講習会

ロボット用ミドルウェアを活用した自律走行ソフトウェア入門【ROS 編】

日　時　2020 年 12 月 3 日（木）10:00～17:00
2020 年 12 月 4 日（金）10:00～17:00

場　所　地方独立行政法人東京都立産業技術研究センター（本部）
東京都江東区青海 2-4-10
●ゆりかもめ「テレコムセンター」駅前
●りんかい線「東京テレポート」駅下車 徒歩 15 分［朝夕無料送迎バスあり 3 分］
都営バス海 01 テレコムセンター駅前下車

定　員　10 名

受講料　8,500 円

本講習会では、ロボット用ミドルウェア（ROS Melodic）を活用してロボットのソフトウェアを開発する手法について紹介し、その具体例として自律走行ロボットを開発します。

ロボット用ミドルウェアは、分割されたソフトウェアを、相互に通信させるための仕組みを提供します。自律的なロボットのソフトウェアは、ロボットが認識、計画、動作を行う際に、多数のソフトウェアを並列的に実行します。ロボット用ミドルウェアは、このような実行に際し、開発スピードを向上させるための重要なキーテクノロジーとなります。

自律走行ロボットの開発には、ROS で商用利用可能なオープンソースを用いて作成します。さらに機能拡張や改良を行うことで、ROS の基礎と用途に応じた自律走行ソフトウェアの開発が習得できます。

本講習会では、受講者が Python または C++言語によるプログラミングを行えることを想定しています。講習で使用するソフトウェア環境を持ち帰られたい場合には、事前にご相談下さい。必要機材をご紹介致します。

【新型コロナウイルス感染防止対策へのご協力のお願い】
ご来場の際には必ずマスクの着用および弊センター備え付け消毒液で手指消毒のご協力をお願いいたします。

月　日	時　間	科　　目	講　　師
12/3（木）	10:00～12:00	ロボット用ミドルウェアROSの概要とROS関連ツールの使い方	東京都立産業技術研究センター ロボット開発セクター　職員
	13:00～15:00	実機とシミュレータ(Gazebo)での移動ロボット制御	
	15:00～17:00	自律走行の概要+地図作成(SLAM)	
12/4（金）	10:00～12:00	自律走行制御(自己位置推定+経路計画+走行制御)	
	13:00～15:00	PythonとC++によるROSプログラミング	
	15:00～17:00	自律走行の改良	

＊各科目の講習の途中で休憩時間を設けております。

地方独立行政法人
東京都立産業技術研究センター
TOKYO METROPOLITAN INDUSTRIAL TECHNOLOGY RESEARCH INSTITUTE

1. ROSの概要

1-1. 自律走行ロボットの用途
1-2. 自律走行に必要なソフトウェア
1-3. ロボット用ミドルウェアについて
1-4. ROSとは
1-5. ROSが提供してくれるもの
1-6. ROSの長所・短所
1-7. ROSのディストリビューション
※. LinuxとUbuntu
※. ソフトウェアライセンスについて
※. 本講習会で扱うパッケージのライセンス

自律走行ロボットは、何ができる?どうやって作る?
ROSとは何か?なぜROSを使うのか?

2. ROSの基礎とROSコマンド

2-1. ROSの動作原理
2-2. ROSを動かしてみる(turtlesim)
2-3. ROSの通信方式
演習① ノードとトピックの情報を調べる
演習② サービスの情報を調べて使ってみる
演習③ 自分でノード間通信をつなげてみる
2-4. launchシステムを使って複数のノードを動かす
演習④ launchファイルを書いてみる

ROSは基本的にコマンドライン操作、ROSコマンドを覚えよう！
オープンソースパッケージを利用するにはどうすればよい?

1-3. ロボット用ミドルウェアについて

- 従来の開発スタイル
  - 単一の大きなプログラムとして開発
    ⇒再利用性がない(センサの)

- ロボット用ミドルウェアを用いた開発スタイル
  - プログラムを部品化し、通信インタフェースを共通化
    ⇒同じインタフェースであれば、部品の入れ替えが簡単

ロボット用ミドルウェアを用いることでソフトウェアが部品化でき、開発効率が向上

1-4. ROSとは

ROS

- ROSとは、Robot Operating Systemの略で、ロボットソフトウェアプラットフォーム（ミドルウェア）の1つ。
- ROSの特徴
  - 世界で最も利用されているロボットミドルウェア
  - 便利な開発ツール・ライブラリが豊富(Gazebo, rviz等)
  - コミュニティの議論が活発
  - インターネットに情報が豊富
  - サポート言語：C++,Python
  - 管理団体：OSRF(Open Source Robotics Foundation)
  - ライセンス：BSD(商用利用可能)
  - 対応OS：Linux

GAZEBO　RViz

近年では、AWSやMatlab連携やROS-Industrial等の団体も発足している
ユーザ数が多く、関連ツールが豊富なため最も利用されている

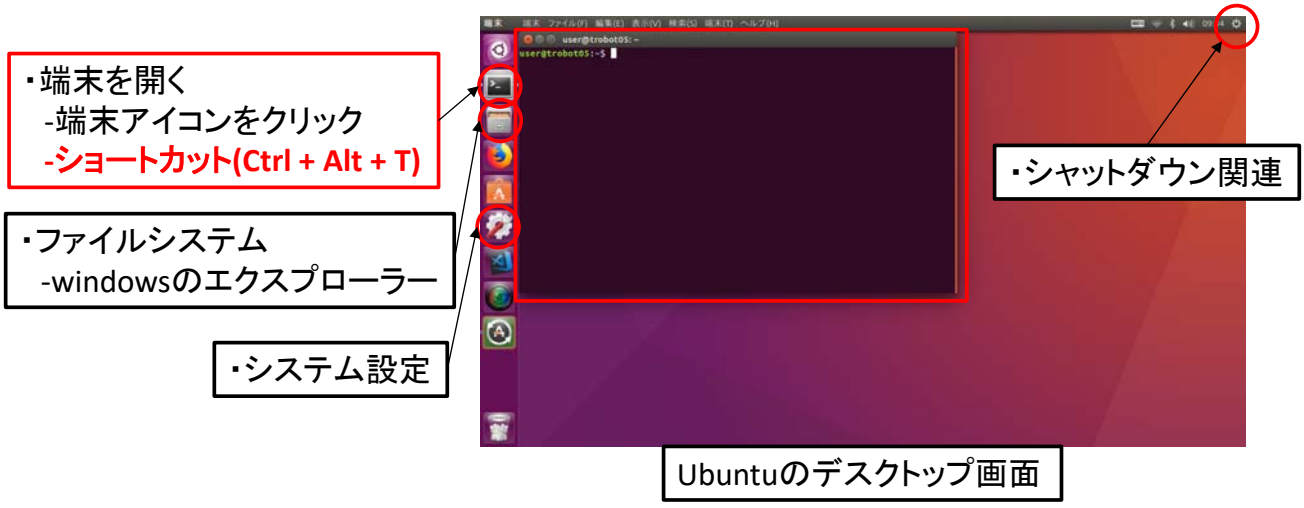
1-7. ROSのディストリビューション

バージョン名	リリース日	サポート終了	備考
ROS1(Kinetic)	2016/05/23	2021/04	Ubuntu16.04
ROS1(Melodic)	2018/05/23	2023/05	Ubuntu18.04、有名パッケージは対応済み
ROS1(Noetic)	2020/05/23	2025/05	Ubuntu20.04、ROS1の最終バージョン
ROS2(Dashing)	2019/05/31	2021/05	Ubuntu18.04、ROS2(初の長期メンテ)
ROS2(Eloquent)	2019/11/22	2020/11	Ubuntu18.04
ROS2(Foxy)	2020/06/05	2023/05	Ubuntu20.04、ROS2(最新バージョン)

- 更新が早く、ディストリビューション間の互換性がない場合もあるので注意が必要
- 今回は、利用ユーザー数とサポート期間から
  ubuntu18.04 + ROS1（Melodic）
  を対象として行う。

LinuxとUbuntu

- LinuxはwindowsとOS(オペレーションシステム)
- UbuntuはLinuxディストリビューションの1つ



ROSは主にubuntuで動作し、端末上でコマンド(文字)を用いて操作する

2-2. ROSを動かしてみる(turtlesim)

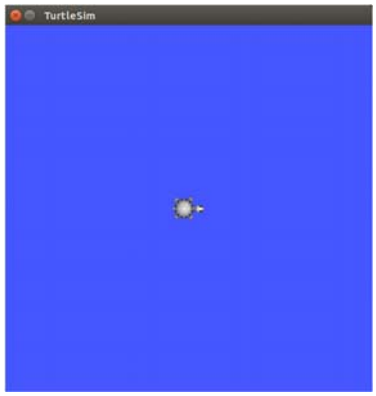
- ROSをインストールした時点で入っているパッケージturtlesimを動かしながらROSの基礎を学ぶ

- turtlesimを動かす

① 端末を開く
② `$ roscore`
③ 端末を開く
④ `$ rosrun turtlesim turtlesim_node`
⑤ 端末を開く
⑥ `$ rosrun turtlesim turtle_teleop_key`
⑦ 各端末でCtrl + Cを入力し終了する

Tabキーで入力すると楽!!

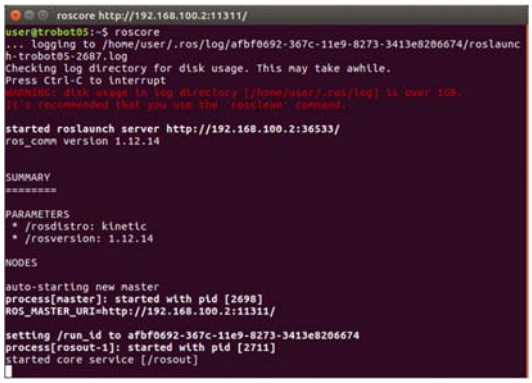
↑
←↓→
キーボードの矢印キーで操作する



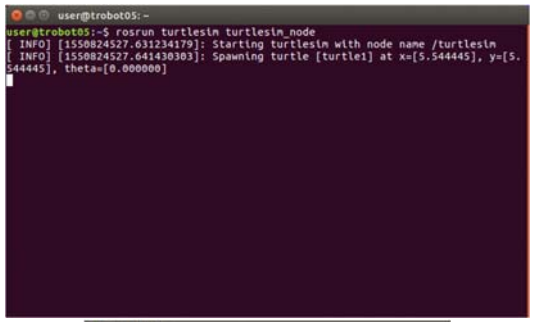
以上の手順(①～⑥)でキーボード入力でカメのアイコンが移動する

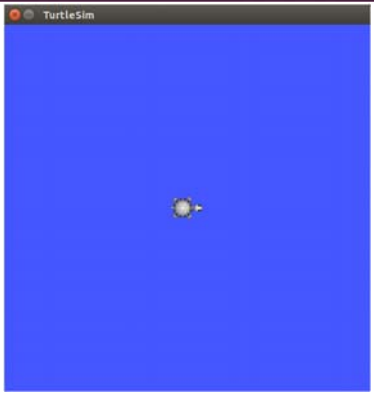
2-2. ROSを動かしてみる(turtlesim)

①②

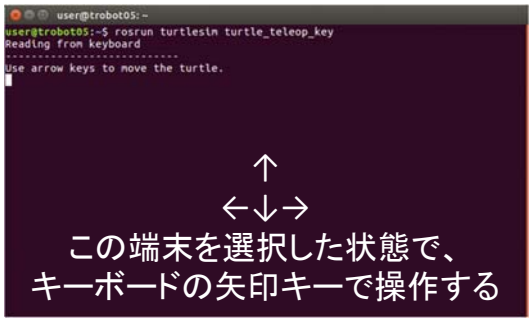


③④



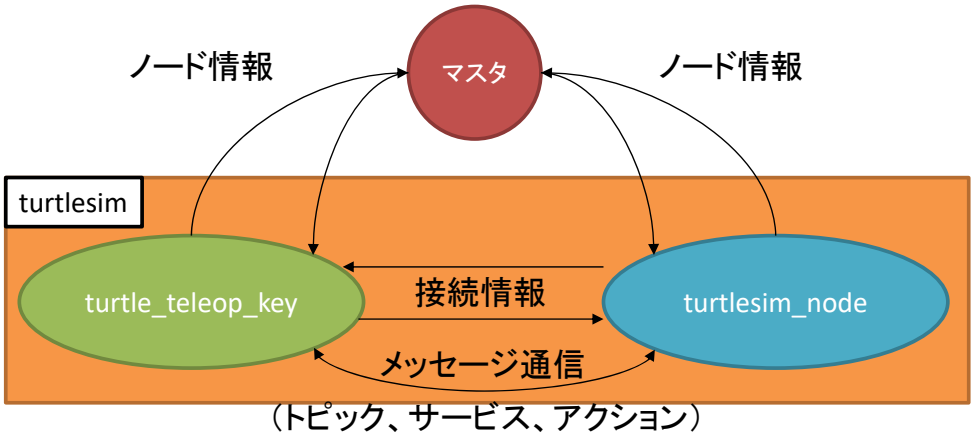


⑤⑥



⑦各端末でCtrl + Cを入力し終了する

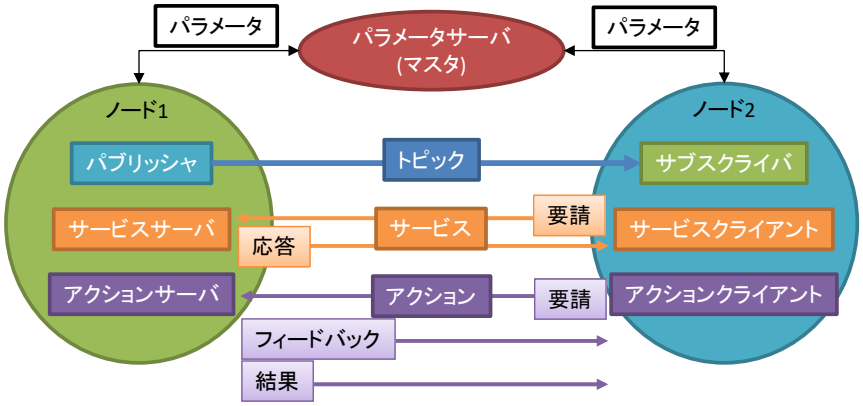
2-2. turtlesimの動作原理



ROSコマンド	説明
roscore	マスタを起動する
rosrun [パッケージ名] [ノード名]	ノードを起動する
rosrun turtlesim turtlesim_node	速度指令で動くカメのシミュレータを起動する
rosrun turtlesim turtle_teleop_key	キーボード入力を速度指令に変換し出力する

ノードを起動するには、パッケージ名とノード名が必要

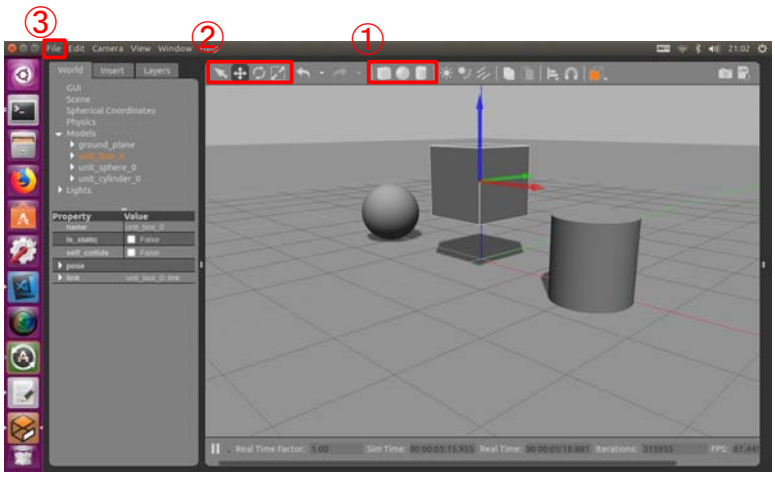
2-3. ROSの通信方式



ROS用語名称	説明
トピック	センサ等の連続的なデータ通信が必要な時(単方向非同期通信)
サービス	状態確認等の要請後に即時応答が必要な時(双方向同期通信)
アクション	目標地点への移動を指示された時など、要請後の応答に遅延がある場合や処理中の中間結果が必要な時(双方向非同期通信)
パラメータ	移動速度の最大値など、既定値を設定し頻繁に変更しない時

2-2. Gazeboを動かしてみる

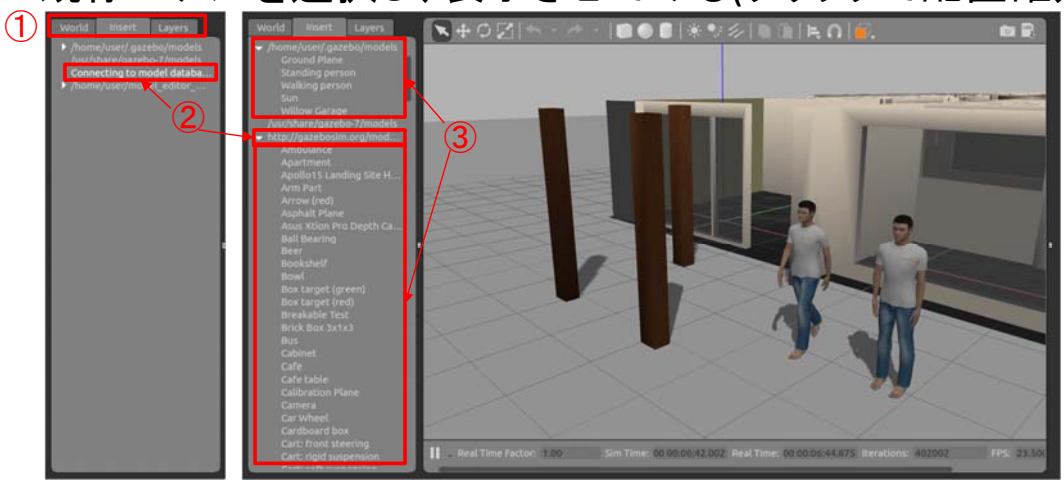
① 基本的なモデル(立方体・球体・円柱)を表示(クリックすると出現する)
② モデルを動かしてみる(下図のように選択し、動かしたい矢印をドラッグ)
③ 環境を保存する(.worldファイル)(Fileタブ→Save World As)



上図のように持ち上げて離すと物理シミュレータの重力により落下する

2-2. Gazeboを動かしてみる

① Insertタブを選択する
② Connecting to databaseがhttp://gazebosim.org/になるまで待つ
③ 既存モデルを選択し、表示させてみる(クリックで配置確定)

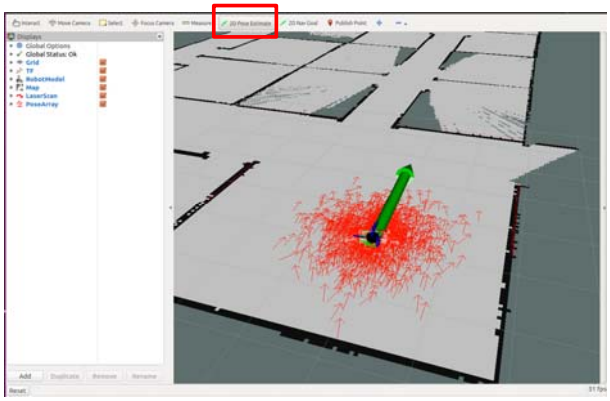


一度表示させたモデルはダウンロードされオフラインでも表示可能

1-4.シミュレータで自己位置推定を動かしてみる(amcl)

① 端末を開く

② `$ roslaunch trobot_sim navigation_trobot_sim.launch`

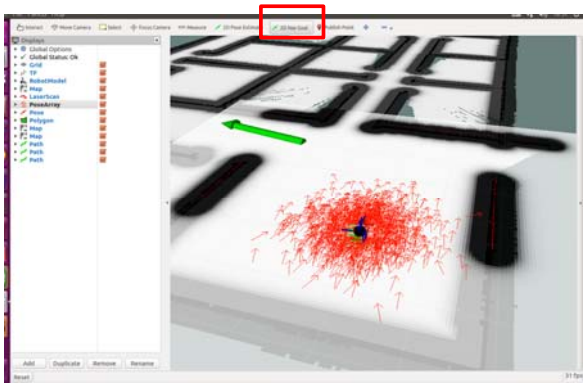


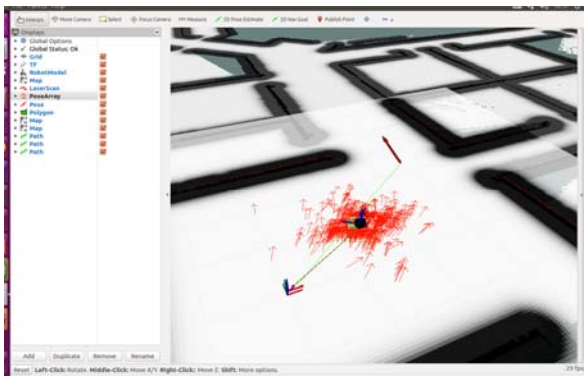
launchファイルの場所
~/catkin_ws/src/trobot_sim/launch/navigation_trobot_sim.launch
上記launcファイルで起動しているパッケージ(ノード)

1. trobot_sim:ロボットシミュレータ
   - gazebo_ros(gzserver)
   - gazebo_ros(gzclient)
   - gazebo_ros(spawn_model)
   - joint_state_publisher(joint_state_publisher)
   - robot_state_publisher(robot_state_publisher)
2. joy(joy_node):ジョイスティックの入力
3. teleop_twist_joy(teleop_node):速度指令変換
4. rviz(rviz):描画デバッグ
5. map_server(map_server):地図管理
6. amcl(amcl):自己位置推定
7. move_base(move_base):自律走行制御

2D Pose Estimateを選択し、クリックで位置指定し、その後ドラッグで方向を指定することで初期位置を設定する。
このとき、/initialpose [geometry_msgs/PoseWithCovarianceStamped]トピックを配信している

1-5.シミュレータで自律走行を動かしてみる(move_base)

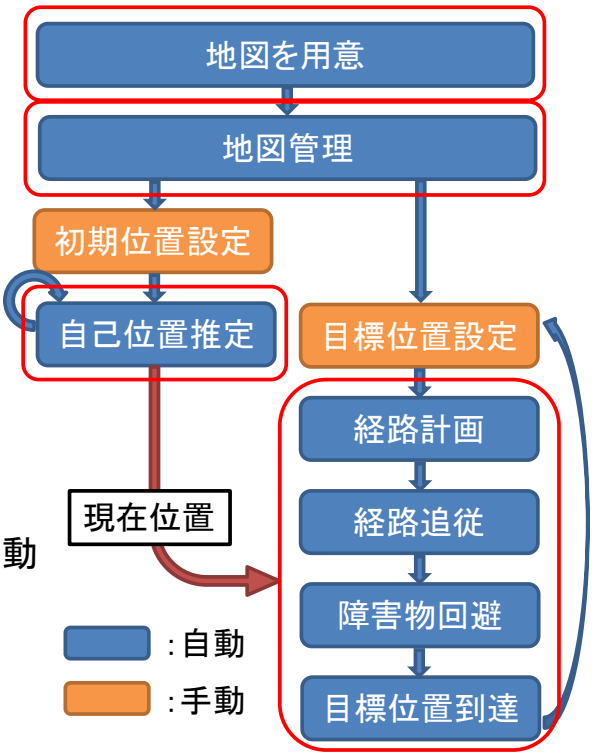




2D Nav Goal を選択し、クリックで位置指定し、
その後ドラッグで方向を指定するとロボットが目標まで移動する
このとき、/move_base_simple/goal [geometry_msgs/PoseStamped]トピックを配信している

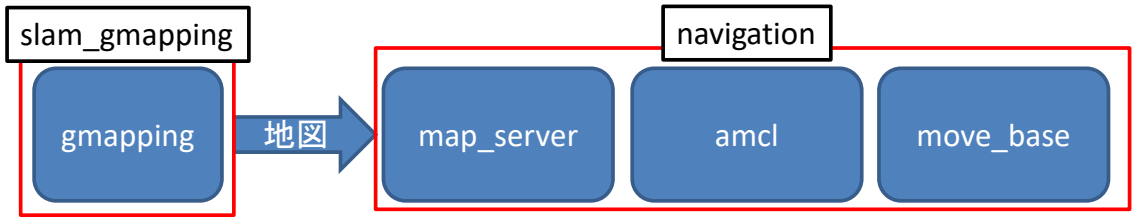
2-4.基本的な自律走行までの手順

① まず、地図を用意する
  – SLAMで地図作成
  – 手書きで地図作成
② 地図を読み込み管理する
③ 地図中の初期位置を設定(推定)する
④ (局所)自己位置推定し続ける
⑤ 目標位置を設定する
⑥ 地図上で(大域)経路計画を行う
⑦ 経路に沿って移動する(経路追従)
⑧ 障害物を回避しながら目的位置に移動
⑨ 目標位置に到達する(⑤に戻る)



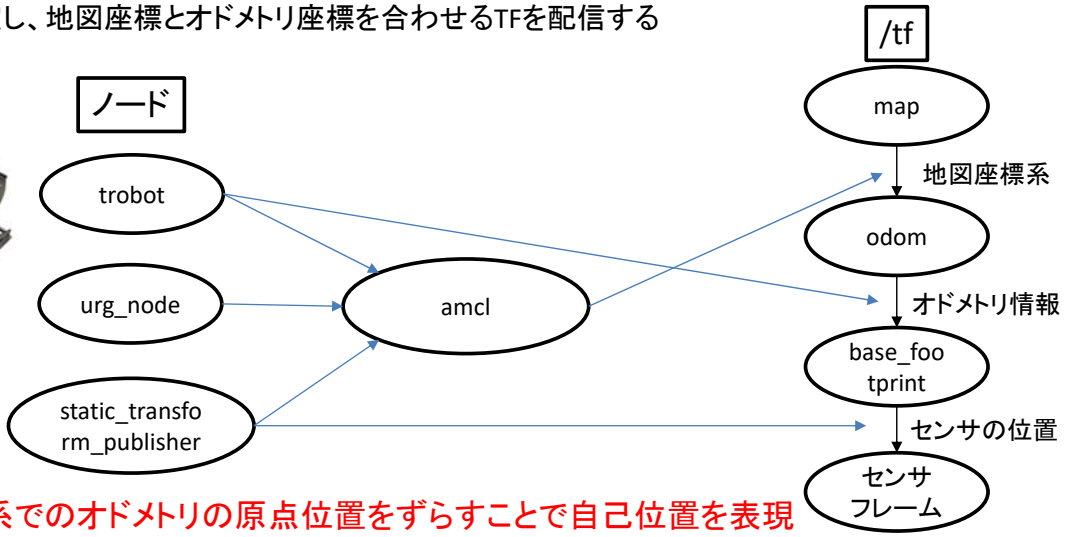
2-5.ROSのナビゲーション関連機能

- ROSの標準的なナビゲーション関連機能として「slam_gmapping」「navigation」の2つのメタパッケージが提供されている。
- 「slam_gmapping」は、レーザセンサとオドメトリを用いたSLAMによる地図作成機能を提供する。
- 「navigation」は、主に以下の3つで構成され、目的地まで自律走行する機能を提供する。
  – map_server :地図の保存と管理を行う
  – amcl :自己位置推定(最重要機能)
  – move_base :経路計画、経路追従、障害物回避等(巨大なパッケージ)



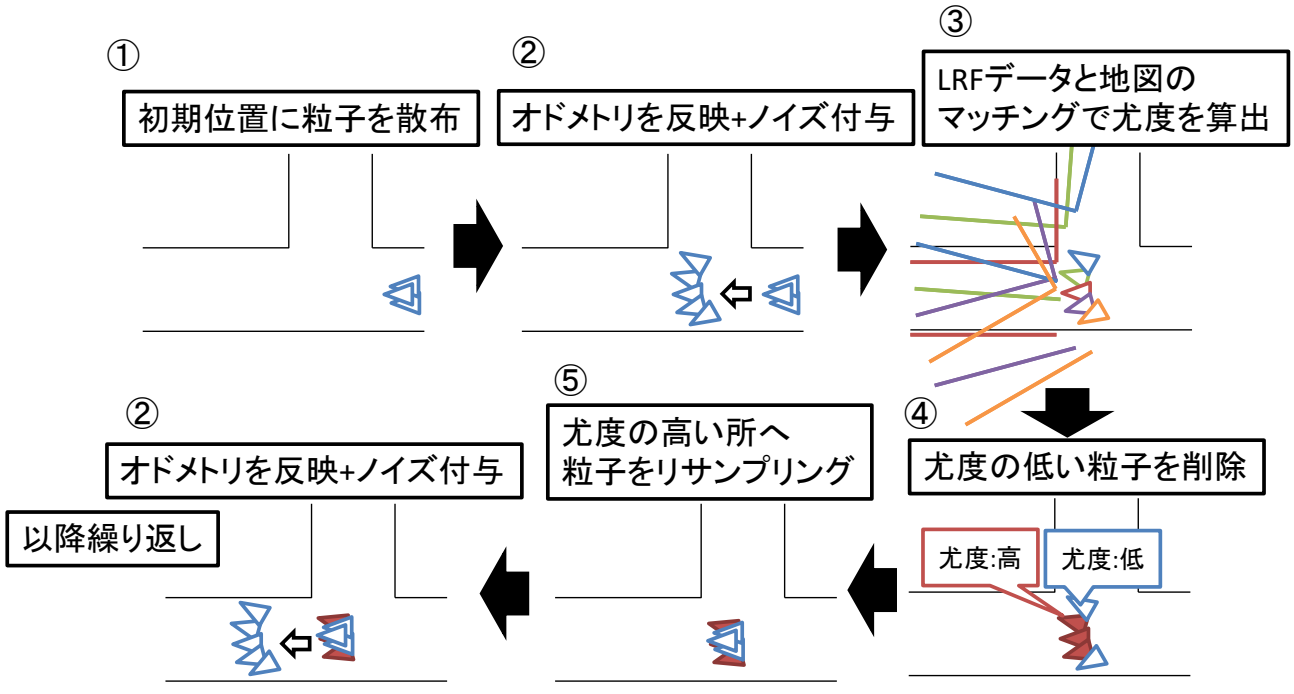
4-5. 自己位置推定時に必要なノード構成

- 最低限必要なノード
  - trobot/trobot_sim : geometry_msgs/Twist型で動作し、オドメトリ(TF)を配信するノード
  - urg_node : sensor_msgs/LaserScan型を配信するノード
  - static_transform_publisher : ロボットとセンサの位置関係(TF)を配信するノード
  - amcl : TF情報(オドメトリとセンサ位置)とsensor_msgs/LaserScan型を購読し、自己位置推定し、地図座標とオドメトリ座標を合わせるTFを配信する



地図座標系でのオドメトリの原点位置をずらすことで自己位置を表現

4-6. 自己位置推定(amcl)のアルゴリズム

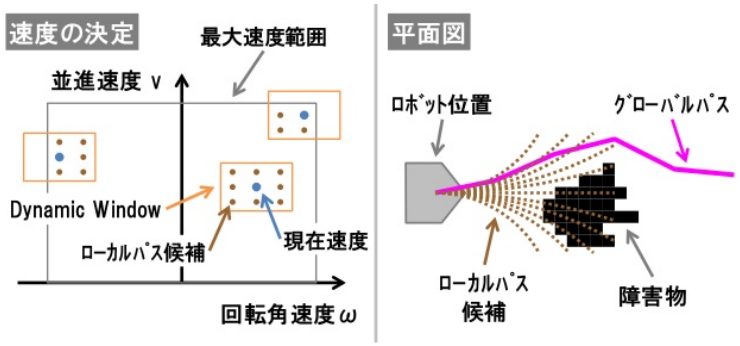


local_Planner

参照経路 + 局所的コストマップ + オドメトリを入力として受け取り、参照経路に追従しながら、障害物を回避する経路を生成する

DWA(ダイナミックウインドウアプローチ)

- 現在の速度付近でシミュレートして評価値が高い経路を選択
- 評価関数はグローバルパスに近いかつ障害物から遠いほど評価値が高い



Plannerの種類

move_baseは、プラグイン方式で実装されており、Global Planner(大域的経路計画)とLocal Planner(局所的経路計画)は入れ替えることが可能です。

move_baseノードのパラメータ(base_global_planner,base_local_planner)にパッケージ名/プラグイン名という形式で指定することで入れ替えできる。

Global Planner

パッケージ名	プラグイン名	方向付与	備考
navfn	NavfnROS	×	デフォルト
global_planner	GlobalPlanner	○	経路に向きの指定が可能

Local Planner

パッケージ名	プラグイン名	差動	全方向	ステア	
base_local_planner	TrajectoryPlannerROS	○	○	×	デフォルト
dwa_local_planner	DWAPlannerROS	○	○	×	上記の改良版
teb_local_planner	TebLocalPlannerROS	○	○	○	回避がなめらか

上記の他にも多くのプラグインがあり、自作も可能