ROS 対応 3D シミュレーターモデル
メガローバー版
取扱説明書

(2021.01.06)

Vstone®
ヴイストン株式会社

内容

1　はじめに / 注意事項

本書は、ROS 対応 3D シミュレーターモデル メガローバー版の取扱説明書兼チュートリアルです。シミュレーションに関する部分のみの説明になりますので、実機のメガローバーにおける ROS の利用方法及びメガローバー本体の取り扱い方法は、メガローバーをご購入の際に添付される各種資料をご参照ください。

本製品の使用にあたっては下記注意事項に従い、正しくご使用ください。

○ 本製品を無許可で複製、再配布、再販することはできません。ただし、著作権法で認められた範囲における複製については許可されます。
○ 本製品の対応環境は、ネイティブの Ubuntu 18. 04 と ROS Melodic です。それ以外の環境での使用についてはサポート対象外となる他、それによって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。
○ 本製品の使用にあたっては、 本製品に含まれない公開ライブラリおよびアプリケーションを多数使用する必要 がございます。本製品に含まれないライブラリについてはサポート対象外となる他、その使用によって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。
○ 本製品を使用する PC はお客様にてご準備ください。 Ubuntu のデバイスドライバの対応状況等により、一部の機能が正常に動作しない可能性がございますが、 デバイス固有の問題についてはサポート対象外となる他、それによって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。

2 ROS について

2.1 ROS の概要

ROS（Robot Operation System）は、OSRF（Open Source Robotics Foundation）によって開発・メンテナンスされているロボット用のミドルウェアです。分散処理が求められる複雑なロボットシステムを制御できる性能を備えており、世界中の研究者や開発者が作成した豊富なライブラリを使用することができます。ロボット制御システムの作成を効率化できることから、人型ロボットから車両型ロボット、水上・水中ロボットやドローンに至るまで、幅広い分野で活用されています。

ROS の特徴のひとつが、BSD ライセンスに基づくオープンソースとして公開されており、誰でも開発に参加し貢献できることです。ROS には強力な開発者コミュニティが存在し、誰でも使用可能な 5000 以上のライブラリのほとんどは、OSRF ではなくコミュニティによって開発・メンテナンスされています。

ROS 本体が BSD ライセンスによって提供されているため､ROS を用いて開発した成果物は、商用利用することが可能です。ROS を用いて動作する様々なロボットが発売されており、メガローバーもそのひとつです。ただし、ライブラリによっては BSD ではないライセンスによって提供されているものも存在するため、商用利用ではご注意ください。

2.2 ROS が提供する機能

ROS によって提供される主要な機能について、説明します。

- メッセージ通信

  ROS を用いて構成されるロボットシステムは分散処理が基本となっており、ユーザは「ノード」と呼ばれるプログラムを複数立ち上げることでシステムを作成します。例えば、ゲームパッドで操作できる台車ロボットであれば、「ゲームパッドの入力値を取得するノード」、「入力値に従って移動指令を出すノード」、「移動指令をもとにモータを回転させるノード」などが必要となります。当然、ノード間で情報を通信する仕組みが求められます。各ノード間で、センサ入力値の情報やカメラの映像、制御指令値といったデータをやり取りするために、ROS では Pub/Sub 方式によるメッセージ通信が提供されています。開発者はわずか数行のコードにより、任意の情報をパブリッシュ（配信）したり、サブスクライブ（購読）したりすることができます。メッセージには、構造体のような型が定められているため、ROS ノード同士であれば互換性を気にする必要はありません。また、型は自作することもできます。

- パッケージ管理

  ROS のライブラリやプログラムは、パッケージという単位で管理されています。パッケージの中にはノードやその設定ファイル、起動スクリプトなどが含まれており、ユーザは使用したい機能を持つパッケージをインターネット上からダウンロードし、ローカル環境に組み込むことができます。パッケージの追加や削除といった操作は非常に簡単に行う

事ができます。また、ユーザが独自の制御プログラムを開発する際には、まずパッケージを作成し、その中で開発を行う事になります。

- デバイスドライバ

  ROS では様々なデバイスのドライバがパッケージの形式で提供されています。対応しているデバイスであれば、パッケージを導入し、デバイスを接続するだけで使用することができます。

- ハードウェア抽象化

  ROS による制御システムは複数のノードによる分散処理によって動作します。これにより、ハードウェアが異なるロボットでも、上流の制御システムは同じものが使用できます。例えば未知環境の地図を作成する SLAM 機能を提供するパッケージ「gmapping」では、周囲の障害物の情報をレーザー光を用いた測距センサである LRF で取得することになっています。この LRF から情報を取得する部分は、LRF の種類に応じて異なるパッケージが提供されているため、ユーザは使用したい LRF を自由に選択することが可能です。そして開発者は、デバイス毎の違いを意識しなくても汎用的に使用可能な制御システムを作成することができます。

- ライブラリ

  ROS では、5000 を超える公開ライブラリを使用することができます。それらはデバイスドライバのようなものから、経路計画や動作生成といったものまで様々です。

- 視覚化ツール

  ROS には、いくつかの視覚化ツールが存在しており、ロボットの操縦 UI やデバッグ等のために活用されています。中でも「Rviz」は強力なツールです。3D 表示機能を持つこのツールは、実に多くのパッケージで情報を表示するために使われています。表示情報のカスタマイズが容易ですので、ユーザオリジナルの UI を作成することも容易です。

2.3 ROS に関する情報の集め方

ROS を用いた開発を行う際には、使用するパッケージの情報など、様々な情報が必要になります。ROS やその開発に関する情報は書籍から集めることもできますが、ここではインターネットから情報を集める際に参考になるサイトをいくつかご紹介します。

- ROS Wiki - http://wiki.ros.org/

  ROS の公式 Wiki です。ROS のインストール方法からチュートリアル、各公開パッケージの情報まで、様々な情報が公開されています。ただ、パッケージの情報などが更新されないまま古くなっていることもありますので、ご注意ください。

- ROS Wiki(ja) - http://wiki.ros.org/ja

  ROSWiki の日本語訳版です。現在も有志による精力的な翻訳作業が行われていますが、古い情報も多いので、英語版とあわせて確認した方がよいでしょう。

- ROS Answers - https://answers.ros.org/questions/

  ROS の Q&A フォーラムです。パッケージを使用した際のエラーの解消法など、様々な情報が蓄えられています。

3　ROS のインストール

ROS Melodic を Ubuntu にインストールする方法を説明します。ここで述べる手順は、ROS Wiki で紹介されているインストール方法から一部を抜粋したものです。より詳しい情報が欲しい方は、下記ページを確認してください。

ROS Melodic の Ubuntu へのインストール - http://wiki.ros.org/melodic/Installation/Ubuntu

ROS のファイルはインターネットから取得するため、インターネット接続が必要です。PC をインターネットに接続してください。ネットワークの接続制限があったり、プロキシが設定されている環境では、ファイルのダウンロードに失敗することがあります。

① 端末（シェル）を立ち上げる

ランチャー上部の「コンピュータの検索」から“端末”を検索するか、ショートカットキー“Ctrl+Alt+T”を使用して、新しい端末（シェル）を起動します。

図 3-1　コンピュータの検索

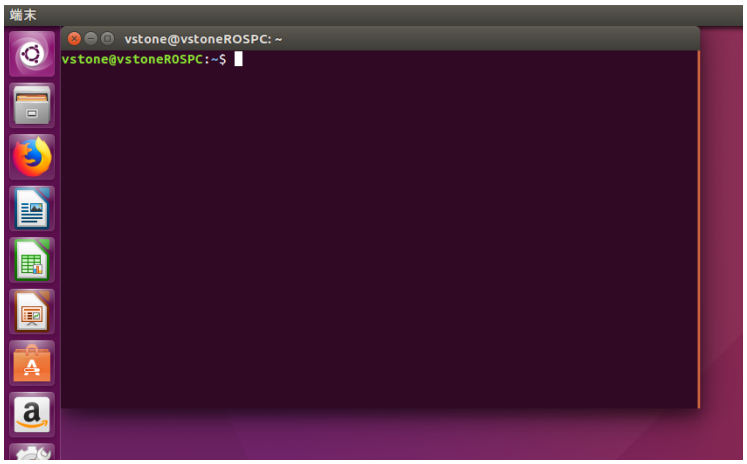

図 3-2　端末（シェル）

② sources.list を設定する

以下の青枠内のコマンドを端末にコピー&ペーストし、エンターキーを押して実行してください。コマンドラインへの貼り付けは右クリックまたは “Ctrl+Shift+V” で行えます。ひとつの青枠内にひとつのコマンドを書いていますので、複数行に分かれていてもまとめてコピーしてください。コピー&ペーストできない場合は手で入力してください。

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

③ 鍵を設定する

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

④ インストール

先にパッケージインデックスをアップデートしておきます。

```
sudo apt update
```

ROS Melodic をインストールします。ネットワーク環境によってはかなり時間がかかります。

```
sudo apt install ros-melodic-desktop-full
```

ROS のライブラリをバイナリでインストールする場合、上記のように「ros-バージョン-パッケージ名」で指定します。以下、バージョンに相当する箇所は、melodic をご利用の場合は「melodic」に置き換えて入力してください。

⑤ rosdep の初期化

以下の 2 つのコマンドを順に実行してください。

```
sudo rosdep init
```

```
rosdep update
```

⑥　環境設定

パスの設定を行います。Melodic の場合、以下の 2 つのコマンドを順に実行してください。

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

⑦　パッケージ構築のための依存ツールのセットアップ

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

以上で、ROS Melodic 本体のインストールは完了です。ROS の基幹プログラムを立ち上げるコマンド“roscore”を使って、起動することを確認しておきましょう。

```
roscore
```

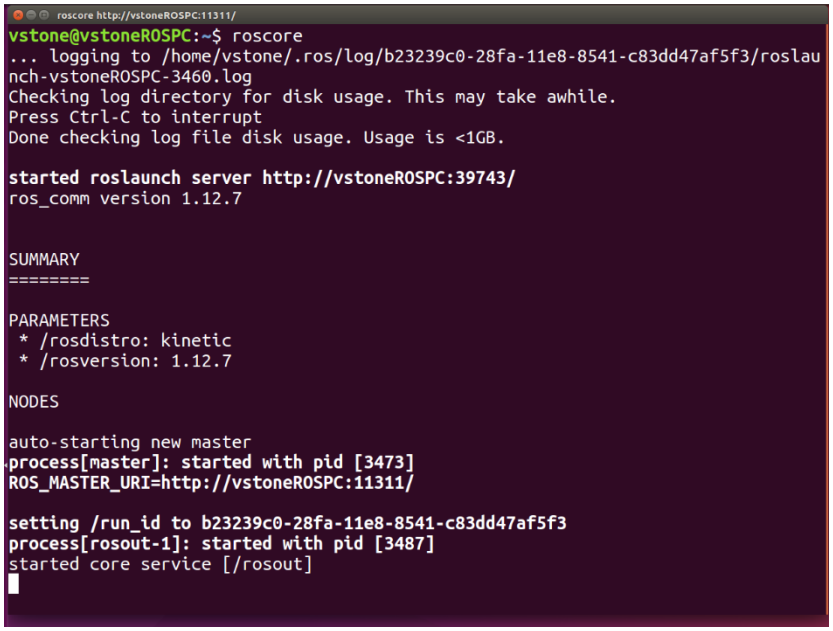

図 3-3 roscore 起動時のシェル

上のようなメッセージが表示されれば、ROS のインストールは正常に行われています。

roscore は、ROS の各サービスを提供する基幹プログラムです。ROS を使用する際には、必ずこの roscore を起動しておく必要があります。

4　メガローバー用サンプルパッケージのセットアップ

メガローバーを ROS で制御するサンプルパッケージのセットアップを行います。下記の手順に従って作業を行ってください。

4.1 catkin ワークスペースの作成

ROS のビルドシステムである catkin のためのワークスペースを作成します。この作業は、ROS Wiki に掲載されているチュートリアル「ROS 環境のインストールとセットアップ」（http://wiki.ros.org/ja/ROS/Tutorials/InstallingandConfiguringROSEnvironment）にも記載されています。

端末を起動して、以下のコマンドを順番に実行してください。

・フォルダの作成

```
mkdir -p ~/catkin_ws/src
```

・作成した src フォルダに移動

```
cd ~/catkin_ws/src
```

・ワークスペース作成

```
catkin_init_workspace
```

これでワークスペース「catkin_ws」ができあがりました。ワークスペースの src フォルダ内にパッケージフォルダを配置していくことができます。作成したばかりの src フォルダは空ですが、~/catkin_ws で以下のコマンドを実行することで、ワークスペースをビルドすることができます。

・(src フォルダに居る場合) ~/catkin_ws に移動

```
cd ~/catkin_ws
```

・ワークスペースをビルド

```
catkin_make
```

このワークスペース内で作成したパッケージを動作させるためには、ワークスペースをオーバーレイする必要があります。オーバーレイについては ROS Wiki にある catkin のチュートリアル「workspace_overlaying」(http://wiki.ros.org/catkin/Tutorials/workspace_overlaying)を確認してください。オーバーレイを行うためには、~/catkin_ws で次のコマンドを実行します。

```
source devel/setup.bash
```

ただしこの状態では、端末を新しく起動するたびにオーバーレイの作業を行う必要があります。端末起動時に自動的にオーバーレイが実行されるようにするためには、以下のコマンドを用いて設定を.bashrc に書き込みます。

```
echo "source /home/ユーザ名/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

以上で、ワークスペースの作成は完了です。

4.2 サンプルパッケージのダウンロードとビルド

GitHub で公開しているサンプルパッケージをダウンロードしビルドします。以下のコマンドを順番に実行してください。

・~/catkin_ws/src フォルダに移動します

```
cd ~/catkin_ws/src
```

・GitHub よりサンプルパッケージのソースコードをクローンします

```
git clone https://github.com/vstoneofficial/megarover_samples.git
```

・ビルドします、表示が[100%]になればビルド完了です。

```
cd ~/catkin_ws
```

```
catkin_make
```

ROS の公開ライブラリの多くは GitHub でソースコードを公開しています。GitHub で公開されているものについては、同じ手順でワークスペースに追加、ビルドして使用することができます。

4.3 サンプルパッケージの内容について

メガローバーの ROS サンプルパッケージ「megarover_samples」の内容について簡単に説明します。なお、本パッケージは、メガローバー実機とシミュレータ両方のパッケージを統合しているため、シミュレータでのご利用には無関係なファイルも含まれています。

- ノード

  megarover_samples にはメガローバー実機用に準備されたノードが含まれますが、シミュレータには関係ありません。

- launch ファイル

  launch フォルダ内に存在する拡張子が「launch」のファイルは、ROS で使用できる非常に便利なファイルです。ROS で制御システムを構築するためには数多くのノードを起動する必要がありますが、それをひとつひとつ手動でやるのは大変です。launch ファイルを使えば複数のノードを同時に起動することができます。

  他にも、パラメータの設定や remap 機能など、便利な機能を使用することができますので、積極的に利用しましょう。

  megarover_samples の launch フォルダ内には、サンプルプログラムを実行するための複数の launch ファイルが格納されています。各 launch ファイルについては、5 章以降で説明します。

- configuration ファイル

  configuration_files フォルダ内に存在する、拡張子が「yaml」や「lua」のファイルです。サンプルプログラム実行時に使用する様々なパラメータ等について記載された設定ファイルです。

- map ファイル

  navigation サンプルで使用する map ファイルを格納するフォルダです。初期状態では参考用の map ファイルが入っています。

- CMakeLists.txt と package.xml

  CMakeLists.txt と package.xml は ROS パッケージに必須な、catkin の設定ファイルです。詳しくは下記のサイトを参照してください。

  CMakeLusts.txt: http://wiki.ros.org/catkin/CMakeLists.txt
  Package.xml: http://wiki.ros.org/catkin/package.xml

4.4 パラメータについて

ROS を用いて SLAM や自律移動を行う場合、車両の大きさや、旋回中心とセンサとの位置関係といったパラメータは大変重要です。これらのパラメータが誤って設定されていた場合は、良好な動作結果が得られない可能性があります。

配布しております mecanurover_samples のシミュレータでは、メガローバーVer2.0 と Ver2.1 を基準にパラメータが設定されています。よって、お客様によってセンサ配置や向きの変更といった情報を変更される際は、パラメータの調整を行っていただく必要があります。

パラメータは環境によって最適値が変化するものが多いため、お客様の環境に合わせて調整を行ってください。なおパラメータの調整についてはサポート対象外となります。

シミュレータにおける車両サイズおよび LRF と旋回中心の位置関係について記載があるのは以下のファイルです。また、SLAM・navigation に関連する設定を記載したファイルについても列挙します。

シミュレーション上の寸法・軸配置等に関する設定

- robots/ vmegarover.urdf.xacro
- urdf/body/body.gazebo.xacro
- urdf/body/vbody.urdf.xacro
- urdf/wheel/vwheel.urdf.xacro
- urdf/wheel/wheel.gazebo.xacro
- urdf/wheel/wheel.transmission.xacro
- urdf/lrf/lrf.gazebo.xacro
- urdf/lrf/lrf.urdf.xacro
- configuration_files/vmegarover_controller.yaml
- configuration_files/vmegarover_costmap_common_params.yaml

SLAM・Navigatioin などで用いられる設定

- configuration_files/vmegarover_costmap_common_params.yaml
- launch/vmegarover_gmapping.launch
- launch/megarover_move_base_dwa.launch

8　シミュレータの起動

シミュレータを利用するための起動方法及び概要について説明します。下記の手順に従って作業を行ってください。

5.1 起動用 launch ファイルの実行

megarover_samples には、シミュレータを起動するための launch ファイルがいくつか準備されています。これを用いて一度シミュレータを起動してみます。以下のコマンドを実行してください。

```
roslaunch megarover_samples vmegarover_with_empty_world.launch
```

実行すると、下図のような画面が開きます。図中の中央にあるロボットがシミュレータ上のメガローバーです。サンプルの設定では前方に LRF が取り付けられており、青色がセンサの取得情報を可視化したものです。

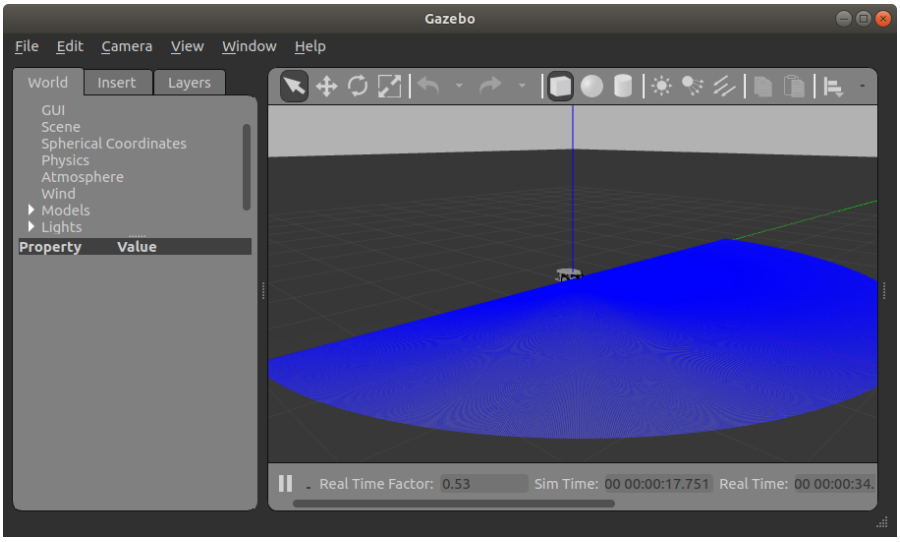

図 5-1 障害物がない空間でシミュレータを起動

シミュレーションの表示領域内をマウスでドラッグすると、視点の位置や角度を変更できます。ドラッグ時に押しているマウスのボタンに応じて変更するパラメータが変わります(左クリック:位置、右クリック:拡大/縮小、ホイールクリック:回転)。

シミュレータを終了させる場合は、ウィンドウ右上の×印をクリックしてウィンドウを閉じ、続いて launch ファイルを実行した端末上で Ctrl+C キーを押してください。しばらくするとシミュレータのプロセスを終了します。

同様に、下記のコマンドをそれぞれ実行すると、シミュレーション上に最初から障害物が存在する状態でシミュレータを起動します。

```
roslaunch megarover_samples vmegarover_with_sample_world.launch
```

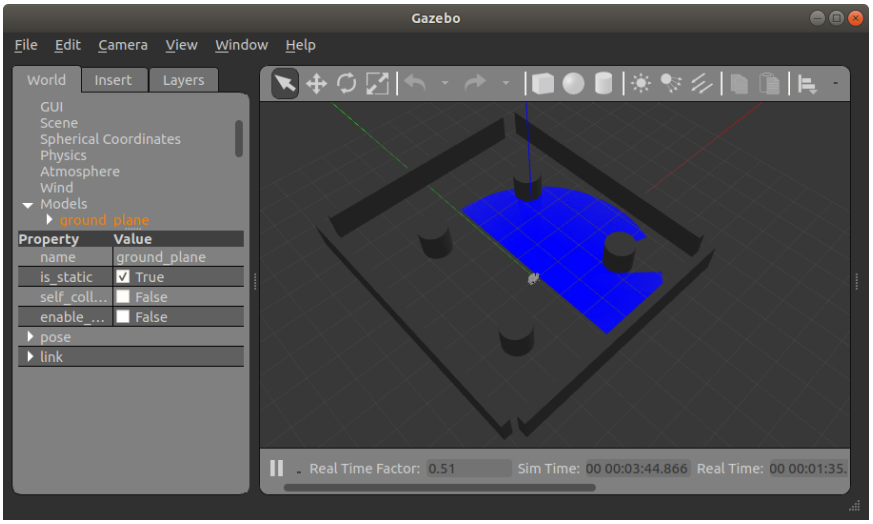

図 5-2 サンプルのワールドを読み込んでシミュレータを起動

```
roslaunch megarover_samples vmegarover_with_husky_playworld.launch
```

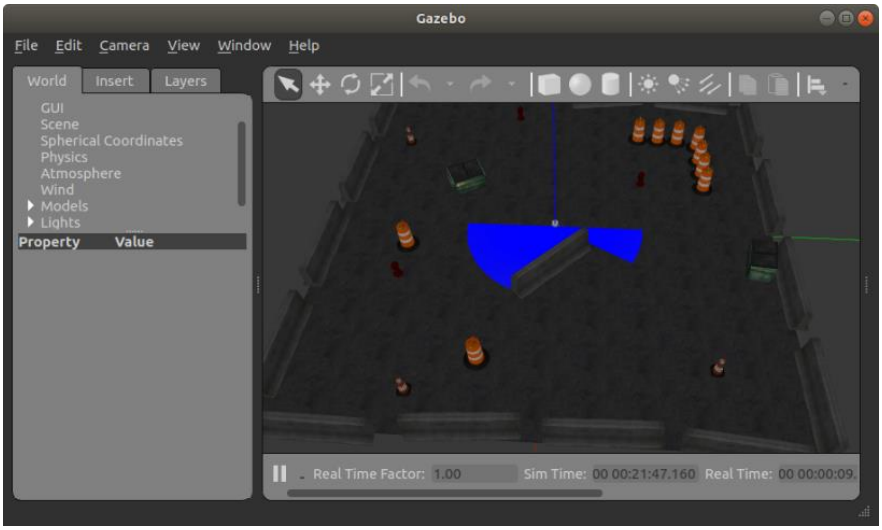

図 5-3 サンプルのワールドを読み込んでシミュレータを起動

次章以降の launch ファイルの説明については、必ずこれらのいずれかの launch でシミュレータを実行した状態で行ってください。

5.2 メガローバー用メッセージの仕様

シミュレータ上のメガローバーは、diff_drive_controller と言う独立二輪型のロボットをモデルとしたコントローラを利用しています。このコントローラは 1 つずつメッセージをサブスクライブ（購読）及びパブリッシュ（配信）しています。ここでは、各メッセージの仕様について解説します。

〔サブスクライブ〕

- /vmegarover/diff_drive_controller/cmd_vel

  シミュレータ上のメガローバーへの移動指令を記載するメッセージです。シミュレータ上のメガローバーはこのメッセージをサブスクライブし、記載されている並進移動量[m/s]および旋回量[rad/s]の指令値に従って動作します。

  表 6-1 に詳細を示します。メガローバーの座標系は、前方向を x 軸の正方向、左方向を y 軸の正方向にとる右手系で定義しています。メガローバーは並行二輪移動台車ですので、x 軸方向への移動と z 軸回りでの旋回が行えます。

  よって、“linear.x”に並進移動量[m/s]を入力し、“angular.z”に旋回量[rad/s]を入力してパブリッシュすることで、メガローバーをコントロールすることができます。

表 8-1　/rover_twist の詳細

メッセージ名	/vmegarover/diff_drive_controller/cmd_vel
型	geometry_msgs/Twist
内容	linear: 　x:　　//　x 軸方向の並進移動量指令値（float64） 　y:　　//　不使用 　z:　　//　不使用 angular: 　x:　　//　不使用 　y:　　//　不使用 　z:　　//　z 軸周りの旋回量指令値（float64）

〔パブリッシュ〕

- /vmegarover/diff_drive_controller/odom

  現在のホイールオドメトリを記録しているメッセージです。表 6-3 に詳細を示します。このメッセージは nav_msgs/Odometry 型のメッセージです。メッセージは、ロボットの現在の位置・方向を表す PoseWithCovariance 型と、現在の移動量を表す TwistWithCovariance 型の二つで構成されています。いずれも WithCovariance 型のため共分散が含まれています。

表 8-2　/vmegarover/diff_drive_controller/odom の詳細

メッセージ名	/vmegarover/diff_drive_controller/odom
型	nav_msgs/Odometry
内容	child_frame_id:　//子要素のフレーム名（string) pose:　//ロボットの現在の推定位置・方向 　pose: 　　position:　//　現在地の座標情報(point) 　　x:　　//　x 軸座標（float64） 　　y:　　//　y 軸座標（float64） 　　z:　　//　z 軸座標（float64） 　　orientation:　//　現在の方向情報(quaternion) 　　x:　　//　x 要素（float64） 　　y:　　//　y 要素（float64） 　　z:　　//　z 要素（float64） 　　w:　　//　w 要素（float64） 　covariance:　// 共分散（float64[36]） twist:　//ロボットの現在の移動量 　twist: 　　liner:　　//並進移動量（vector3） 　　x:　　//　x 軸方向の並進移動量（float64） 　　y:　　//　不使用 　　z:　　//　不使用 　　angular: //旋回量（vector3） 　　x:　　//　不使用 　　y:　　//　不使用 　　z:　　//　z 軸周りの旋回量（float64） 　covariance:　//　共分散(float64[36])

6　キーボード操作サンプル

キーボード操作サンプルは、PC に接続したキーボードを用いてメガローバーを走行させることができるサンプルプログラムです。本章では、その使用方法を説明します。

① キーボード入力パッケージのインストール

ROS でキーボードの入力を取得するためのパッケージをインストールします。端末を起動し、以下のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-teleop-twist-keyboard
```

② キーボード操作サンプルの実行

launch ファイルを用いてサンプルを起動します。あらかじめシミュレータを実行していることを確認してから、以下のコマンドを実行してください。

```
roslaunch megarover_samples vmegarover_keyboardctrl.launch
```

ロボットの操作には、キーボードの”u”,”i”,”o”,”j”,”k”,”l”,”m”,”,”,”.”の 9 個のキーを使います。k キーを中心として、各キーの配置がロボットに与える進行方向に相当します。前後方向はロボットの前進・後退、左右方向はロボットの左右旋回に変換されます。

7　マウス（タッチパッド）操作サンプル

マウス（タッチパッド）操作サンプルは、マウスまたはタッチパッドを使い、メガローバーを走行させることができるサンプルです。本章ではその使用方法を説明します。

① mouse_teleop パッケージのインストール

マウス（タッチパッド）操作サンプルの実行には mouse_teleop パッケージのインストールが必要です。端末を立ち上げ、以下のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-mouse-teleop
```

② マウス（タッチパッド）操作サンプルの実行

launch ファイルを用いてサンプルを起動します。あらかじめシミュレータを実行していることを確認してから、以下のコマンドを実行してください。

```
roslaunch megarover_samples vmegarover_mousectrl.launch
```

vmegarover_mousectrl.launch を呼び出すと、下図のようなウィンドウが画面に現れます。白いエリアをクリックし、上下方向にドラッグすると移動量 v が、左右方向にドラッグすると旋回量 w が出力され、シミュレータ上のロボットも走行します。

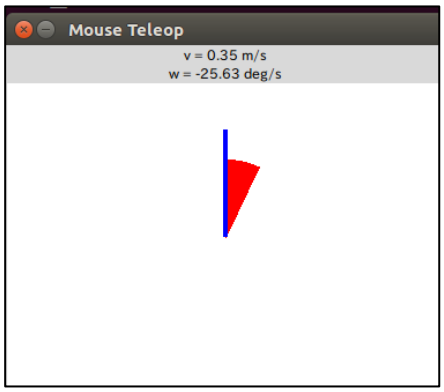

図 7-1　Mouse Tekeop の GUI

8　SLAM（gmapping）サンプル

SLAM(Simultaneous Localization and Mapping)は、ロボットの自己位置推定と環境地図作成を同時に行う手法です。ロボット掃除機などの自律移動ロボットでよく用いられており、現在も盛んに研究されている高度な技術です。

ROS には、この SLAM 手法を実装したパッケージが存在しており、誰でも簡単に SLAM を利用することが可能となっています。ROS で SLAM を実装したパッケージはいくつか存在しますが、ここでは「gmapping」を利用して、SLAM を実行してみましょう。

SLAM（gmapping）サンプルは、メガローバー前部に取り付けられた LRF で取得した情報をもとに、自己位置の推定と周囲の環境地図の作成を行います。メガローバーの移動は手動で行う必要がありますので、キーボード操作サンプルまたはマウス（タッチパッド）操作サンプルを一緒に使用します。

LRF(Laser Rangefinder）は、周囲に存在する物体の位置を検出することができるセンサです。レーザー光を発射し、その反射を見て物体の有無を確認しています。視野が広く、計測精度が高いことから、自律移動台車ロボットなどでよく使用されています。

「gmapping」は ROS でメジャーな SLAM パッケージのひとつです。LRF のデータと、車輪の回転数から移動量を計測するオドメトリの情報を用いて SLAM を行います。「gmapping」を用いて作成した地図の例を下図に示します。

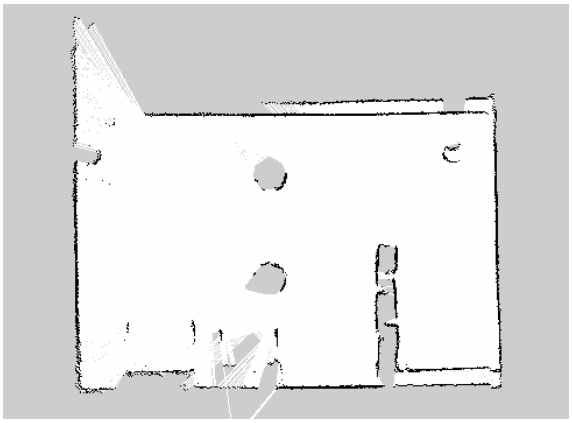

図 8-1　gmapping の地図作例

図中の黒い部分は障害物が存在する通行不可能な箇所、白い領域は通行可能な場所、灰色は未知領域です。地図はグリッド状になっており、各グリッドには障害物が存在する確率（占有確率）を示す 0～100 の値もしくは、未知領域を示す-1 の値が与えられます。この占有確率を用いた地図の表現方法は、ROS の 2D マップ表現方法として標準化されています。gmapping は絶対に障害物が存在しない 0 もしくは、絶対に障害物が存在する 100 の 2 種の値しか出力しません。

8.1　SLAM（gmapping）サンプルの実行

SLAM（gmapping）サンプルを実行する手順を説明します。以下の手順に従って作業を行ってください。

① gmapping のインストール

slam-gmapping パッケージをインストールします。次のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-slam-gmapping
```

② シミュレータのセットアップ

5.3 項に従いシミュレータを立ち上げます。なお、シミュレーション空間で SLAM を行うためには、障害物や壁を設置した world ファイルの準備が別途必要です。megarover_samples にはサンプルの world ファイルがあらかじめ収録されているため、ここでは最初からそれが組み込まれた launch を使います。

```
roslaunch megarover_samples vmegarover_with_sample_world.launch
```

続いて、メガローバーを手動で操作できるようにします。キーボードを使用する場合は 6 章の、マウスまたはタッチパッドを使用する場合は 7 章の手順に従い、各サンプルを使用して、メガローバーを操作できることを確認してください。

③ SLAM の実施

新しい端末で次のコマンドを実行し、SLAM（gmappig）サンプルを起動します。

```
roslaunch megarover_samples vmegarover_gmapping.launch
```

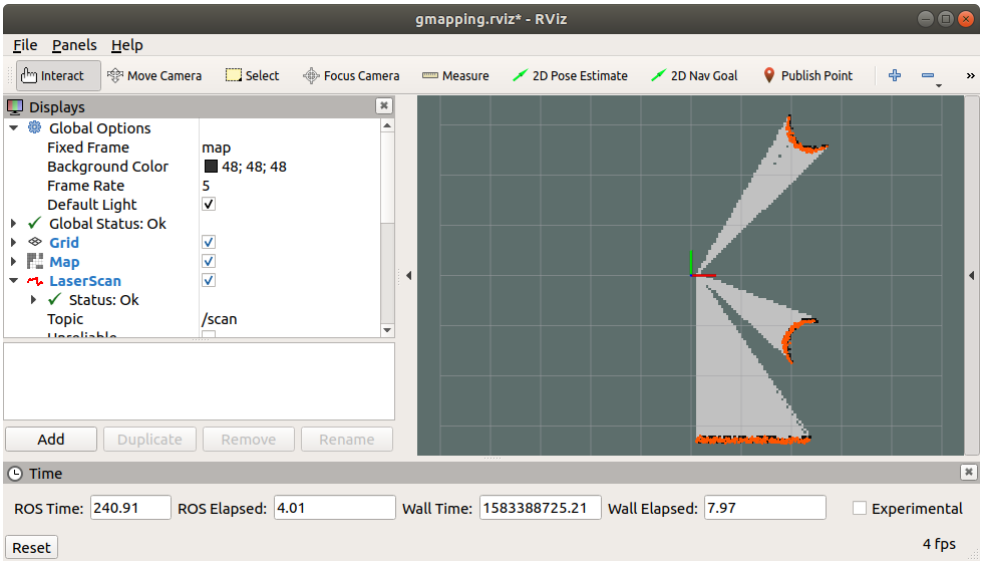

図 8-2　Rviz による gmapping の表示

サンプルが起動すると、上図に示すように Rviz が起動します。Rviz 上には LRF が捉えている障害物と、生成される地図がリアルタイムで表示されます。画面が表示されなかったり、LRF のデータが表示されなかったりした場合は、起動中に何らかのエラーが発生した可能性があります。一度プログラムを終了し、再度実行しなおしてください。

地図が生成され始めたら、メガローバーをゆっくりと移動させていきましょう。移動した範囲の地図が徐々に作成されていくはずです。急発進や急停止、急旋回などを行うと地図が乱れやすいので注意してください。

8.2 map の保存

作成した map を保存する方法を説明します。map の保存には map_server パッケージの map_saver ノードを使用します。map_server は navigation スタックに含まれるパッケージですので、まずは navigation スタックをインストールします。

```
sudo apt install ros-ROS バージョン-navigation
```

navigation スタックインストール後、次のコマンドを実行することで、ホームフォルダに地図の画像ファイルとデータファイルが保存されます。ファイル名は適宜設定してください。

```
rosrun map_server map_saver -f ファイル名
```

9　navigation サンプル

ロボットの自律移動を行うためには、自己位置推定や移動経路のプランニングといった技術が必要です。ROS では navigation スタックという形でこれらの機能が提供されています。スタックとは、複数のパッケージを束ねたパッケージ群のことです。ここでは、SLAM（gmapping）サンプルを用いて作成した地図を使って、メガローバーを目標地点まで自律的に移動させてみましょう。

9.1navigation スタックのインストール

まだインストールしていない場合は、navigation スタックをインストールします。navigation スタックに属する各パッケージは、以下のコマンドを実行することで一括でインストールすることができます。

```
sudo apt install ros-ROS バージョン-navigation
```

9.2 地図の作成と launch ファイルの編集

まず、ロボットが移動する環境の地図を作成します。自律移動を行うためには、できるだけ正確な地図が必要です。また、launch ファイルを編集し、作成した地図が読み込まれるようにします。

① 地図の作成

8.1 項の手順に従い gmapping を実行し、navigation で移動させたい領域の地図を作成してください。本来障害物が無いはずの場所に障害物があると判定された場合は、再び同じ個所を走行することで地図が正しく修正されることがあります。

② 地図の保存

8.2 項の手順に従い、map_saver を用いて地図を保存してください。地図が保存されていることを確認したら、SLAM（gmapping）サンプルは終了させてください。

③ 地図ファイルの移動

保存した pgm ファイルと yaml ファイルを以下のアドレスに移動させてください。

```
~/catkin_ws/src/megarover_samples/map
```

④ launch ファイルの編集

vmegarover_move_base_dwa.launch を編集し、作成した地図が読み込まれるようにします。以下のコマンドで launch ファイルを開いてください。

```
gedit
~/catkin_ws/src/megarover_samples/launch/vmegarover_move_base_dwa.launch
```

下記の行を探し、“ファイル名.yaml”を、先ほど保存，移動させた地図ファイルのファイル名に合わせて設定します。

```
<arg name="map_file" default="$(find megarover_samples)/map/ファイル
```

9.3 navigation サンプルの実行

それでは navigation サンプルを実行して、メガローバーに自律移動をさせてみましょう。なお、環境によっては上手く自律移動できないことがあります。その場合は、各機能のパラメータを調整いただくことで、改善する可能性があります。サンプルプログラムではオドメトリの影響が小さくなるようパラメータを設定しています。

平面で、オドメトリの誤差原因となるタイヤの滑りが少なく、分かれ道や柱の凹凸など、分かりやすい特徴が存在する通路といった環境でお試しください。LRF では捉えられない高さに障害物があったり、イスやテーブルの脚といった小さな障害物が多く存在する環境では正常に動作させることが難しくなります。

① launch ファイルの呼び出し

メガローバーと ROS が接続された状態で、以下のコマンドを実行して launch ファイルを呼び出します。

```
roslaunch megarover_samples vmegarover_move_base_dwa.launch
```

図 9-1 に示すように、Rviz 上に 9.1 項で作成した地図と現在の LRF が捉えた物体情報が表示されることを確認してください。

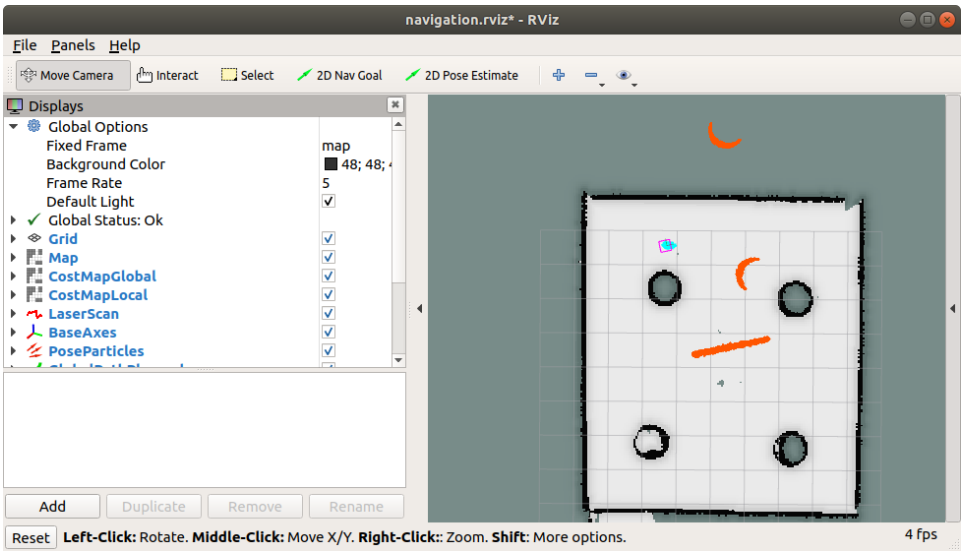

図 9-1　navigation サンプルの Rviz 画面

② 初期位置の設定

navigation サンプルが起動出来たら、メガローバーのおおよその初期位置を教えてやる必要があります。画面上部の「2D Pose Estimate」ボタンをクリックしてから、図 9-2 に示すように、シミュレータ上のメガローバーの実際の初期位置をクリックし、ドラッグして方向を決めます。

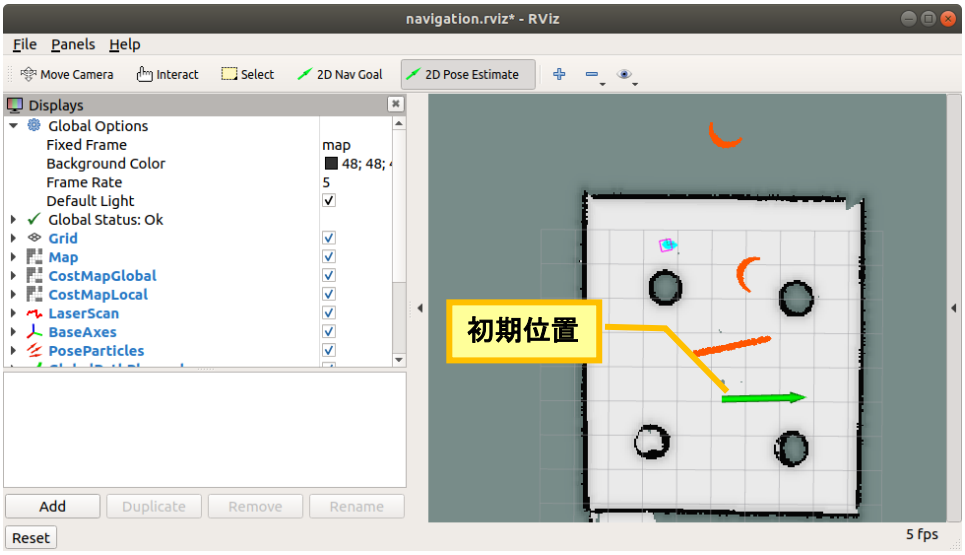

図 9-2　メガローバー初期位置の設定

図 9-3 に示すように、map の壁や障害物と、LRF の検出結果がおおよそ一致するようになるまで調整を行ってください。

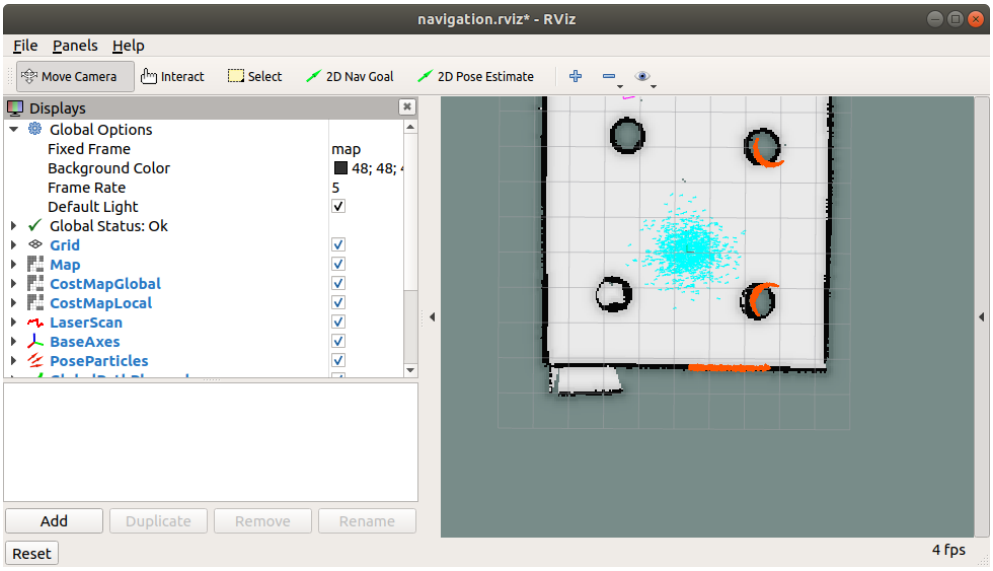

図 9-3　初期位置の設定が完了した状態

このとき、指定したメガローバーの位置周辺に青い小さな点がいくつも散らばっていることが確認できます。これらは、navigation スタック内で自己位置推定を担うノード「amcl」が計算する、メガローバーの推定自己位置です。動作開始前は自己位置が定まっていないため、青い点は大きく散らばっています。

③　目標位置の設定

初期位置の設定を完了したら、いよいよメガローバーの移動目標地点を設定します。

画面上部の「2D Nav Goal」ボタンをクリックしてから、図 9-4 に示すように、目標位置をクリックし、ドラッグして方向を決めてください。

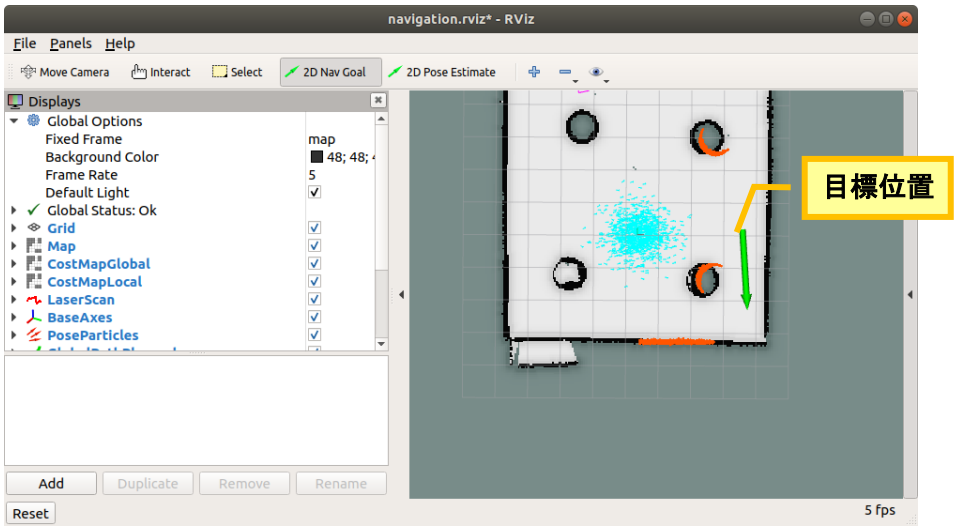

図 9-4　目標位置の設定

目標位置を設定すると、メガローバーは即座に走行を開始します。走行中の Rviz の様子を図 9-5 に示します。計画された移動経路が緑線で、目標位置姿勢が赤矢印で示されてい

ます。また、検出されている障害物の周りには色が付いた領域が存在します。これらは、観測結果をもとに数値化された衝突危険性を表すコストマップです。

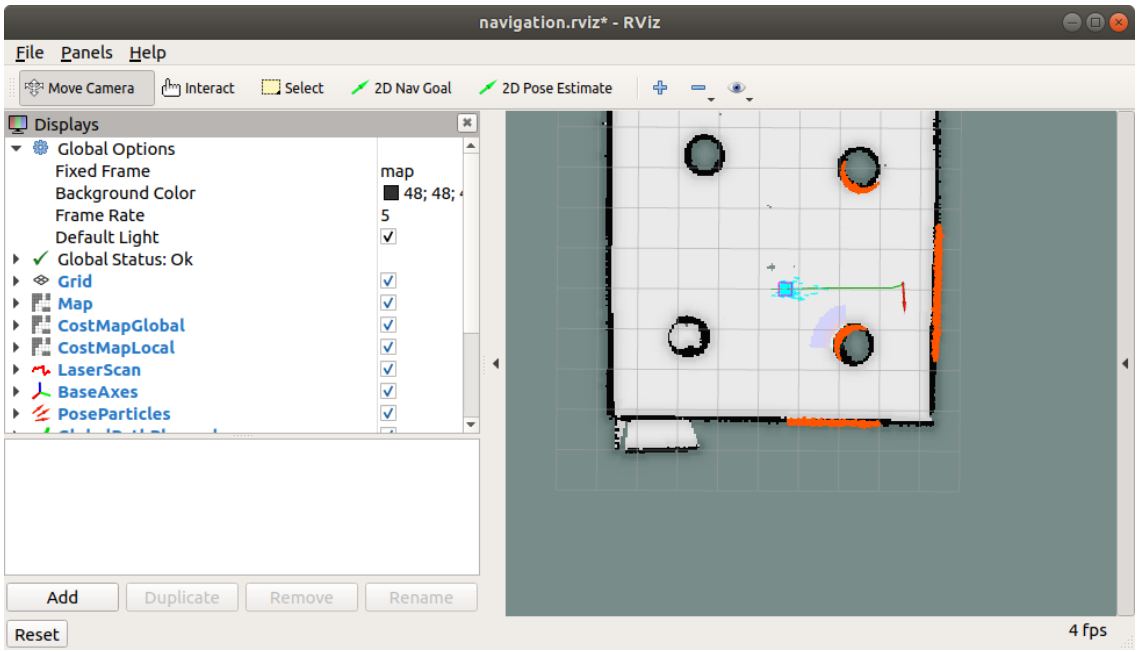

図 9-5　navigation による走行の様子

④　走行の終了

　メガローバーは目標位置に到達すると、自動的に停止します。そこから新たに目標位置を設定することも可能です。

10 動作確認バージョンについて

メガローバーの各サンプルが使用しているパッケージについては、以下のバージョンで動作確認を行っております。“apt”コマンドを用いてバージョン指定のオプションを付けずにバイナリを導入した場合、動作確認を行っていないバージョンが導入されシステムが正常に動作しない可能性がございます。また、Github からソースコードを取得してビルドする導入方法でも同様です。その際は、バージョン指定のオプションを用いて弊社にて動作確認を行っているバージョンのパッケージを導入してください。

表 10-1 動作確認パッケージのバージョン

パッケージ名	バージョン
rosserial	0.8.0
mouse-teleop	0.2.6
keyboard-teleop	0.6.2
gmapping	1.3.10
cartographer	1.0.0
navigation	1.14.4

商品に関するお問い合わせ

TEL: 06-4808-8701　　FAX: 06-4808-8702　　E-mail: infodesk@vstone.co.jp

受付時間　：9:00～18:00（土日祝日は除く）

ヴイストン株式会社　　www.vstone.co.jp

〒555－0012　大阪市西淀川区御幣島 2-15-28