ブログ
BLOG

ROS2でSLAM入門１：ROS2インストールからマップ作成編

2021-06-25 14:00 技術ブログ

ショッピングセンターや空港を走る自律移動ロボットが多く開発されていますが、どういった技術で動いているかご存知でしょうか。 メインとなるのはSLAM(Simultaneous Localization and Mapping)と呼ばれる技術です。 センサ情報などから「自己位置推定」と「環境地図作成」を同時に行うというものです。 この技術を使用すれば、ロボットにとって未知の場所でも自分の位置を認識して走行できます。 以下にSLAMの入力と出力のイメージを示します。

このような自律移動ロボットを開発する場合、ROSというロボット制御のミドルウェアを活用することが多いと思います。 ROSの基本的な説明はこちらにまとめました。 また最近は、ROSからROS2への移行が進んできています。

今回のブログでは、これらの技術を体験するために、次の内容について解説していきます。

- ROS2のセットアップ
- シミュレーション用ロボットデータのインストール
- シミュレータでSLAMの実行（地図の作成）

＊ ROS2のセットアップ

まずはROS2をインストールします。環境は以下のとおりです。

- OS：Ubuntu 20
- ROS2 version：foxy

公式チュートリアルを踏襲して、インストール方法を解説します。

ターミナル(端末)を開いてください。（ターミナルはデスクトップを右クリックすると出てくるメニューの「端末で開く」で開けます）

必要ツールのインストールとダウンロードサーバの登録を行うために、以下のコマンドを実行してください。

注意１ 以降行頭に$(ダラー)があるときは、ターミナル上でのコマンドを示します。

注意２ 先頭がsudoで始まる時は、ユーザパスワードを要求されます。Ubuntu20インストール時に設定したパスワードを入力してください。

```
$ sudo apt update && sudo apt install curl gnupg2 lsb-release
$ sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key
-o /usr/share/keyrings/ros-archive-keyring.gpg
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-
archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main"
| sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

以下のコマンドでROS2のインストールを実行します。

```
$ sudo apt update && sudo apt install ros-foxy-desktop
```

以下のコマンドでROS2の初期設定を行います。即ち、ROS2関連のパスが自動で通るように、.bashrcファイルの末尾へ「source /opt/ros/foxy/setup.bash」を追記しています。 （.bashrcファイルはターミナルを開く度に自動で実行されるスクリプトファイル）

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

最後に以下のコマンドで、追加のツールとこの後必要になるパッケージをインストールします。

```
$ sudo apt install git python3-argcomplete python3-colcon-common-extensions
python3-rosdep
$ sudo apt install ros-foxy-gazebo-* ros-foxy-turtlebot3 ros-foxy-turtlebot3-msgs
```

ROS2の依存関係を解決するために、下記のコマンドを実行してください。

```
$ sudo rosdep init
```

以上でインストールは完了です。とりあえずサンプルの動作を確認してみましょう。

ターミナルを全て閉じてください（閉じないと.bashrcへの変更が有効になりません）。

新たに２つターミナルを立ち上げてください。

各ターミナルで、以下のコマンドを実行してください。 実行した結果、以下のような文字列が出力されていれば、ROS2が正常にインストールされています。

ターミナル1

```
$ ros2 run demo_nodes_cpp talker
[INFO] [1623807368.825666956] [talker]: Publishing: 'Hello World: 1'
[INFO] [1623807369.825594312] [talker]: Publishing: 'Hello World: 2
```

ターミナル2

```
$ ros2 run demo_nodes_py listener
[INFO] [1623807399.114323428] [listener]: I heard: [Hello World: 1]
[INFO] [1623807400.103016926] [listener]: I heard: [Hello World: 2]
```

コマンドを停止する時は、ターミナルをアクティブにしてCtrl+cキーを押下してください。

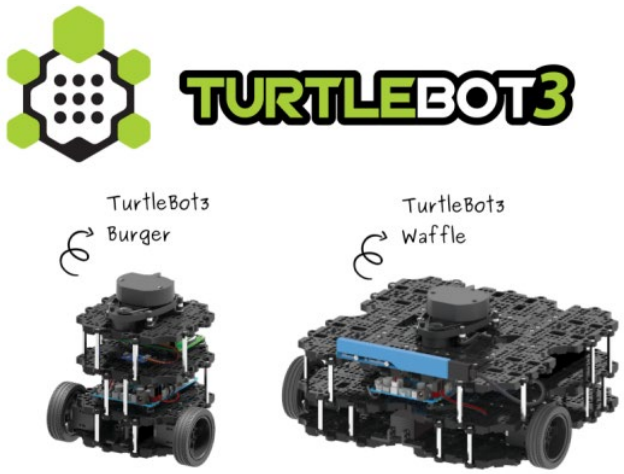
また、シミュレータが起動することを確認しておきます。以下のコマンド実行後に空のシミュレータが立ち上がればOKです。

```
$ gazebo
```

＊ シミュレーション用ロボットデータのインストール

シミュレーション用のロボットとしてTURTLEBOT3のモデルを使用します。

TURTLEBOT3は、差動2輪型のロボットで搭載マイコンから簡単に制御が行えます。また、LRFやカメラなどのセンサを搭載すれば制御に加えて環境認識も行えるようになり、学生向けロボット教材や研究機関でのアルゴリズムテスト用途として広く普及しています。



TURTLEBOT3のチュートリアル を踏襲して、インストール方法を解説します。

ターミナルより以下のコマンドを実行することで、ROS2のワークスペース（turtlebot3_ws）の作成と、そこへのプログラム等のダウンロード(git clone)が行われます。

```
$ mkdir -p ~/turtlebot3_ws/src/
$ cd ~/turtlebot3_ws/src/
$ git clone -b foxy-devel
https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

次に依存ライブラリのインストールとビルドを行います。

```
$ rosdep update
$ cd ~/turtlebot3_ws
$ rosdep install --from-paths src --ignore-src --rosdistro foxy -y
$ colcon build --symlink-install
```

エラーなく、finishedになればビルド完了です。

ワークスペースへのパス設定行うために、以下のコマンドで再度.bashrcへ追記を行います。

```
$ echo 'source ~/turtlebot3_ws/install/setup.bash' >> ~/.bashrc
$ echo 'export
GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/turtlebot3_ws/src/turtlebot3/turtlebot3_simul
 >> ~/.bashrc
$ echo 'export TURTLEBOT3_MODEL=burger' >> ~/.bashrc
```

追記後は、ターミナルを一旦全て閉じてください。（閉じないと.bashrcへの変更が有効になりません）。

再度ターミナルを開き、下記のようにcatコマンドで追記されていることを確認できればOkです。

```
$ cat ~/.bashrc
・
・
・
```

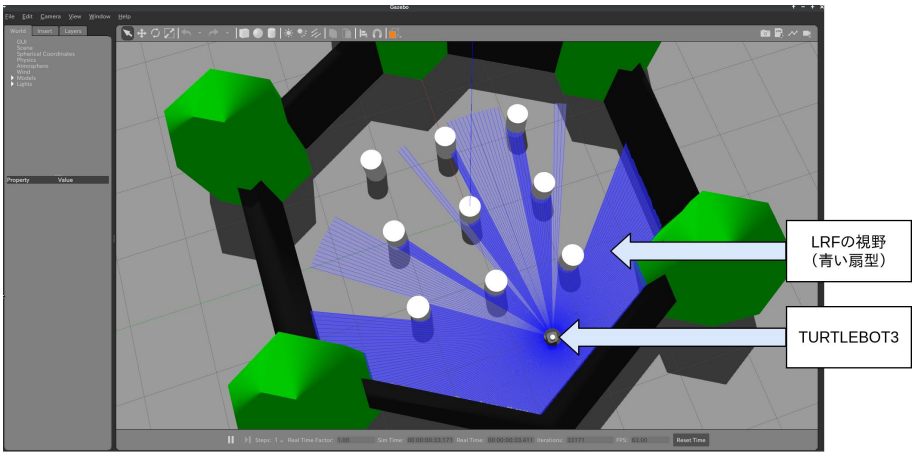
```
source /opt/ros/foxy/setup.bash
source ~/turtlebot3_ws/install/setup.bash
export
GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/turtlebot3_ws/src/turtlebot3/turtlebot3_simul

export TURTLEBOT3_MODEL=burger
```

＊　シミュレータでSLAMの実行（地図の作成）

＊　シミュレータ(gazebo)起動

いよいよ、シミュレーションを開始します。シミュレーションにはROS2に付属しているgazeboを使用します。下記のコマンドを実行して、以下の写真のような画面がでることを確認します。

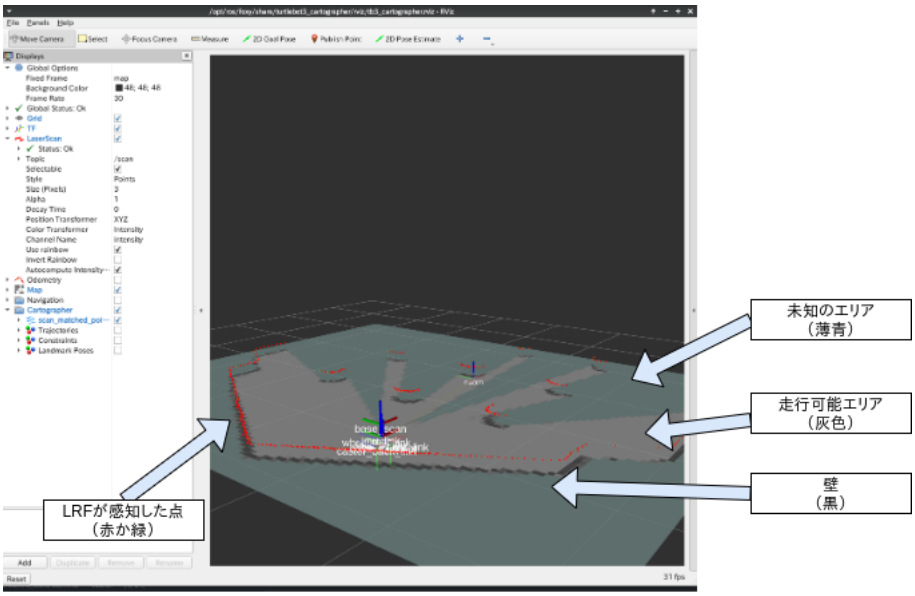
```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```



＊　SLAM起動

次に別ターミナルを開いてください。そこで下記のコマンドを実行して、写真のようにRviz2（可視化ツール）の画面が表示されたら、成功です。

```
$ ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=True
```

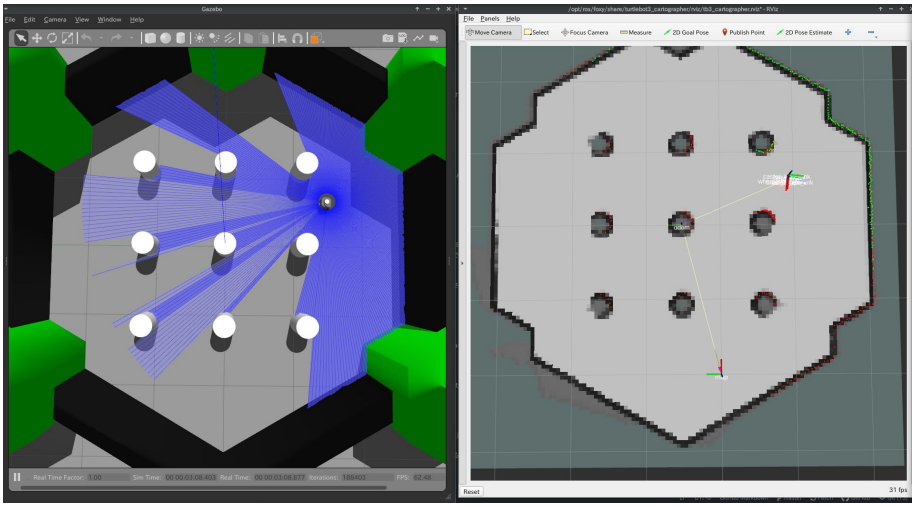


画面上ではSLAMによって、すでに地図が作成されはじめています。

＊　地図の作成

キーボードでTURTLEBOT3を動かして、地図を完成させます。 以下のコマンド入力後、そのままWキーで前進、Xキーで後退、Aキーで左旋回、Dキーで右旋回を行えます。 止める時はSキーを押下します。

```
$ ros2 run turtlebot3_teleop teleop_keyboard
```



TURTLEBOT3をシミュレータ上で満遍なく動すと、右上図のような地図が完成します。またSLAMの自己位置推定の機能により、シミュレータ上とRviz2上のそれぞれのTURTLEBOT3の位置が合っている事がわかります。

＊　地図の保存

最後に、次回の自律移動へ向けて地図を保存します。 コマンド実行後、ホームディレクトリ直下に「map.yaml」と「map.pgm」が保管されていればOKです。

```
$ ros2 run nav2_map_server map_saver_cli -f ~/map
```

＊　終了

地図を保存したら、各コマンドを実行したターミナル上でCtrl+cキーを押下して、アプリ停止させてください。

＊　上手く動かないときは・・

慣れないうちは~/.bashrcへの書き込みがされていないケースが多いと思います。今回の例では４行追加されていることを確認してください。また、書き換え後は、一旦ターミナルを閉じてから、再度開いてください。

＊　まとめ

ROS2でSLAMのサンプルプログラムを動かして、地図の作成を行いました。 SLAM自体は確率、統計、グラフ理論などなど難しい数学を用いたアルゴリズムですが、 とりあえずの動作確認だけならば、今回のように簡単に行えます（ありがたいことに）。 次回は、SLAMと対を為す技術である「自律移動」のサンプルを動かしたいと思います。

次回→ROS2でSLAM入門２：自律移動編

＊　宣伝

ROS/ROS2を使うと自律移動型ロボットのソフトは意外と簡単に動きます。ただ、これを実運用で安定稼働させるには、ソフト以外に電気や機械など様々なノウハウが必要となります。 テクノロードではロボットハードの開発からソフトやクラウドまで一括して開発を請

け負うことが可能ですので、お気軽にご相談ください。 ROSを扱った開発事例はこちらをご覧ください。