書かれた 1 番初めの座標がスタート、1 番最後の座標がゴール、それ以外はパスポイントとなる。

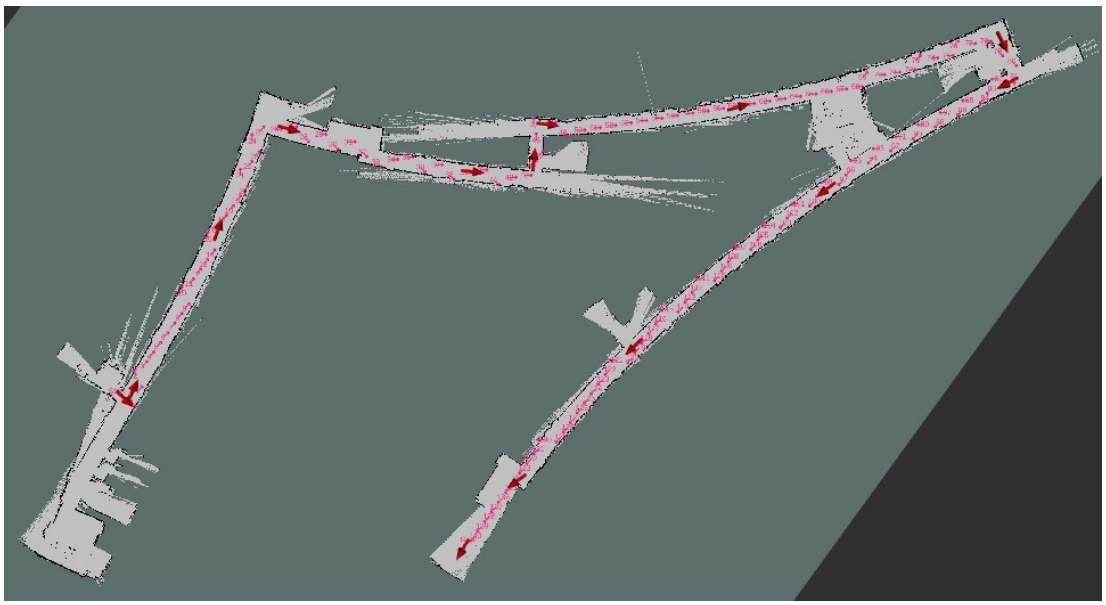


図 5: ウェイポイント指定ノード B によるウェイポイント指定
（赤矢印がパスポイント、ピンク矢印がウェイポイント）

3.3 ナビゲーションノード

ナビゲーションノードは予め作成した地図とウェイポイントの座標が書かれたファイル（以下、ウェイポイントファイル）を用いて自律走行を行うノードである。地図とウェイポイントファイルを指定して、ナビゲーションノードを起動すると、ウェイポイントファイルの座標が先頭から 1 つ読み込まれ、ロボットの現在地の座標との距離が計算される。そして、その距離が予め定めたトレランスの値より小さい場合に、ウェイポイントに到達した判断し、次のウェイポイントの座標が読み込まれる。この作業をすべてのウェイポイントの座標に対して行うことで、ゴールに到達させる。

従来のナビゲーションノードでも自律走行は可能であったが、以下のような問題があった。

1. 推定したロボットの自己位置が急に違う場所に飛んでしまう
   →すでに通過したはずのウェイポイントまでもどってしまい、また同じウェイポイントに到達しようと

する。さらに、先のウェイポイントを複数飛ばして、ゴール近くまで飛んでしまうこともある。

2. 自律走行が長距離になるほど自己位置がずれやすくなる
   →ロボットにはセンサを 2 つ取り付けているが、2 つのセンサからの情報をマージして自己位置推定などを行っている。そのため、長距離の走行では誤差が蓄積し、自己位置がずれてしまう。

3. 地図が歪んでいることにより、マッチングが困難である
   →センサの誤差蓄積により、地図が歪み、ロボットが本来の環境とのマッチングを行うことができない。
   そこで、これらの問題を解決するために、以下に述べるような改良を行ったノードを作成した。

- 概要
  従来のナビゲーションノードでは、1 枚の地図と 1 つのウェイポイントファイルを使用していたが、改良を行ったナビゲーションノードでは、複数の地図と地図 1 枚あたり 1 つのウェイポイントファイルを使用する。複数の地図は、地図作成時に地図を分割することで用意する。また、ウェイポイント指定時にそれぞれの地図に対応するウェイポイントファイルを作成する。そして、自律走行時にはそれらの地図を自動で切り替えながら走行を行う。
  これにより、まず、地図の分割によって地図と自己位置をマッチングする範囲が狭まるため、1 つめの問題点である、自己位置が違う場所に飛んでしまい、自律走行ができなくなる問題は起きにくくなる。また、もし自己位置が飛んでしまっても、大きなズレにはならないため、自律走行を続けることが可能である。さらに、センサ誤差の蓄積は新たな地図を開くごとにリセットされるので、2 つ目の問題点である長距離走行による自己位置のズレの発生を抑えることができる。また、3 つ目の問題点である地図の歪みやマッチングの失敗についても同様に、地図作成ごとにセンサの誤差蓄積がリセットされるので、起こる確率が低くなる。

- 地図の切り替え
  まず、地図作成時にファイル名を「1 枚目の地図目_通し番号」と作成しておく。例えば、1 枚目の地図が「tizu.yaml」という名前ならば、2 枚目以降は「tizu_0.yaml」、「tizu_1.yaml」…、のように続ける。また、ウェイポイントファイルは地図ファイル名に「waypoint.json」をつけた「tizu_waypoint.json」のようなファイル名とする。
  ナビゲーションノードを起動するには、コマンドライン引数に 1 枚目の地図名を指定する。地図名からウェイポイントファイルが検索され、自律走行がスタートする。そして、1 枚目の地図のゴール（1 つ目のウェイポイントファイルの最後の行のウェイポイントの座標）に到達したとノードが判断した時に、次の地図とウェイポイントファイルが読み込まれる。通し番号「0」から順に地図とウェイポイントファイルが検索され、該当するものがない場合に最終的なゴールに到達したとみなされ、ナビゲーションが終了する。このようにして、人間が操作をすることなく、自動で切り替えが行われていく。

- その他改良点
  ナビゲーションに使用するウェイポイントはウェイポイントファイルから読み出されて Python のリストに格納され、rviz で開いている地図上に表示されている。従来は到達したウェイポイントはリストに残されたままで、地図上にも表示されていた。しかし、改良したノードでは到達したとみなされたウェ

イポイントはリストから削除されるように変更を加えた。これにより自己位置推定が失敗した際に、前のウェイポイントに戻ってしまうという問題の発生を防ぎ、人間が rviz の画面を見た時に現在、どのウェイポイントを目指しているかを瞬時に把握できるようにした。

4 実験走行

実験走行は、環境の異なる 3 箇所で行った。4.1 節では奈良女子大学内の屋内と屋外の 2 つの環境で行った実験について、4.2 節では「中之島チャレンジ」で行った、人通りが多い中央公会堂の周辺の公道での実験について、また 4.3 節では「中之島エクストラチャレンジ」で行った、目印となる建物がない八幡屋公園内での自律走行の実験について述べる。

4.1 奈良女子大学

奈良女子大学内では、屋外と屋内環境によって、自律走行が上手くいく条件が違うことと、3 章で作成したノードのうち、地図編集ノードとウェイポイント指定ノード A の有効性を確かめるために実験を行った。実験は 2 つの環境でそれぞれ 2 回実施し、実験 1 で従来のノードのみの走行実験を行った。そして、実験 2 で 3 章で作成したノードを用いて走行を行い、2 回の実験の走行の違いを調査した。

4.1.1 屋内環境

- コース
  奈良女子大学 G 棟 4 階 G403 → C 棟 4 階 1 周→ G403(図 6)

図 6: 屋内環境のコース
(奈良女子大学キャンパスマップ http://www.nara-wu.ac.jp/nwu/intro/access/campusmap/ を加工)

- 実験 1
  ウェイポイントの指定がうまくできた場合は、G403 から出た後、G 棟と C 棟を結ぶスロープを登り、

C 棟に入る所まで進むことができた。しかし、安定した自律走行を得るためには、何度もウェイポイントの指定を繰り返し、ウェイポイントの間隔や数、位置を調整する必要があった。さらに、C 棟の地図を作成したところ、歪みが生じ、本来平行である道同士が交わってしてしまった (図 7)。このような地図では、ウェイポイントを指定しても自律走行時に本来の環境とのマッチングができない。これらのことから実験 1 では、ロボットが自己位置を見失い、自律走行不可という結果となった。実験 1 で見つかった課題を以下に列挙する。

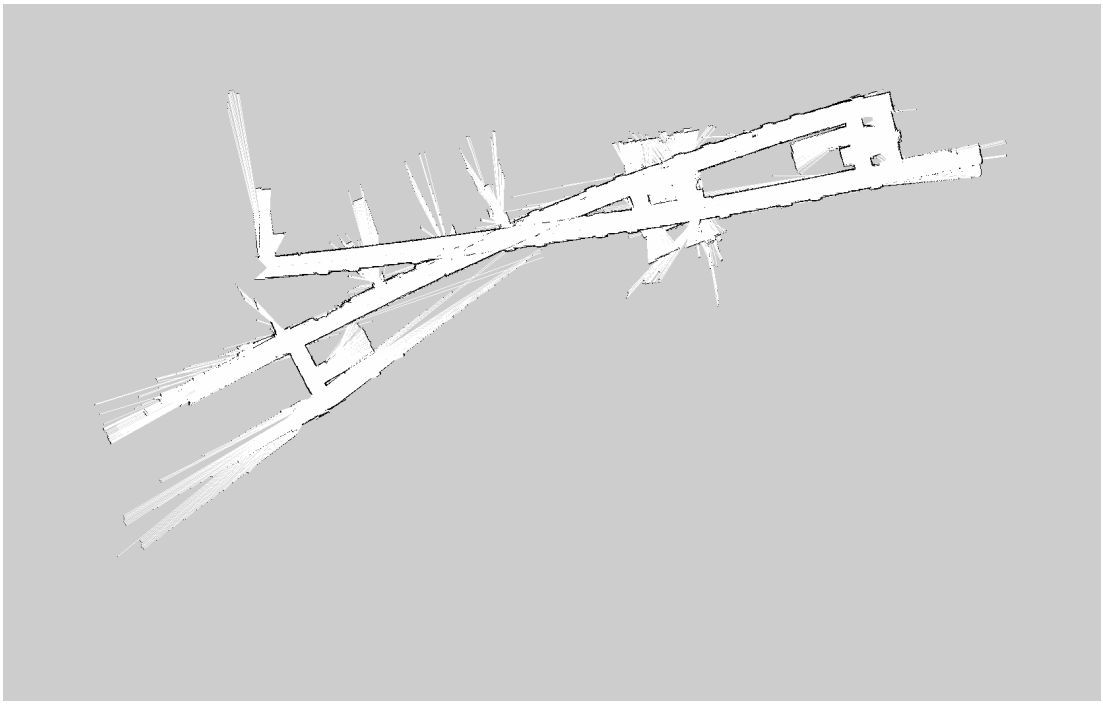


図 7: C 棟を 1 周した地図
途中で道が交わっている

1. G403 からの出入り
   →扉の幅が狭いため、ウェイポイントの指定が少しでもずれると通り抜けることができない

2. G 棟から C 棟へのスロープ
   →スロープ横の廊下と形状が似ており、誤って廊下にウェイポイントを指定してしまう

3. C 棟の地図作成
   →地図にゆがみが生じることで道が本来クロスしない場所でクロスし、地図と環境とのマッチングができない

- 実験 2
  実験 1 で見つかった前述の課題のうち、1 つ目と 2 つ目はウェイポイント指定ノード A を使用し、自動でウェイポイントを指定したところ、解決することができた。ロボットが通った地点にウェイポイン

トを指定することで、扉にぶつかったり、誤ってスロープでなく廊下にウェイポイントを指定してしまうことがなくなったからである。また、その結果、何度もウェイポイント指定を繰り返し、様々な調整を行う必要もなくなった。その際の地図を図 8 に示す。
3 つ目の課題を解決するには、2 つの手法を試した。まず、1 つ目は地図作成方法に人間が注意する手法である。歪んだ地図を分析すると、長距離でかつ、直線の道の地図を作成するときに、地図が少しずつ左に曲がっていっていることがわかる。そこで、C 棟を反時計回りに作成していた地図を時計回りに替えて作成すると、道が交わることなく作成することができた。2 つ目は地図を分割する手法である。地図の分割は、地図作成時に任意の地点で行うことができるので、道がクロスする前に分割を行うことが可能である。以上 2 つの手法、それぞれで実験 2 を行った結果、いずれもゴールまで到達することができた。

- まとめ
  屋内での走行時は扉や棚などの障害物が多く、狭い廊下を走行しながら、それらの障害物にぶつからないように走行するためには、正確なウェイポイント指定が必要であった。その点で、ロボットが通った場所にウェイポイントを指定する教示再生走行の手法を用いた、ウェイポイント指定ノード A は有効であるといえる。また、屋内環境では、壁が常にあり、自己位置推定の目印となるので、自己位置を見失うことがほとんどない。従って、地図の分割を行うよりも地図作成方法に注意する手法のほうが、自律走行までの手間が少なく、効率が良いと考えられる。
  なお、その後の実験で、作成される地図の歪み (真っ直ぐな道が地図上ではカーブになってしまう現象) は、センサの取り付け位置・角度及び相互のセンサの位置関係に関する情報を正確に設定することで解消することが判明した。

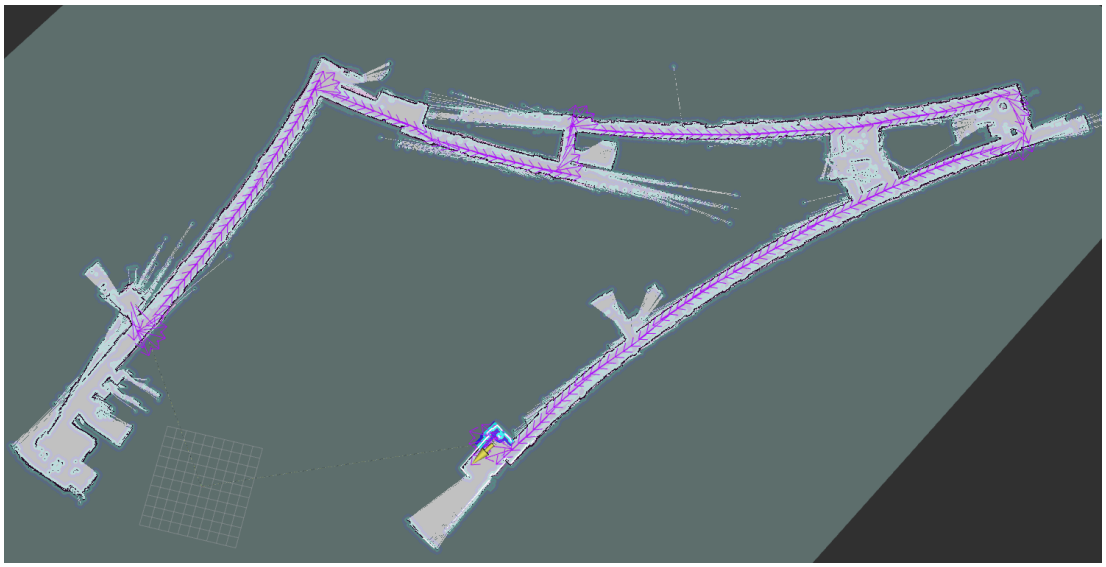


図 8: 屋内環境にウェイポイントを指定したときの rviz の画面
（ウェイポイント指定ノード A を使用）

4.1.2 屋外環境

- コース
  G 棟エントランス→西門前→ F 棟前→南門前→ G 棟エントランス (図 9)

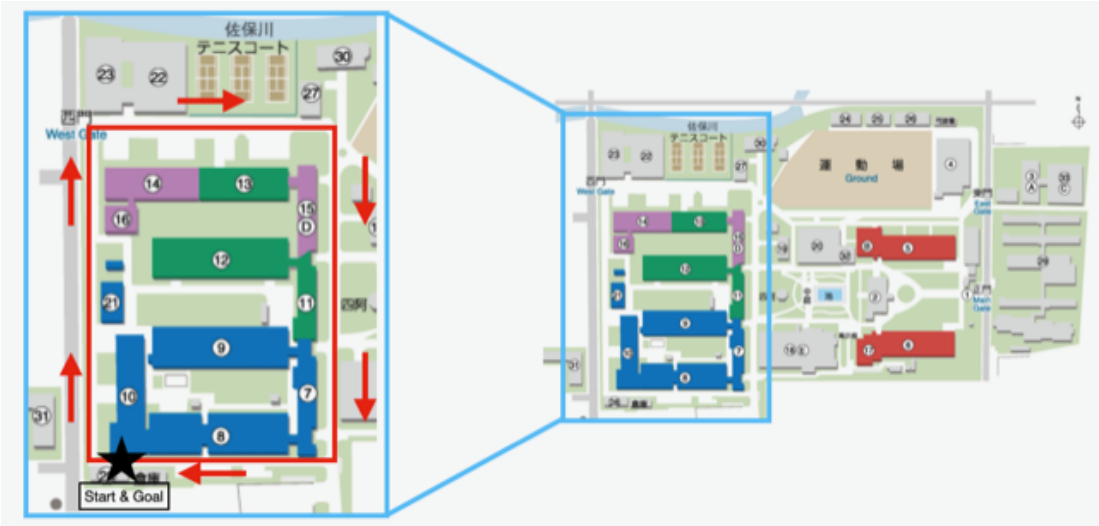


図 9: 屋外環境のコース
(奈良女子大学キャンパスマップ http://www.nara-wu.ac.jp/nwu/intro/access/campusmap/ を加工)

- 実験 1
  地図作成を時計回りに行ったため、地図の歪みによる自律走行の失敗は起こらなかった。しかし、屋内環境よりも地図が大きいので、操縦者の軌跡の削除を行う際に、消し残しがあり、それらを障害物と認識した結果、自己位置を見失ったり、急に全く違う場所に飛んでしまった。また、花壇などのセンサよりも低い位置に存在する障害物が地図には反映されないので、ウェイポイントの指定が困難であった。さらに、屋内環境と比べてコースが長距離であったのでウェイポイントをすべて手動で指定するには時間がかかるうえに、地図が 1 枚なので、自律走行でウェイポイント指定が上手くいかなかった箇所があると、すべてのウェイポイントを指定し直す必要があった。実験 1 では、人間が地図を見ただけでは、地図に反映されていない障害物を避けて、ウェイポイントを指定することができず、スタートから 1 つ目の角を曲がるまでしか自律走行はできなかった。

- 実験 1 で見つかった課題
  1. 地図に反映されない障害物
     →センサよりも低い位置にあるため、地図には反映されておらず、人間がそれらの場所を推定してウェイポイント指定を行うのはかなり難しい

  2. 膨大な数のウェイポイント指定
     →コースが長距離でかつ、屋内よりも目印となるものが少ない屋外環境のため、大量のウェイポイント指定が必要である。また、自律走行が上手くいかなかった場合は、最初からすべて指定し直す

必要があるため効率も悪い

3. 地図編集における見落とし
   →屋内環境よりも環境の変化が激しいので、地図作成時の地図をそのまま自律走行に使用することができず、また地図のサイズが大きいため地図編集において見落としが多く発生し、自律走行が上手くいかない原因となる

- 実験 2
  実験 1 で見つかった課題のうち 1 つ目と 2 つ目は、ウェイポイント指定ノード A を使用し、ロボットが通過した地点にウェイポイントを指定することで解決できた (図 10)。地図に反映されてない障害物の上に、ウェイポイントが指定されることがなく、また、自動で指定が行われるため、膨大な数のウェイポイントを手動で指定する必要がなくなった。さらに、繰り返し指定し直す必要もないため、自律走行までにかかる手間も大幅に削減することができた。
  3 つ目の課題は、地図編集ノードによって解決した。地図のサイズが大きく、細かな障害物を目視では見落としてしまうことが原因だったので、ノードで地図を編集することで障害物を表す黒色の点を強調し、見落としにくいようにした。これによって、すべての障害物の削除を完璧に行えるわけではないが、自律走行時に走行の妨げとなるような大きさの障害物は削除することができた。実験 2 では、1 回のウェイポイント指定でコースを完走することができた。

- まとめ
  屋内環境と同様に、正確なウェイポイント指定が必要なため、ウェイポイント指定ノード A が有効であることが確かめられた。さらに、屋外環境における実験では、屋内環境よりも環境の変化が激しく、作成した地図をそのまま自律走行には使用できないことがわかった。また、地図編集を行う際に、屋外環境では地図のサイズが大きく、障害物を強調する処理を加えたほうが、見落としてしまった障害物によって、自律走行が失敗する可能性も抑えられた。以上より、地図編集ノードの有効性を確認できた。

4.1.3 奈良女子大学での実験のまとめ

奈良女子大学での実験では、屋内環境と屋外環境という環境が異なる場所において、走行実験を行った。いずれの環境においてもウェイポイント指定ノード A によるウェイポイント指定で自律走行の安定性が格段に上がったと言える。さらに、屋外環境においては、地図編集ノードの有効性も確認することができた。

また、環境の違いによって、ウェイポイント指定の間隔や数、自律走行時の速度など、様々な条件を変更したほうが安定した自律走行が得られることが確認できた。例えば、屋内では幅の狭い扉を通り抜けたり、少ないスペースで角を曲がるなど、細かな動作が求められる。また、スピードを出しすぎてしまうと、障害物に勢い良くぶつかってしまうこともあるので、ウェイポイントを細かく指定する必要がある。反対に、屋外ではウェイポイントを少なくすることでスピードを出すことが可能であり、安定した走行が得られる。

さらなる課題として、地図編集の自動化が挙げられる。これについては、地図上で黒で表されている障害物を自動で削除する機能を地図編集ノードに追加する必要がある。これにより、コースが長距離になっても、編集が負担にならないと考える。

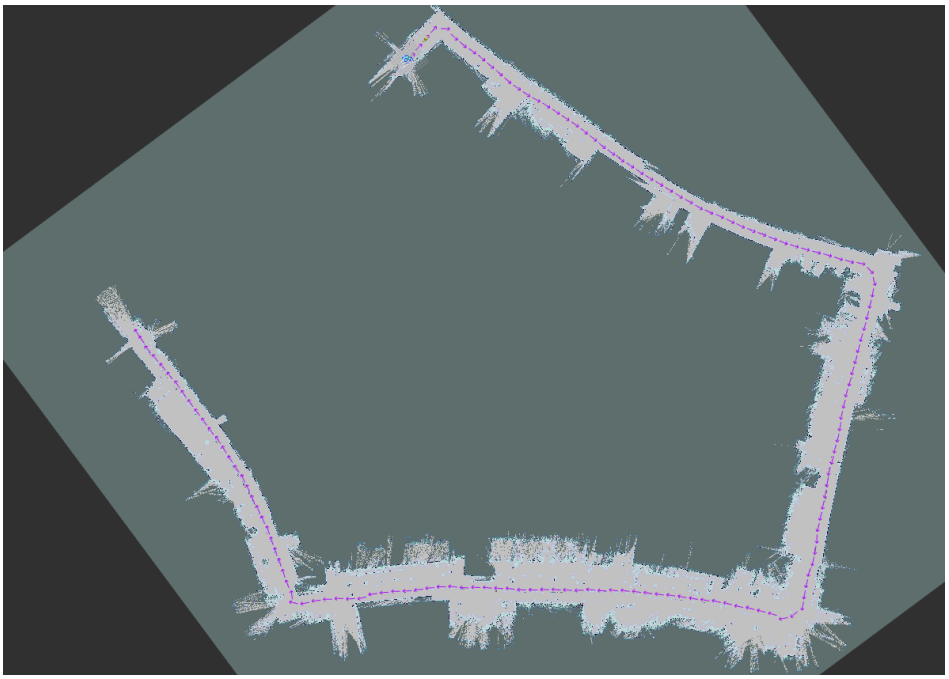


図 10: 屋外環境にウェイポイント作成した時の rviz の画面
（ウェイポイント指定ノード A で作成）

また、奈良女子大学内では、人通りがほとんどなく、歩行者が自律走行の障害となることがほとんどなかった。そこで、多くの歩行者が通行している環境でのロボットの振る舞いを調査するため、学外の人通りが多い環境で実験を行う必要があると考えられる。

4.2 中之島チャレンジ

奈良女子大学内での走行実験は、歩行者の移動のような環境の変化が少なく、道の形状も限られているなど、比較的簡単な環境であった。そこで、より複雑な環境での走行実験を試みるために中之島チャレンジ [3] に参加した。

中之島チャレンジは、2018 年より自律走行ロボットの実環境実験として行われている。中之島公園内の定められたコースを自律走行で 1 周することが基本課題であり、他にオプション課題も定められているが、我々は基本課題に参加した。2021 年度の中之島チャレンジは 7/17, 7/18, 10/23, 10/24 の 4 日間に渡って開催され、実験走行 3 日間、記録として残る本走行 1 日の構成であった。

4.2.1 コース

図 11 のとおりに中之島公園内の中央公会堂、中之島図書館周辺の道路を含む約 477 mを 1 周する。

4.2.2 実験走行

3 日間の実験走行では、本走行で自律走行を行うための地図作成、ウェイポイント指定、地図編集を行った。

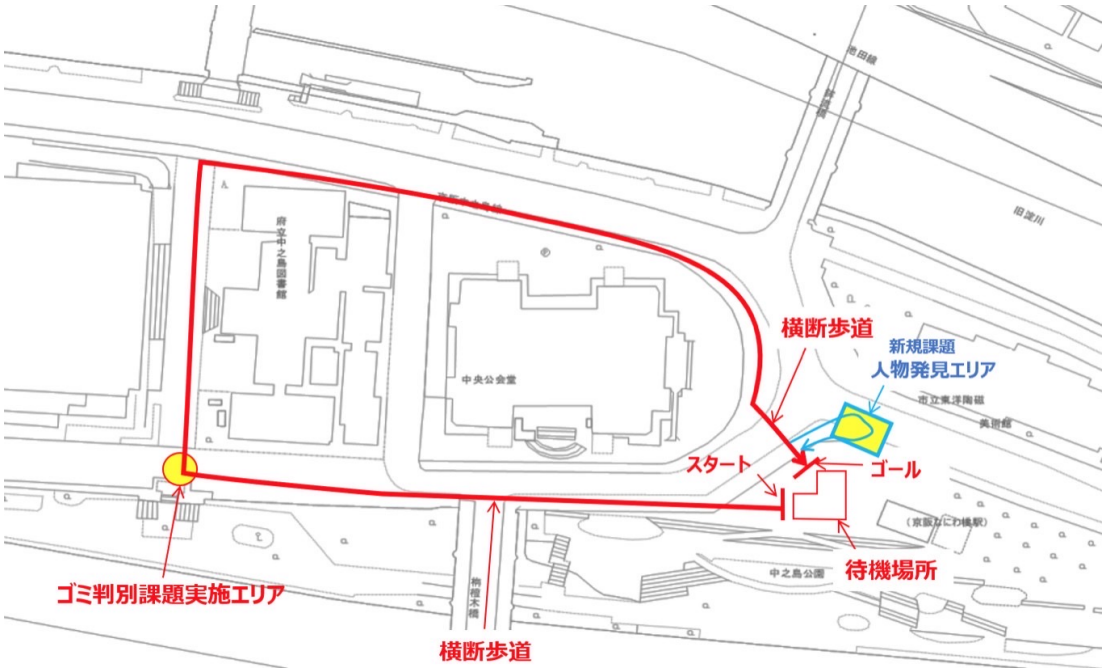


図 11: 中之島チャレンジのコース [3]

- 地図作成
  初めにコースを 1 周する地図を作成した。しかし、コースが長距離で、地図編集が困難であったので、2 分割して作成した。

- ウェイポイント指定
  奈良女子大学での実験同様に、手動での指定が難しかったので、ウェイポイント指定ノード A を用いた。また、並行して、ウェイポイント指定ノード B を用いた指定も行い、2 つのノードのウェイポイント指定の違いも調査した。

- 地図編集
  自動で障害物を削除するノードの作成が間に合わなかったため、手動で行った。

4.2.3 本走行

本走行は各チーム 2 回行われた [4]。スタート時間が決められており、時間内にスタートできなかった場合、そこで終了となる。本走行では、ウェイポイント指定ノード A で作成した、ウェイポイントファイルを使用する予定であったが、ウェイポイントの指定に失敗してしまい、修正が間に合わなかったので、手動でウェイポイントの指定を行った。1 回目は 38m 地点でコース左側の植木に突っ込んでしまったため、非常停止スイッチを押し、終了した。2 回目は 1 回目に植木に突っ込んでしまったことを考慮し、ウェイポイントの指定を再度、手動で行った。そのため 38m 地点を通過することはできたが、70m 地点で 1 つ前にスタートしたロボットに追いついてしまったことに加え、20 人ほどの団体がロボットの前を通過したことで、壁と誤認識し、本来の環境とマッチングができず、自己位置を見失った結果、コースから外れてしまい、終了した。

4.2.4 中之島チャレンジのまとめ

- ウェイポイント指定
  ウェイポイント指定ノード B を用いた結果、ウェイポイントの指定と自律走行が可能であった。しかし、ロボットが通ることができない場所を指定してしまうことがあるので、いくつかのウェイポイントを削除する必要があることがわかった。また、パスポイントの指定場所によってもある程度コントロールが可能であることがわかった。結果として、ウェイポイント指定ノード B はウェイポイント指定ノード A に比べると、自律走行の安定性がそれほど高いとは言えない。今後、さらなる安定性を求めて改良を行う必要がある。

- 地図作成・編集
  今回の実験走行では、地図の分割を行ったが、1 枚目の地図でウェイポイント指定ノード A を使用した際に、初期位置とスタート地点がずれていることに気づかず、指定を行ったため、すべてのウェイポイントの座標がずれてしまった。本走行直前にウェイポイントがずれていることに気づき、指定し直す時間がなかったので、本走行では分割なしの地図を使用した。また、地図編集ノードの作成も間に合わなかったため、手動で地図修正を行った。分割した地図での自律走行とノードによる自動の地図編集の実現は、4.3「中之島エクストラチャレンジ」での目標とした。

- 自律走行
  中之島チャレンジでは、人通りの多い環境で自律走行を試すことができた。ロボットは自律走行時にセンサを用いて環境の認識を行い、障害物を避けるが、その障害物が人間であるか、そうでないかによって行動を変化させることはできない。例えば、多くの人がロボットの進路を横切っている場合、人間が通り過ぎるまでその場で一時停止することはできず、人間を壁と認識してしまい、地図とマッチングを行った結果、推定自己位置が現在地から飛んでしまった。今後の課題として、地図作成時にはなかった障害物があった場合にロボットをどのように行動させるかが挙げられる。

4.3 中之島エクストラチャレンジ

中之島エクストラチャレンジ [5] は、中之島チャレンジの追加実験と位置づけられ、中之島チャレンジとは異なった環境において、実験を行う。2021 年度の中之島エクストラチャレンジは、初の開催地である八幡屋公園で行われた。八幡屋公園内は中之島公園とは異なり、周りに建物がほとんどなく、ロボットが目印とできるものが少ない環境である。2021 年度の中之島エクストラチャレンジは 11/27, 11/28, 12/1 8 ,12/19 の 4 日間に渡って開催され、実験走行 3 日間、本走行 1 日の構成となっていた。

4.3.1 コース

図 12 のとおりに八幡屋公園内の芝生広場の周囲、約 535 mを 1 周する。中之島エクストラチャレンジでもコースの自律走行の実験に加えて、人物発見やゴミ認識などの課題や基本コースの他により長く高低差のある丘コースも用意されていたが、今回は基本コースの完走を目指した。

図 12: 中之島エクストラチャレンジのコース [5]

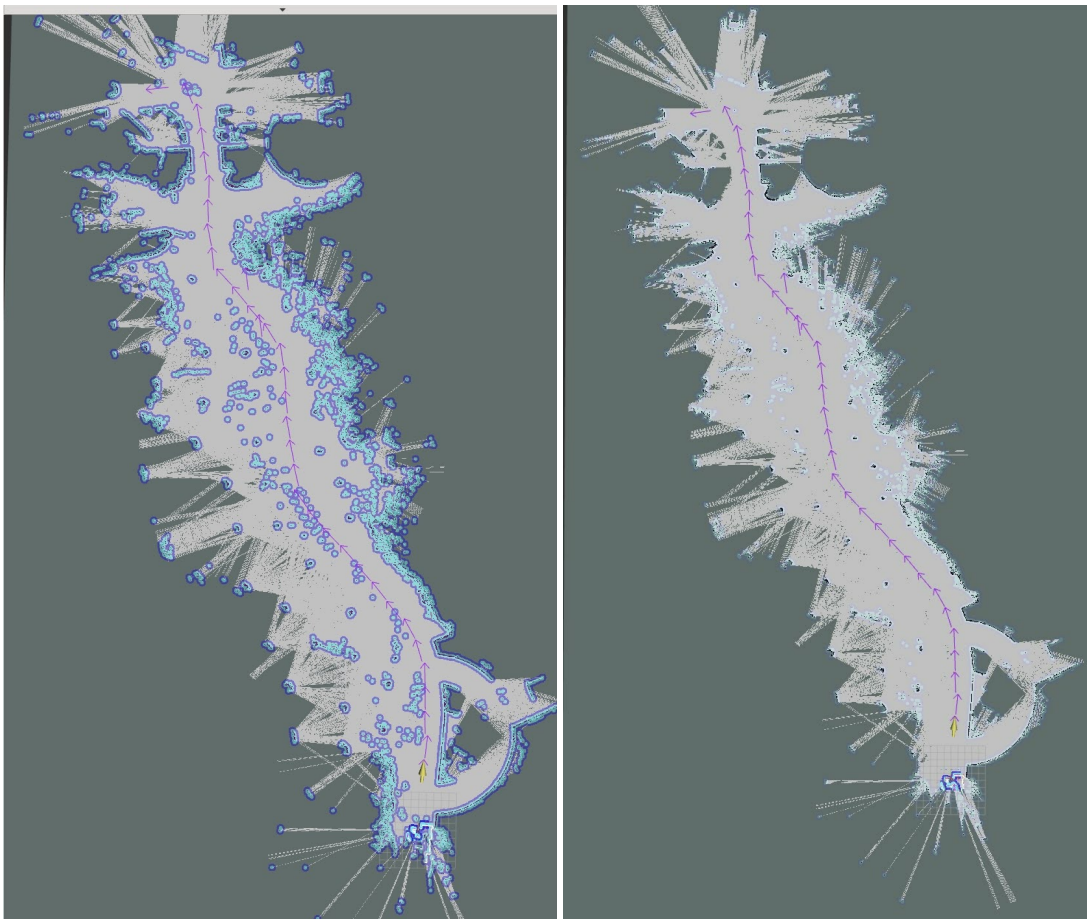
4.3.2 実験走行

3 日間の実験走行では、中之島チャレンジと同様に、本走行で自律走行を行うための地図作成、ウェイポイント指定、地図編集を行った。

- 地図作成
  地図を自動で切り替えることによる自律走行を試すため、地図を分割して作成した。地図を分割する地点は、ウェイポイント指定も同じ地点から始めなければならないことから、手動での地図操作作業を困難にしないためには、分割地点は人間がわかりやすい地点でなければならない。さらに、地図を切り替えたときに、ロボットが地図とのマッチングを行いやすいように、目印となるものがあることが望ましい。しかし、公園内には建物が少なく、目印となるものがほとんどなかったため、分割に適した場所は多くなかった。さらに、分割数を増やすほど、ウェイポイントファイルの作成の手間も増えるので、分割数を増やしすぎない方が良いと考えた。以上を考慮し、最終的に地図を 4 分割した。

- ウェイポイント指定
  ウェイポイントの指定はウェイポイント指定ノード A を用いて行った。分割した地図を開き、ウェイポイントを指定する。そして、次の地図への切り替え地点に到達したら、ノードを終了し、地図を開き直して、ウェイポイント指定を行う。以上の作業をゴール地点まで繰り返した。

- 地図編集
  地図編集ノードにロボットの操縦者の軌跡を削除する機能を追加し、その機能によって図 13 に示すように、地図の編集を行った。



(a) 削除前　　**(b)** 削除後

図 13: 地図編集ノードによる操縦者の軌跡削除

4.3.3 本走行

本走行は中之島チャレンジと同様に 2 回行った。結果は 1 回目 250m、2 回目 230m [6] であった。両走行ともに、同じ地図とウェイポイントファイルを使用し、自律走行を行ったが、1 枚目から 2 枚目の地図への切り替え後、少し進んだところで、自己位置を見失って終了した。自己位置を見失った原因は、切り替え直後の環境が開けた場所であり、地図とのマッチングに失敗したからだと考えられる。

本走行では、改良を加えたナビゲーションシステムを使用し、地図とウェイポイントファイルの切り替えを自動で行った。1 枚目から 2 枚目の地図の切り替えとウェイポイントファイルの読み込みは問題なく行うことができた。また、分割による地図の範囲の縮小と地図 1 枚あたりのウェイポイント数の減少により、自律走行が続行できないほど、地図上の自己位置が飛んでしまうこともなくなった。

本研究で作成したノードによって、中之島チャレンジの時点より自律走行の安定性は飛躍的に向上した。また、参加したチームの中でも長い距離を走るという好成績を残すことができた。しかし、今回の本走行では完走をすることができなかったので、目印が少なく開けた場所における自己位置推定などを見直す必要があると考える。

4.3.4 中之島エクストラチャレンジのまとめ

- ウェイポイント指定

  今回の実験では、ウェイポイント指定ノード A によるウェイポイント指定を行った。奈良女子大学で実験を行ったときと同様に、ロボットが通過した地点をウェイポイントとすることでコースから外れてしまうなどの問題は起こらなかった。また、自動で指定を行うために何度も指定し直す必要はなくなり、自律走行までの事前のトレーニングにかかる手間を軽減することができた。

  今後の課題として、もう一度コースをたどる必要がなく、ロボットがなくてもウェイポイントの指定を行える、ウェイポイント指定ノード B の精度向上が挙げられる。

- 地図作成・編集

  今回の実験では、分割した地図を使用したが、地図を分割する際に目印となるものが必要など注意する点がいくつかあり、何度も地図作成を行い、自律走行が可能か試す必要があった。そこで、今後の課題として地図の分割の自動化が挙げられる。地図の分割地点をロボット自身が決定し、自動で行うことができれば、分割作業を人間が行う手間を省くことができ、さらにロボット自身が地点を決定しているので、自己位置推定に失敗するという問題も解決できると考える。

- 自律走行

  中之島エクストラチャレンジでは、目印となる建物が少ない環境で自律走行を試すことができた。目印が少ない環境では、自己位置推定が難しく、地図上の違う場所に現在地が飛んでしまうことがあるが、地図を分割したことで、現在地が飛んでしまう原因の 1 つである、似ている地形が 1 枚の地図に何箇所も存在する可能性を低くすることができた。しかし、今後の課題として、今回よりもさらに目印が少ない環境では、地図の切り替えが困難だと考えられる。そこで、そのような環境ではどのように自己位置推定を行うかが今後の課題である。

5 まとめ

本研究ではロボットの自律走行の安定を目指して、プログラムの作成と改良を行った。地図編集、ウェイポイント指定の自動化、ナビゲーションシステムの改善によって、Arno にすでに準備されていた従来のプログラムよりも安定した走行を得ることができた。さらに、中之島チャレンジでは初めて走行する場所でも、地図を分割したり、操縦者の軌跡を削除するなど、いくつかの工夫を行うことで、何度も事前のトレーニングを行うことなく自律走行が可能であることも実証できた。

現段階では、自律走行にはまだ事前に人間による操作を多く要しているが、今後地図の分割やウェイポイント指定の間隔の設定など、さらなる作業の自動化を行うことで、人間による操作ミスの削減、自律走行のさらなる安定性の向上、自律走行に至るまで時間の短縮を目指したいと考える。

6 謝辞

本研究を進めるにあたり、最後まで熱心にご指導いただいた、新出尚之准教授、並びに、ロボットの提供や中之島チャレンジで多大なるご協力をいただいた北陽電機の皆様に、深く感謝の意を表します。

参考文献

[1] Open Source Robotics Foundation. ROS. https://www.ros.org.

[2] Math Works. SLAM(自己位置推定と環境地図制作). https://jp.mathworks.com/discovery/slam.html.

[3] 中之島ロボットチャレンジ実行委員会. Nakanoshima Robot Challenge 2021. https://www.nakanoshima-rc.jp.

[4] 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 2021 10 月 24 日 本走行結果. https://www.nakanoshima-rc.jp/2021_nakanoshima-rc_result.pdf.

[5] 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 2021 エクストラチャレンジ. https://www.nakanoshima-rc.jp/extra2021.html.

[6] 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 2021 エクストラチャレンジ 12 月 9 日 本走行結果. https://www.nakanoshima-rc.jp/2021_nakanoshima-rc_extra_result.pdf.