

# 知能ロボット実習

第2回

地図作成と自己位置推定

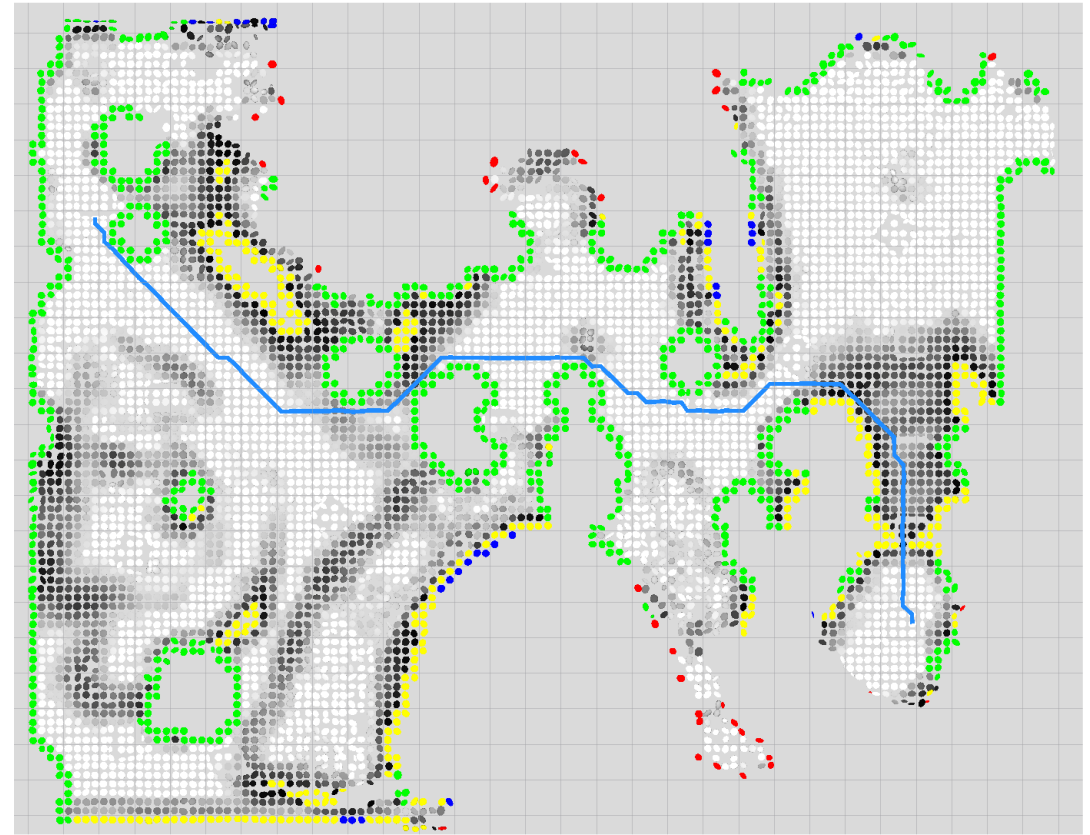
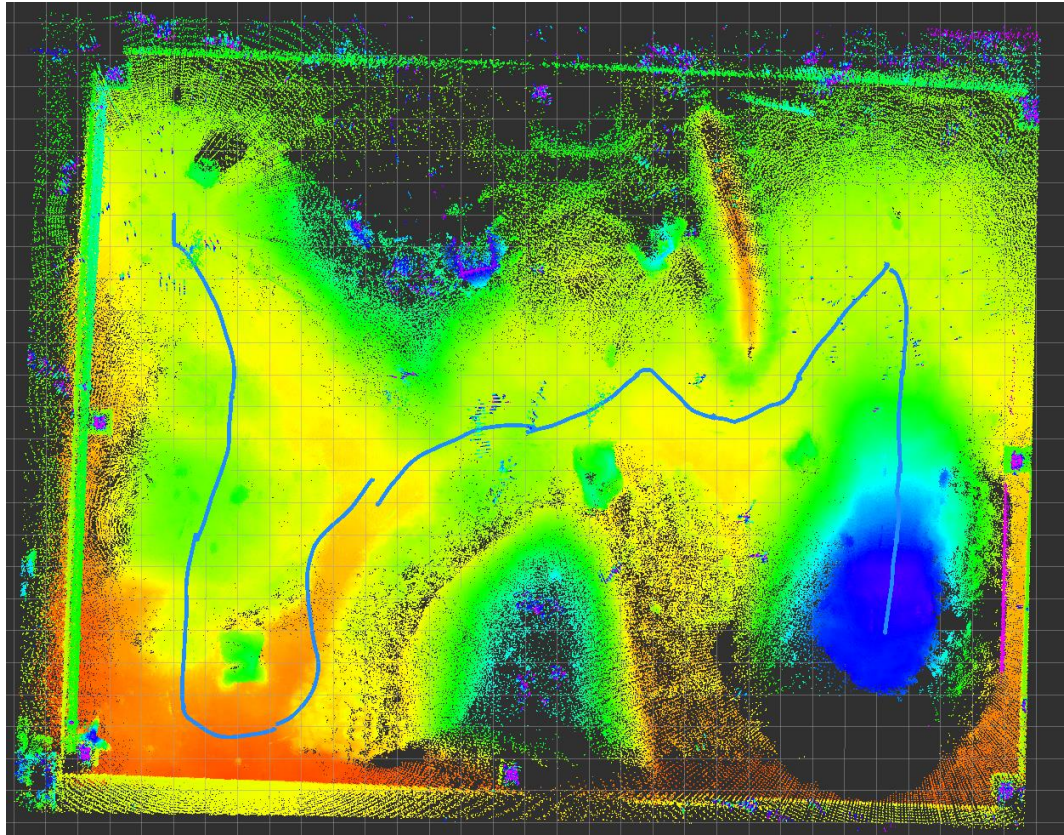
# 地図

- ロボットが自律ナビゲーションを実行するためには、環境の地図が必要
  - 地図と環境計測情報から自己位置を推定
  - 推定位置から目的地までの経路を生成

# SLAM(Simultaneous Localization and Mapping)

- 地図は必ずしもあるとは限らない
- 自己位置推定と環境地図作成を同時に行う
- SLAM (Simultaneous Localization and Mapping)と呼ぶ
- ロボットが未知の環境下にあっても自己位置を推定しながら、環境の地図を作成することができる

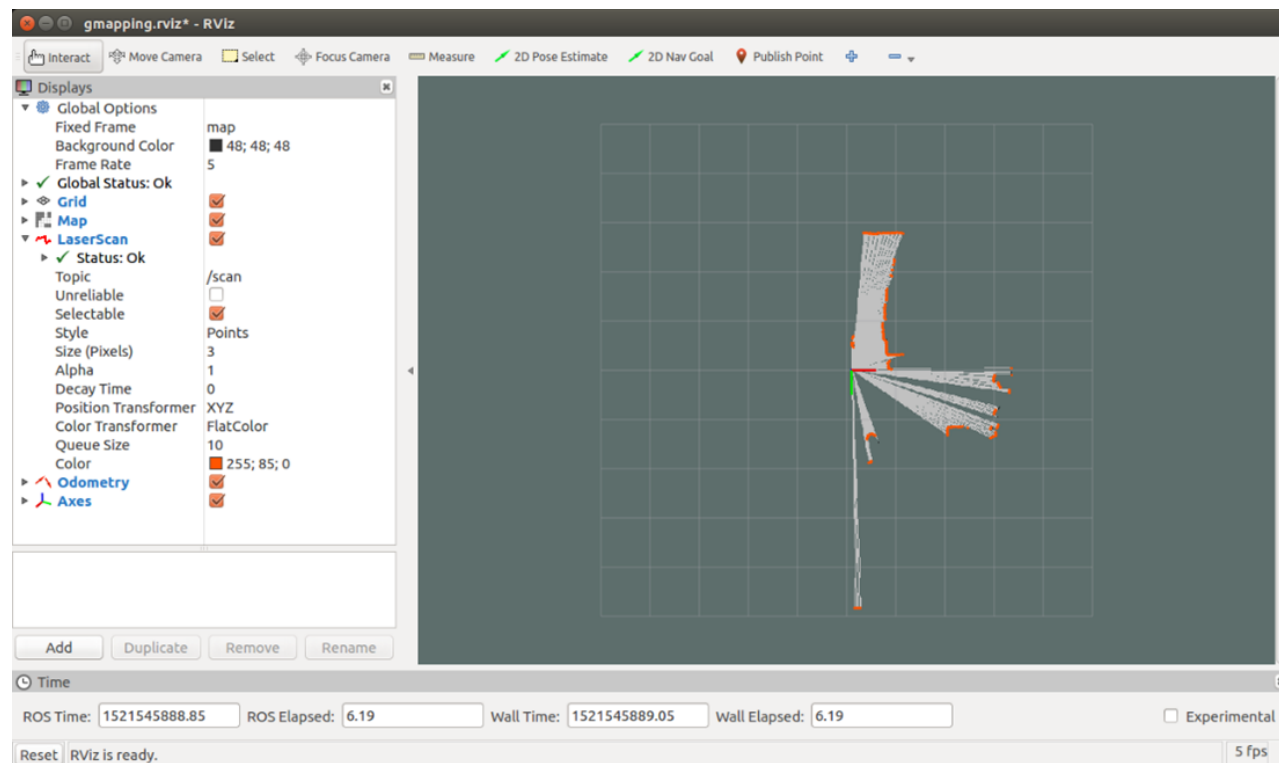
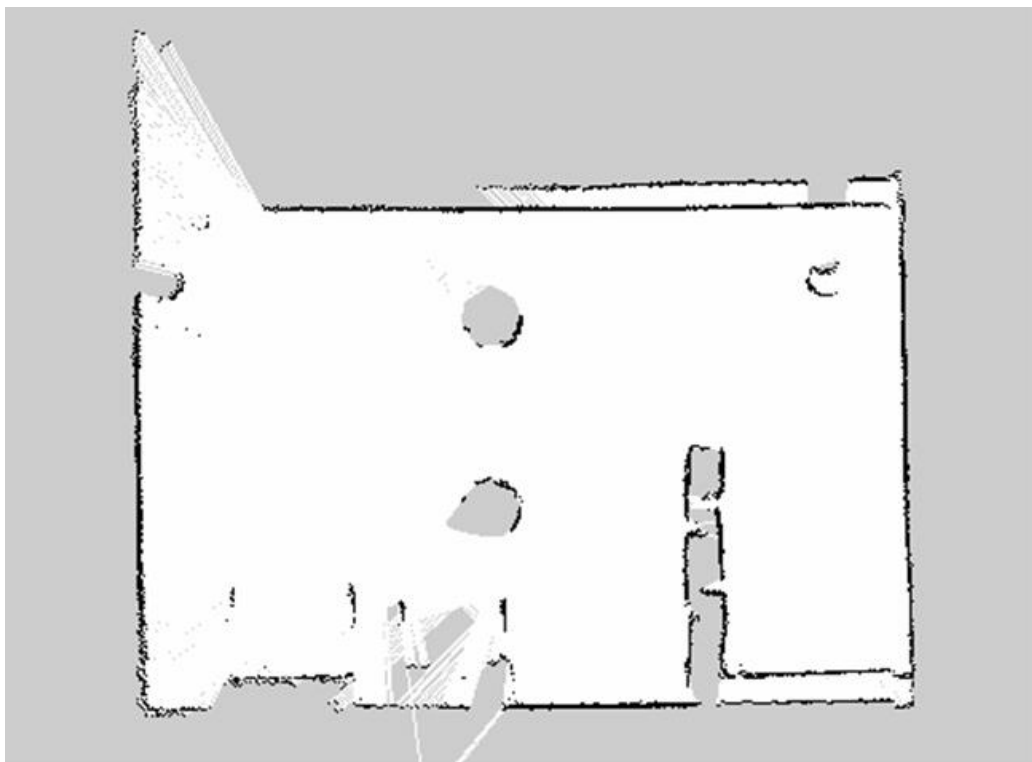
# ロボットにとっての地図



Schadler, M., Stückler, J., & Behnke, S. (2014). Rough terrain 3D mapping and navigation using a continuously rotating 2D laser scanner. *KI-Künstliche Intelligenz*, 28(2), 93-99.

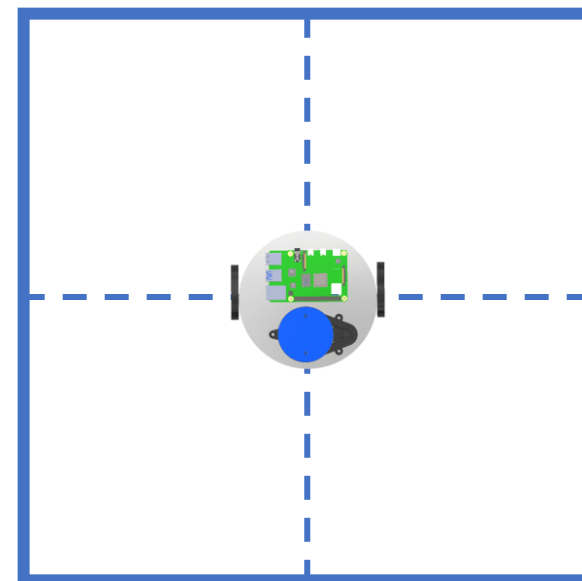
# SLAMパッケージ (gmapping)

- ROSにおけるSLAMパッケージの一つ
- LiDARデータとオドメトリデータを用いてSLAMを実行



# 課題 1 : 地図の作成

- ロボットの地図を遠隔操作によって作成せよ
- 地図作成の際の注意事項
  - ロボットの初期位置が原点になる
  - ロボットをホワイトボードの方向を向けて中心に置く
  - 地図が生成され始めたら，ロボットをゆっくりと移動させる
  - 移動した範囲の地図が徐々に作成されていくことを確認
  - 急発進や急停止，急旋回などを行うと地図作成に失敗するので注意



# 課題 1 : gmappingの実行

- ターミナル1

- `roslaunch lightrover_ros gmapping.launch`
  - ROSのプログラムを一斉に起動する方法 (launch)
  - LiDARと地図生成(gmapping)のプログラムを起動

- ターミナル2

- `sudo ds4drv`

コントローラとペアリング (Bluetoothで通信する)

- **PSボタン**と**SHAREボタン**を同時に押し続ける
- 白く点滅したら離す
- 青く点灯したら通信成立

各班ごとに設定  
します

- ターミナル3

- `roslaunch lightrover_ros pos_joycon.launch`

遠隔操作プログラムを起動

# 課題 1 : 地図の保存

- 地図の保存

- 地図が完成したら, 地図を保存する
- 新しいターミナルを起動
- `roslaunch map_server map_saver -f ファイル名`
  - ホームに地図の画像ファイル(.pgm)とデータファイル(.yaml)が保存される

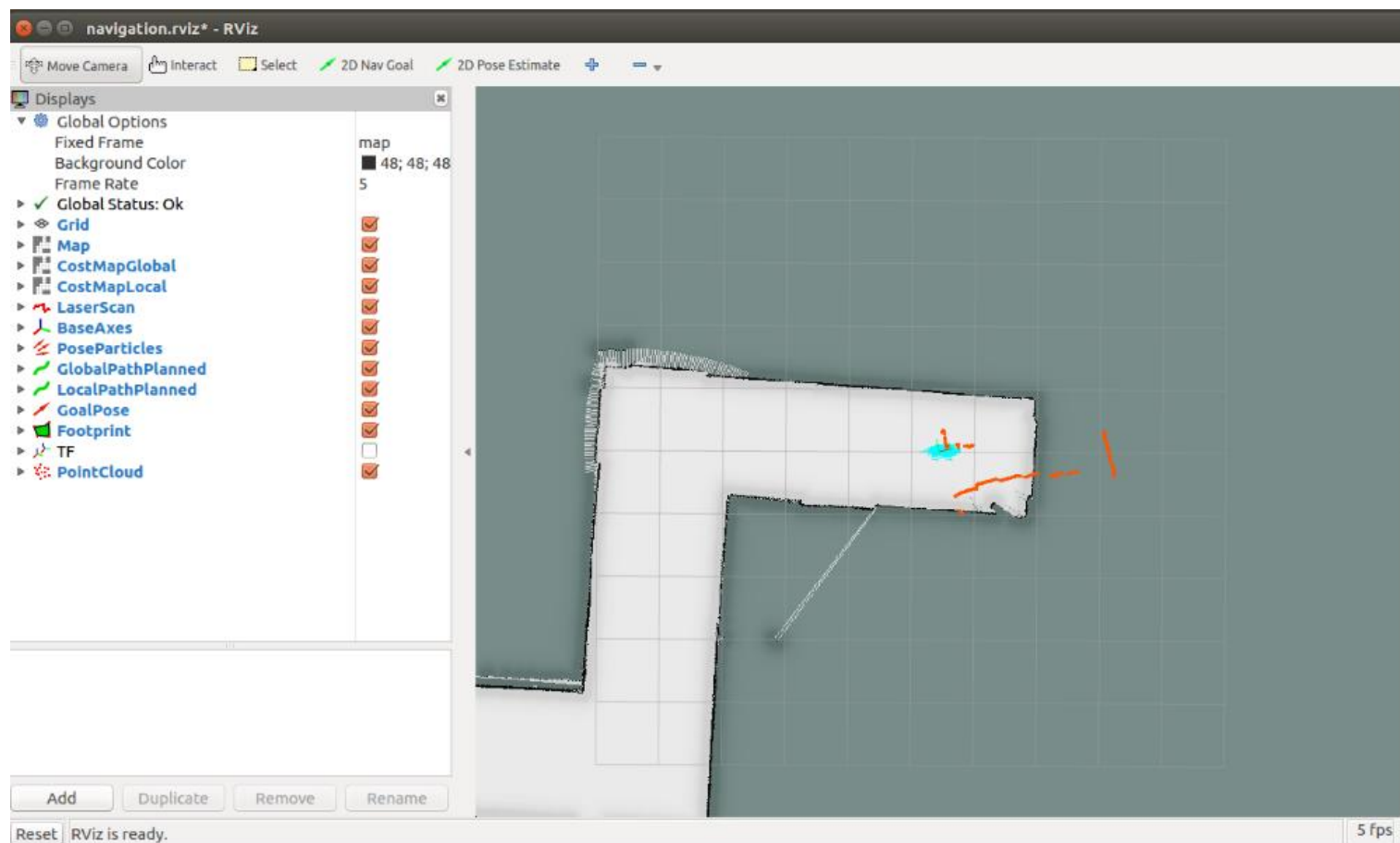
- 作成した地図 (～.pgm, ～.yaml) を以下に配置する  
`/home/robot/catkin_ws/src/lightrover_ros/map`



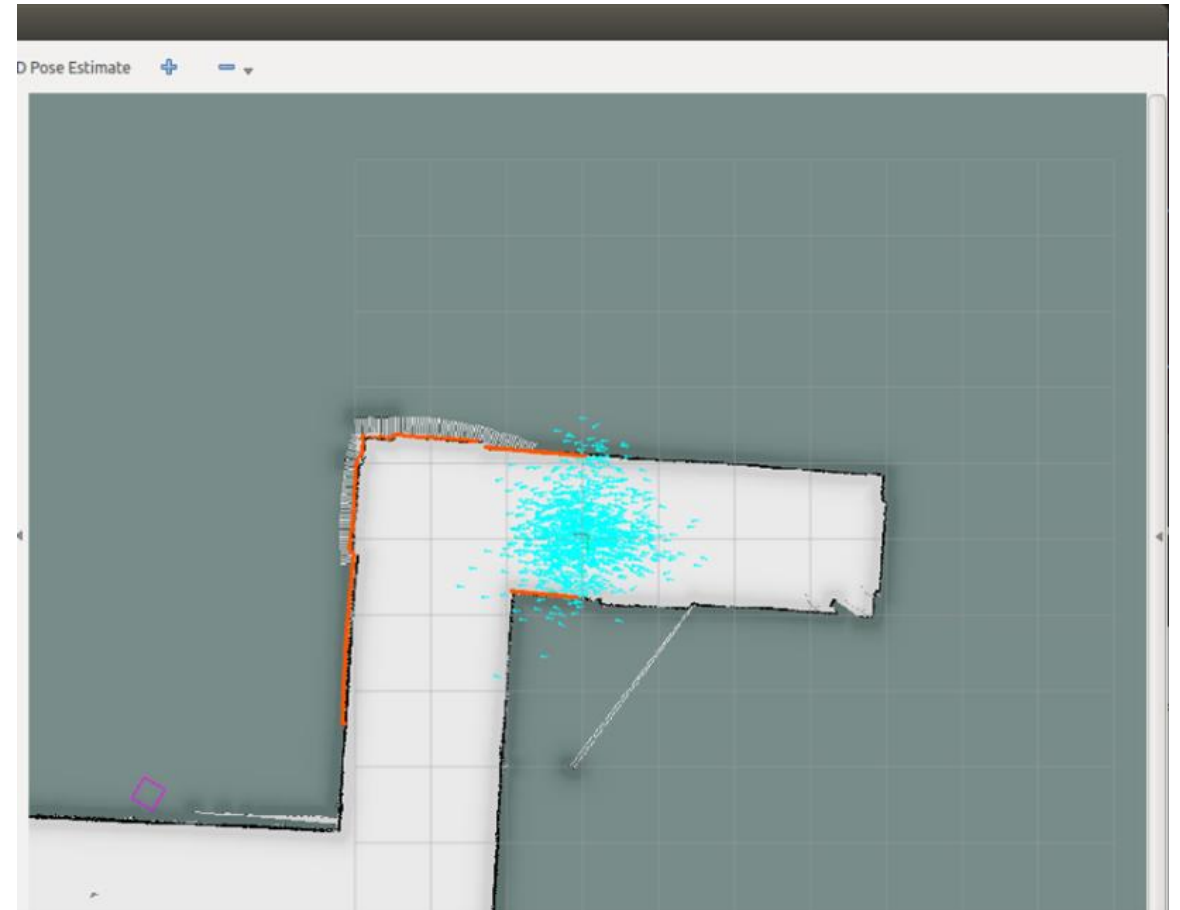
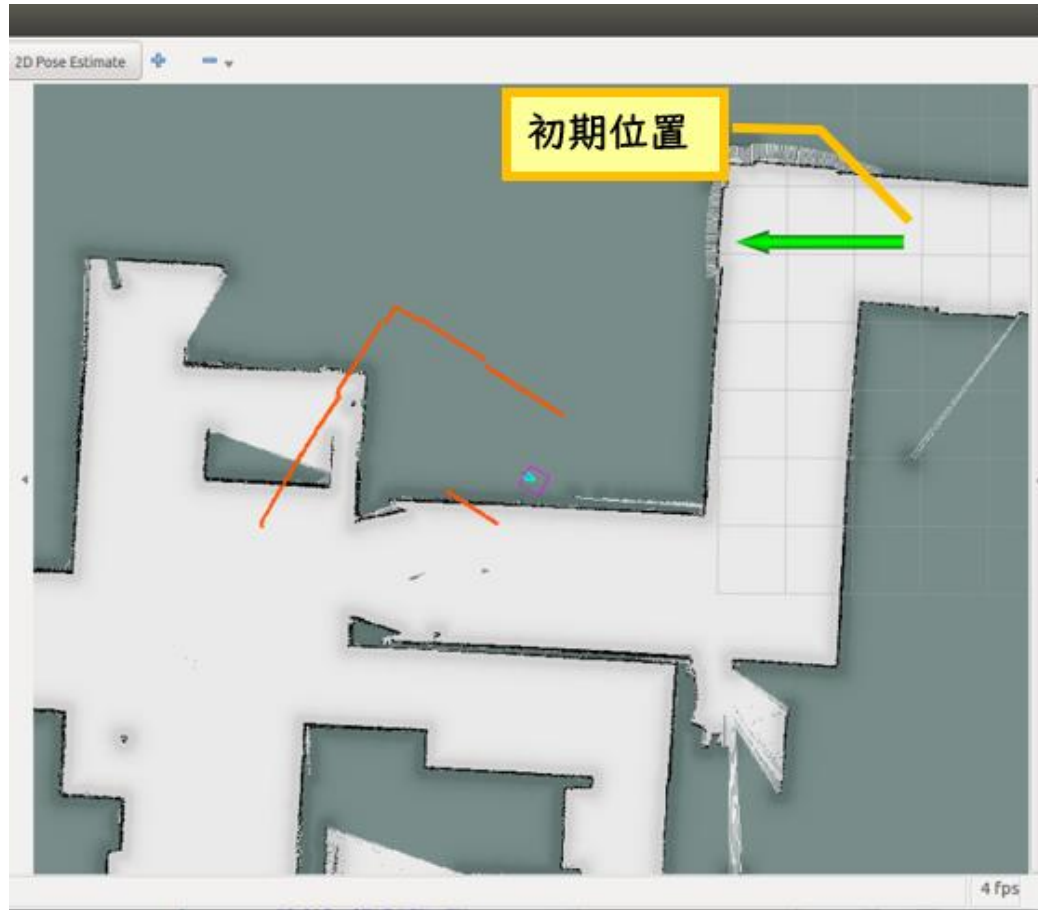
# 課題 2

- 作成した地図を用いてスタートからゴールまで自律ナビゲーションを実行せよ
- 実行ファイルを編集
  - /home/robot/catkin\_ws/src/lightrover\_ros/launch/lightrover\_move\_base\_dwa.launch
  - <arg name="map\_file" default="\$(find lightrover\_ros)/map/ファイル名.yaml" />
- ターミナルを起動
- roslaunch lightrover\_ros lightrover\_move\_base\_dwa.launchで実行可能

# 課題 2



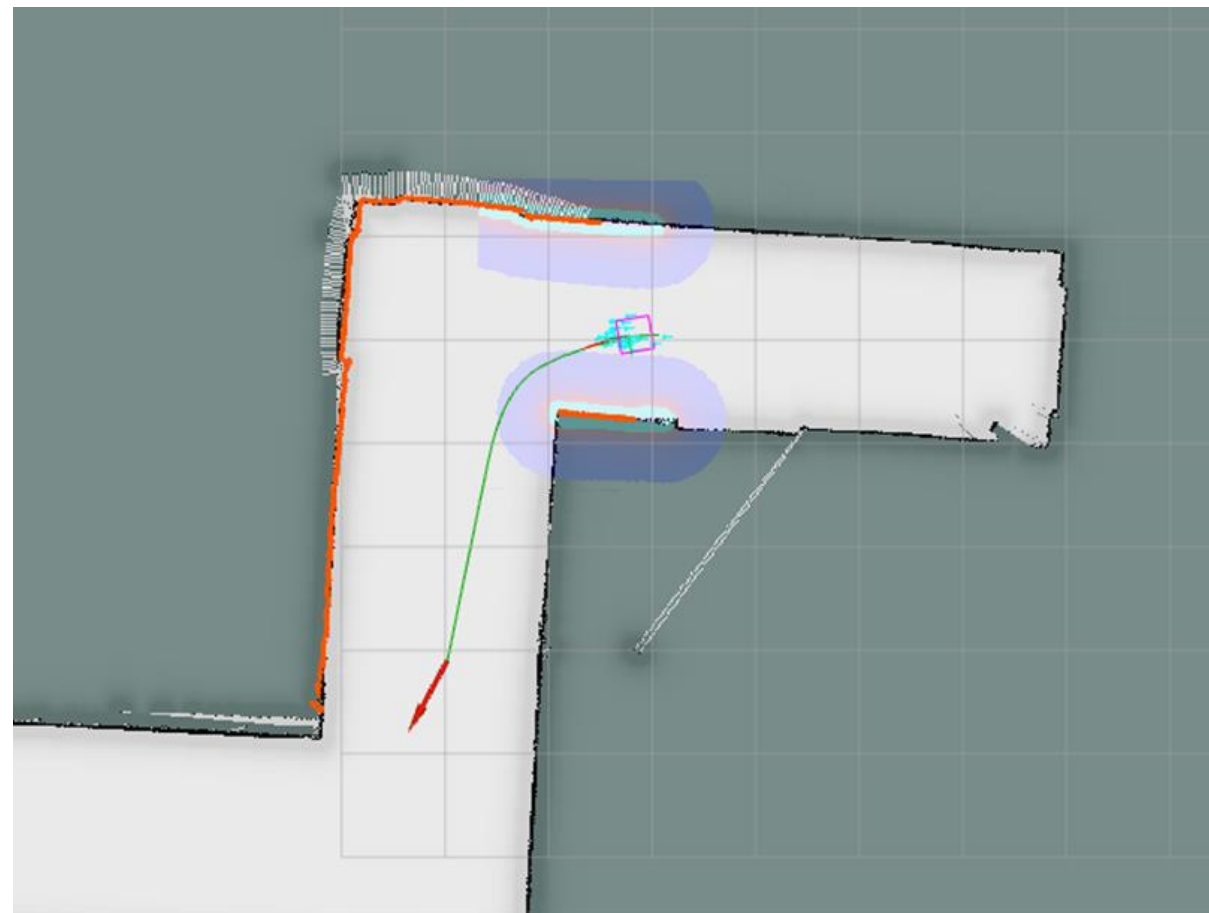
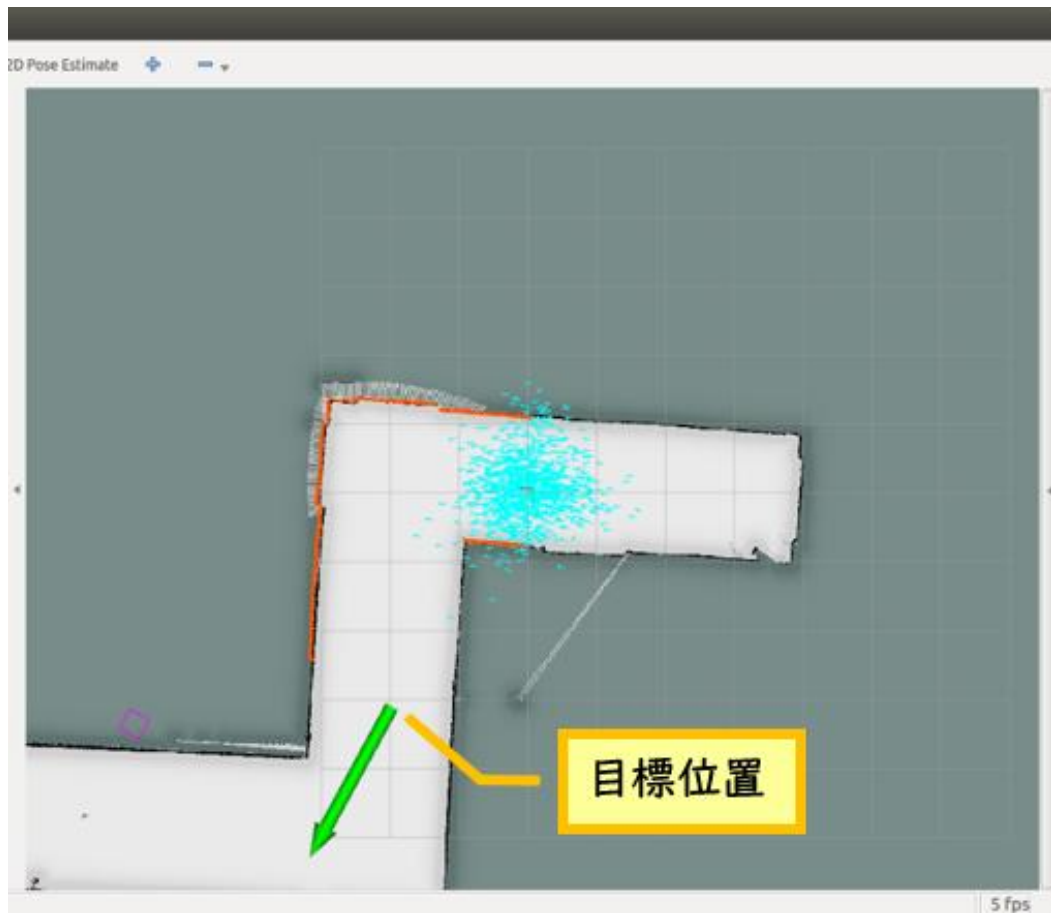
## 課題 2 : 初期位置の設定



画面上部の「**2D Pose Estimate**」ボタンをクリックしてから、ロボットの実際の初期位置をクリックし、ドラッグして方向を決定する

青い点は自己位置推定を担うノード「amcl」が計算するロボットの推定自己位置を示すパーティクルを表す

## 課題 2 : 目標位置の設定



画面上部の「2D Nav Goal」ボタンをクリックしてから、目標位置をクリックし、ドラッグして方向を決定（**ロボットが動くため注意**）

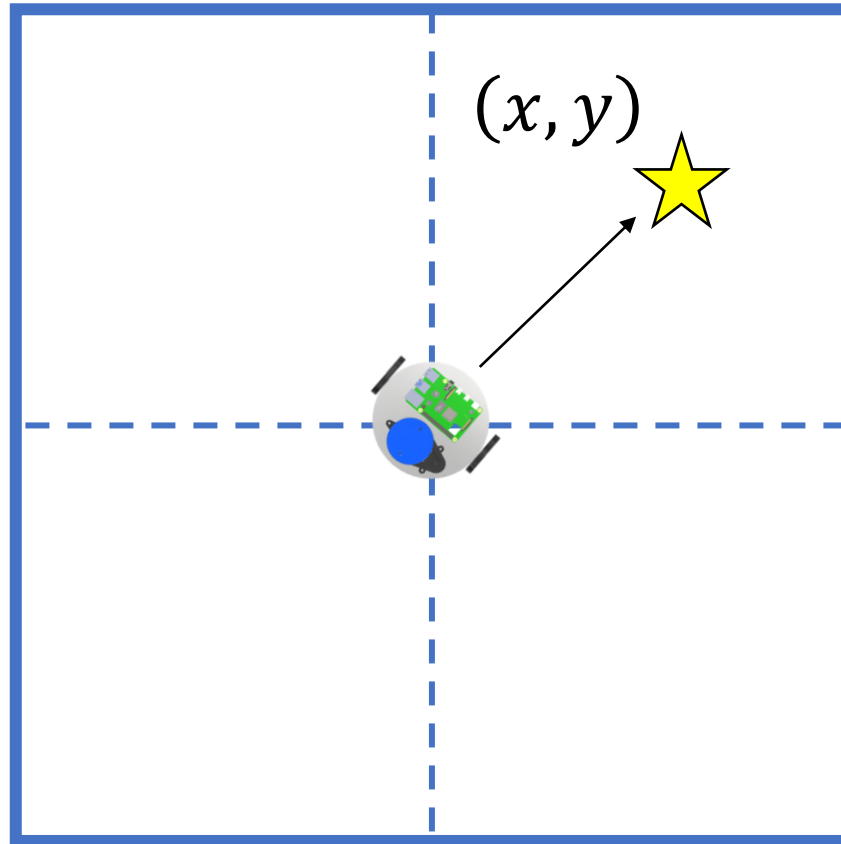
計画された移動経路が緑線で、目標位置姿勢が赤矢印で示される。検出されている障害物の周りには色が付いて表示される

# AMCLによる状態推定

- AMCL(Adaptive Monte Carlo Localization)は、自律移動ロボットのための確率的な状態推定方法のこと
- パーティクルフィルタを使用した確率的状态推定によって地図におけるロボットの位置・方位（姿勢）を推定する
- /amcl\_pose/がAMCLを表すトピック（データ）
- メッセージの型はgeometry\_msgs/PoseWithCovarianceStamped
- 以下の変数を読み込むことで推定位置を取得可能
  - pose.pose.position.x
  - pose.pose.position.y
  - pose.pose.position.z

# 課題 3

地図内の特定の地点をゴールに設定するプログラムを記述せよ



# 課題 3 : AMCL推定位置と目標位置の比較

- /home/robot/catkin\_ws/src/lightrover\_ros/scripts/**navigator.py** を編集
- プログラム内の**callback\_amcl(self, msg)**関数を編集
  - 最新の推定位置情報を受け取る関数
- **最新の推定位置と目標位置（ゴール）**を比較する
- 最新の推定位置
  - msg.pose.pose.position.x : 取得時のx座標[m]
  - msg.pose.pose.position.y : 取得時のy座標[m]
- goal : 目標位置
  - self.goal[self.goal\_number][0] : 目標位置 x [m]
  - self.goal[self.goal\_number][1] : 目標位置 y [m]
- 適切にゴール判定をして, 判定結果をprintで出力させる

# 課題 3 : 目標位置の設定

- `setGoal(self, px, py, px, ow)`
  - 目標位置を設定する関数
- `/move_base_simple/goal`が目標位置を表すトピック（データ）
- メッセージの型は`geometry_msgs/PoseStamped`
- 以下の変数(`x, y, z, orientation`) で目標位置を設定可能,  
`goal_point.pose.orientation.w=1.0`で固定で良い
  - `goal_point.pose.position.x`
  - `goal_point.pose.position.y`
  - `goal_point.pose.position.z`
  - `goal_point.pose.orientation.w`



# プログラムの実行（課題 3 ～ 6）

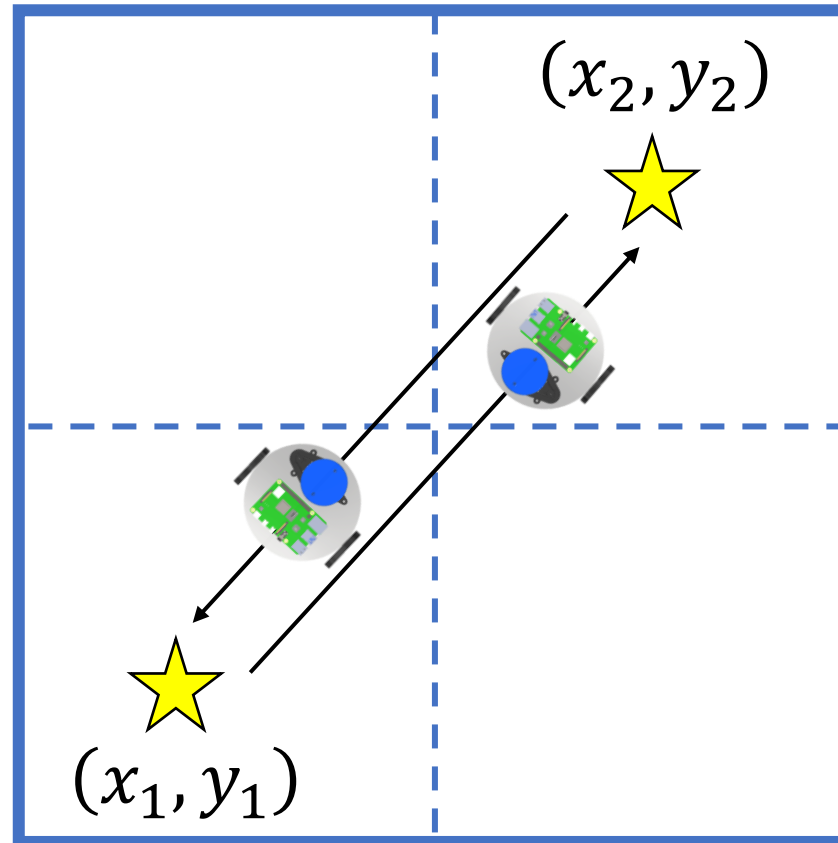
- /home/robot/catkin\_ws/src/lightrover\_ros/scripts/**navigator.py** を編集
- Navigatorクラス内のgoalに目標位置（x, y, z, orientation）が格納されている
- ターミナル1
  - roslaunch lightrover\_ros lightrover\_move\_base\_dwa.launch
  - まず初期位置を設定する
- ターミナル2
  - rosrund lightrover\_ros navigator.py
  - 指定位置に走行する

# ロボットの停止についての注意

- navigator.pyはロボットへの位置の指令を更新するプログラムなので、**navigator.pyを終了しただけではロボットは停止しない**
- 指令を終了するためには、課題2でも使用した  
`/home/robot/catkin_ws/src/lightrover_ros/launch/lightrover_move_base_dwa.launch`  
を終了することが必要なので注意

# 課題 4

地図内の特定の2つの地点（直線上）を往復するプログラムを記述せよ

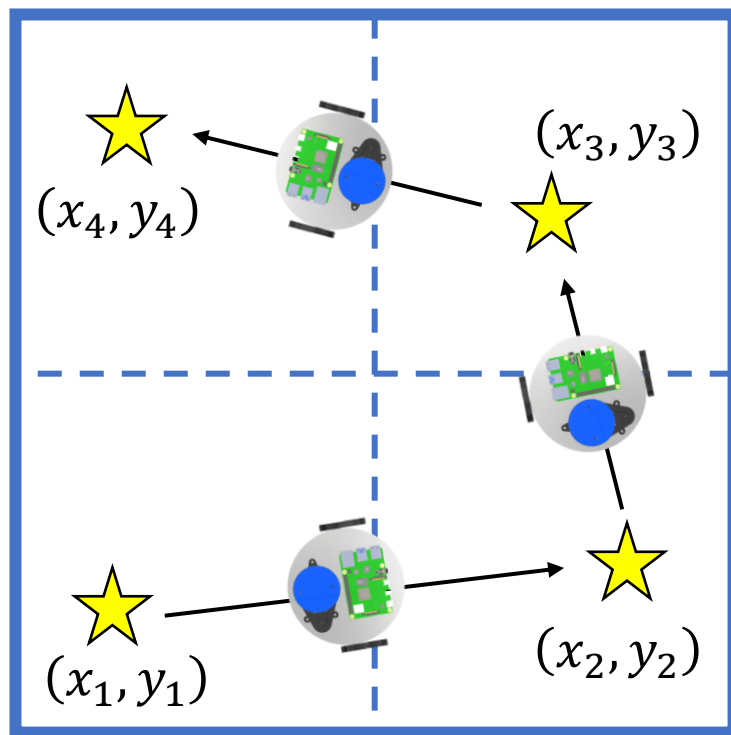


# 課題 4 ～ 6 : ゴール判定

- home/robot/catkin\_ws/src/lightrover\_ros/scripts/**navigator.py** 内を編集
- プログラム内の**callback\_amcl(self, msg)**関数を編集
- 目標位置の編集 (8行目)
- 目標位置に到達したらgoal\_number (9行目) を増加させる
- 適切にゴール判定をして, 判定結果をprintで出力させる

# 課題 5

作成した地図を用いて，スタートからいくつかの指定したポイント  
を通過してゴールまで自律ナビゲーションを行うプログラムを記述  
せよ．また指定したポイントの座標と位置推定結果を比較せよ．



# 課題 6

スタートからゴールまでの自律ナビゲーションの位置推定結果（**AMCL**と**オドメトリによる推定位置**）を比較せよ

- 地図作成をした原点からスタートする
- オドメトリ情報の取得は第1回の実習を参考に作成する

