機能システム学特論レポート

1．自作マニピュレータ

ここでは，csv ファイルを用いて自作したマニピュレータでの Rviz と MoveIt についての操作方法について述べる．

1.1.　Rviz

- 使用するファイル：totsuka_arm.launch
- ディレクトリ：catkin_ws/src/totsuka_arm/launch/

下記コマンドにより Rviz で確認する．

```
$ roslaunch totsuka_arm totsuka_arm.launch
$ rosrun rviz rviz
```

上記コマンドを実行すると以下の Fig. 1 と Fig. 2 が表示される．自作マニピュレータの関節は，Fig. 1 のスライドバーで調整することが可能である．Fig. 2 は最初からマニピュレータが表示されるわけではないため，画像内に示した手順に従って表示させること．実際に Fig. 1 のスライドバーを調整した場合の例を Fig. 3 に示す．自作マニピュレータの姿勢が変化していることが確認できる．

Fig. 1　Rviz[1]

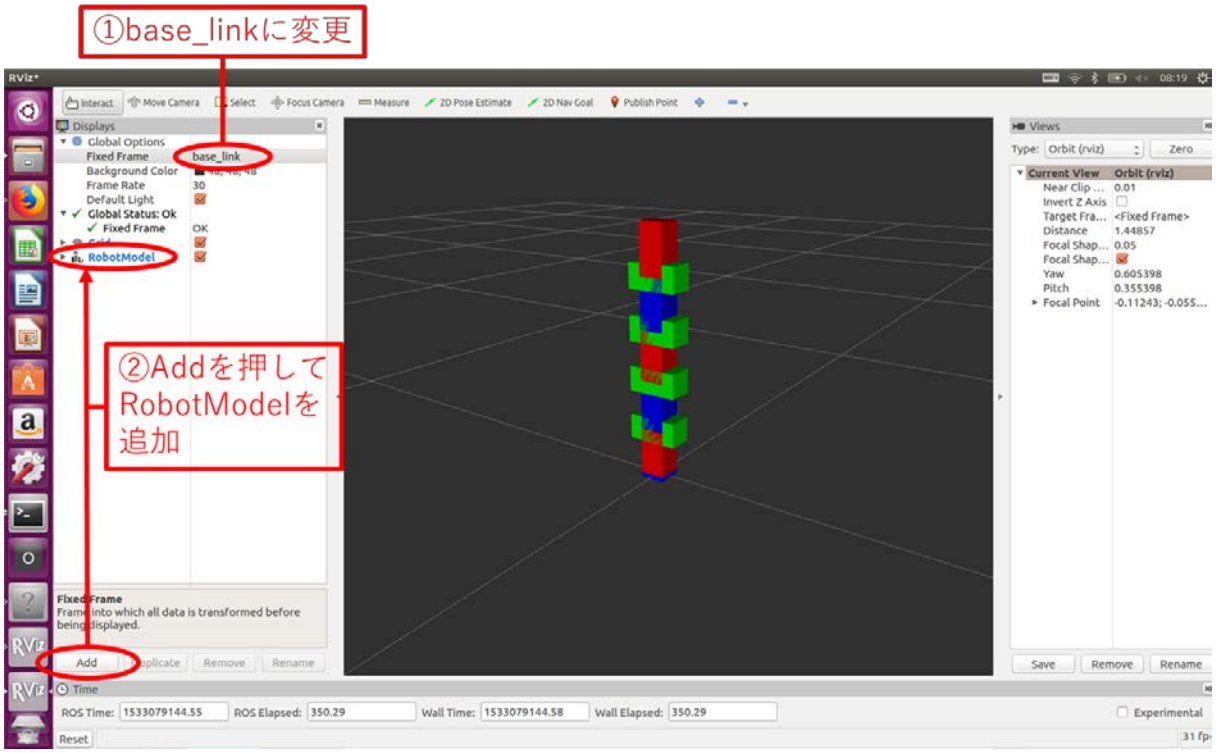

Fig. 2　Rviz[2]

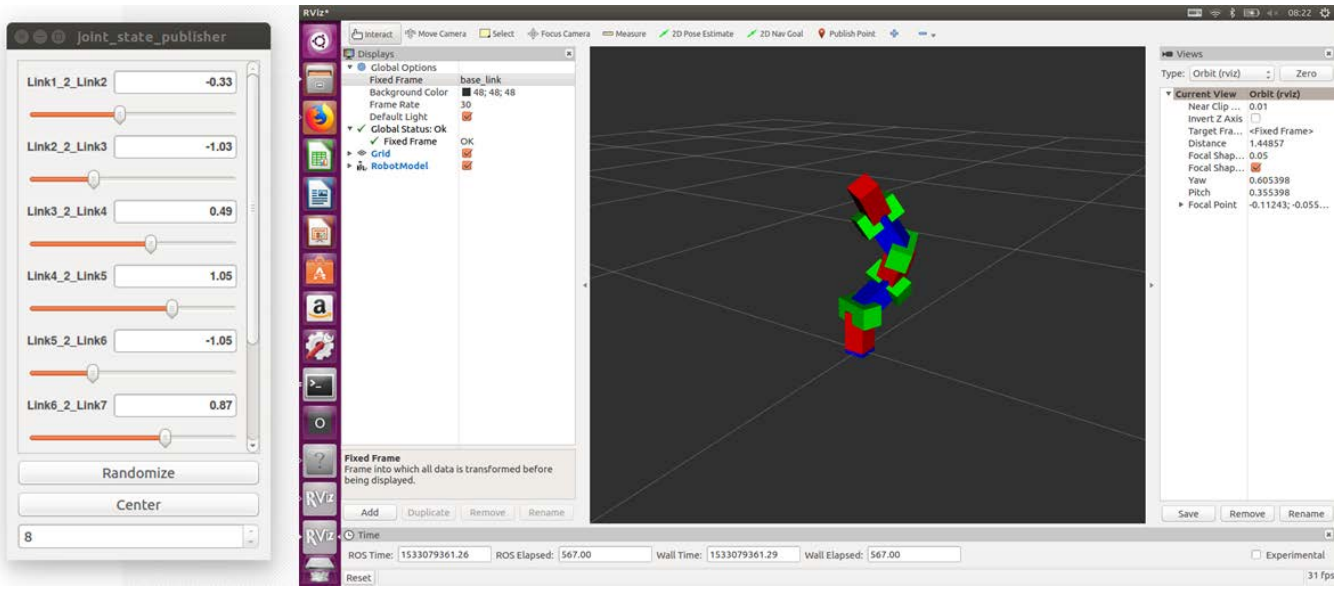

Fig. 3　Rviz[3]

1.2.　MoveIt

- 使用するファイル：demo.launch
- ディレクトリ：catkin_ws/src/totsuka_arm_moveit_config/

下記コマンドで，Rviz 上にマニピュレータを表示させる．

```
$ roslaunch totsuka_moveit_config demo.launch
```

コマンド入力後に Fig. 4 のような画面が現れる．

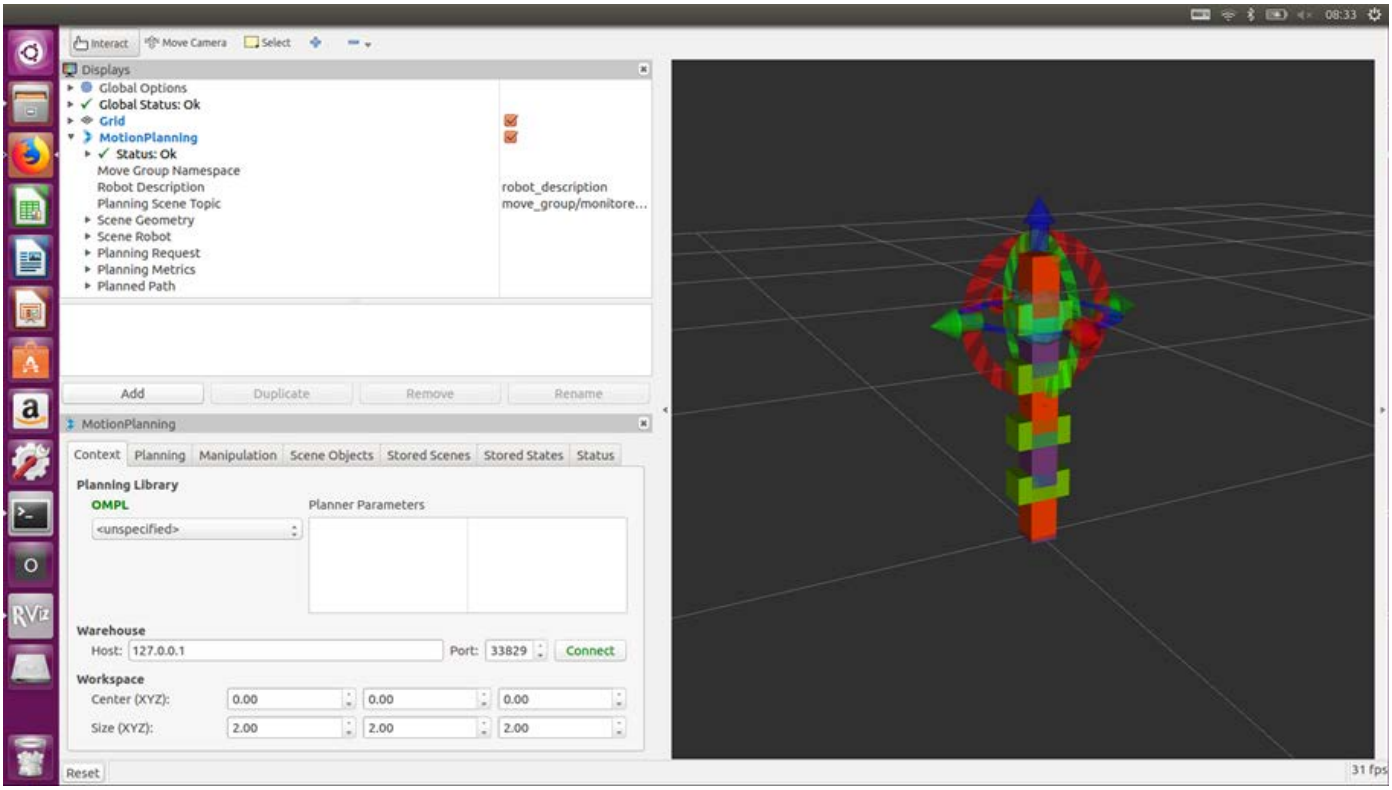

Fig. 4　MoveIt[1]

マニピュレータ先端のボールのようなものをドラッグすると，Fig. 5 のように姿勢を変更することが可能である．

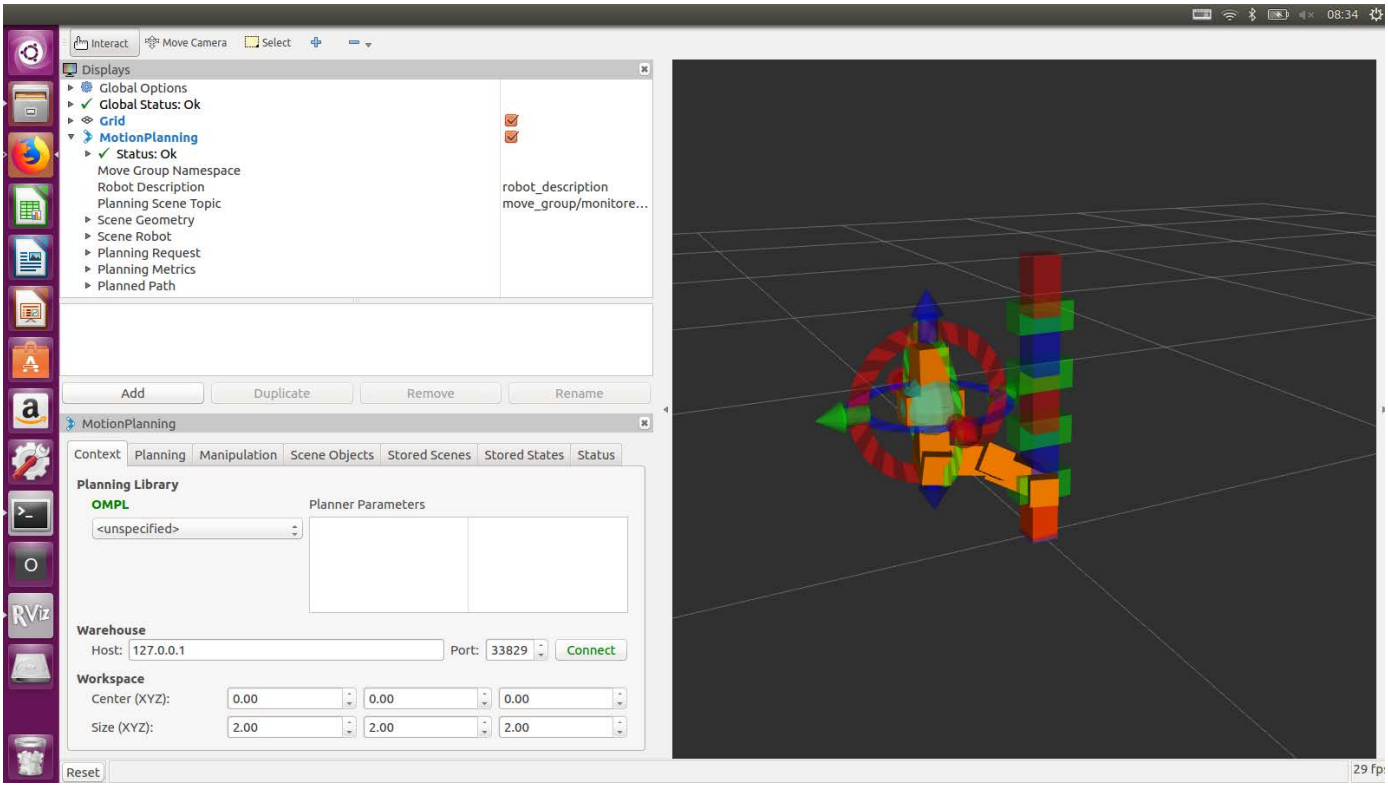

Fig. 5　MoveIt[2]

Fig. 6 の手順を実行すると，Fig. 5 で指定した目標姿勢への軌道を生成する．アニメーションでどのように関節角度が変化していくか視覚的に確認できる．

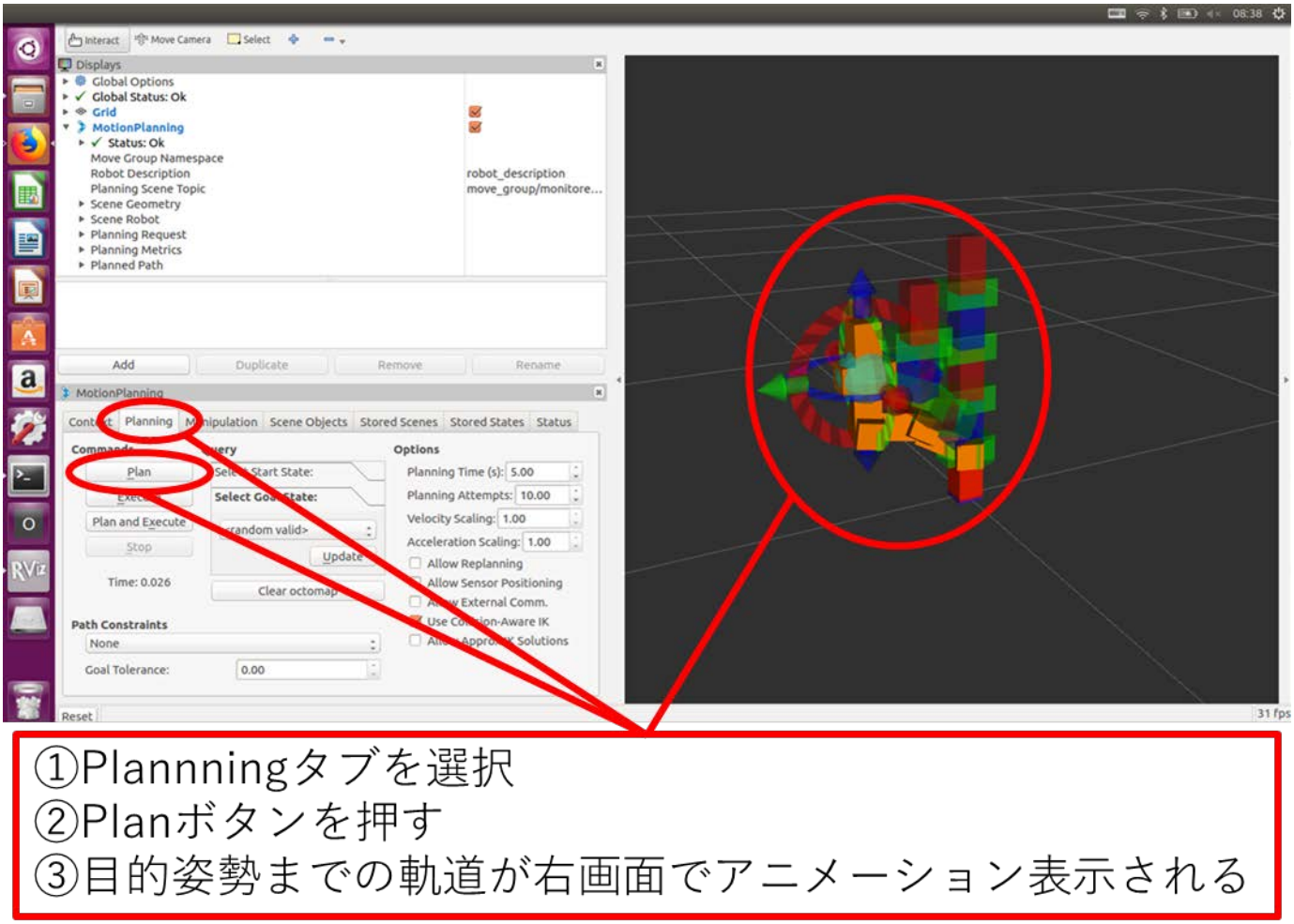

Fig. 6　MoveIt[3]

2．自作マニピュレータ付き差動２輪移動ロボットによるナビゲーション

ここでは，講義中に作成した差動２輪移動ロボットに自作マニピュレータを合体させ，合体させたロボットでのマニピュレータの関節角の制御と合体させたロボットを用いた AMCL によるロボットの自己位置推定について述べる．

2.1. マニピュレータの関節角の制御

- 使用するファイル：
  ①arm_control.launch
  ②diff_mobile_gazebo.launch
- ディレクトリ：
  ①catkin_ws/src/arm_control/launch/
  ②catkin_ws/src/diff_mobile_robot/launch/

下記のコマンドを実行する．

$ roslaunch arm_control arm_control.launch　※端末１つ目で立ち上げる：自作マニピュレータコントロール用

$ roslaunch diff_mobile_robot diff_mobile_gazebo.launch　端末２つ目で実行：シミュレータ起動用

※シミュレータを起動する前に立ち上げておく．

上記のコマンドを実行すると，Fig. 7 のように自作マニピュレータが付いた差動 2 輪移動ロボットが表示される．

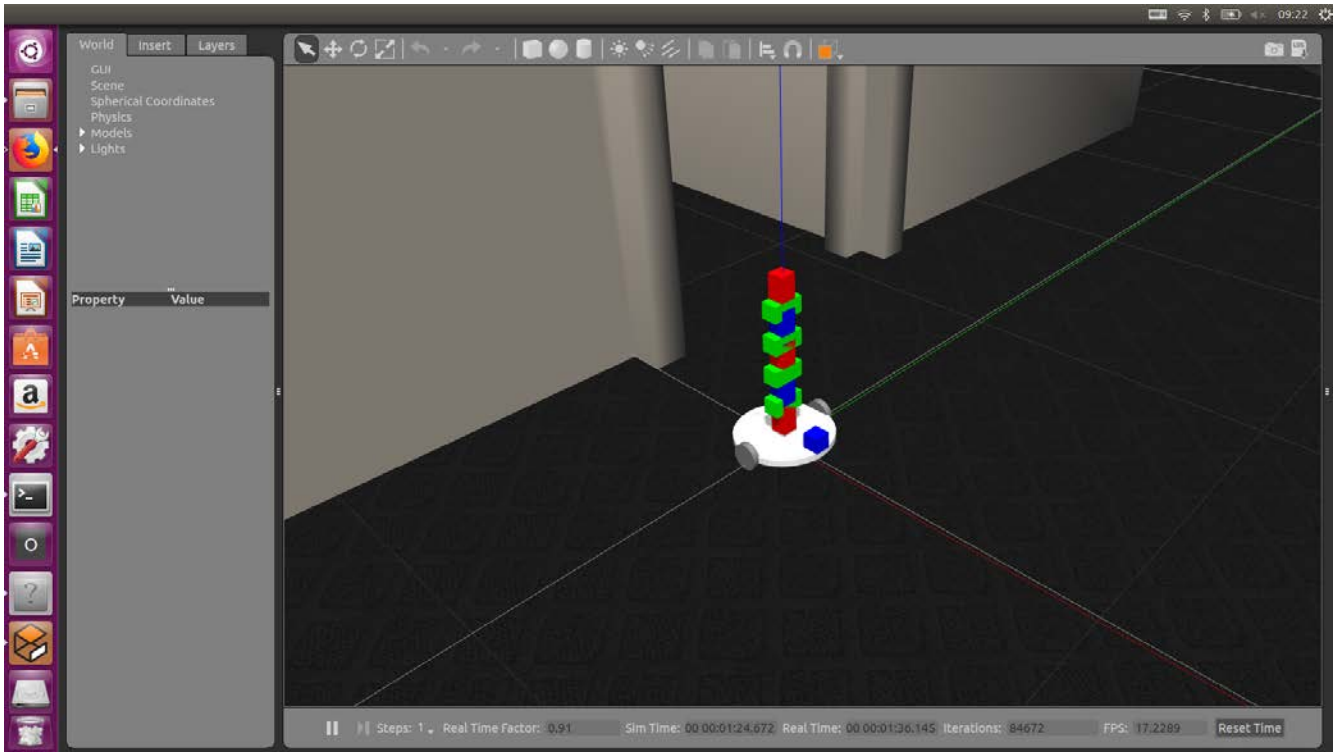

Fig. 7　AMCL[1]

ただし，上記のコマンドにおいて arm_control.launch を先に実行していない場合，Fig. 8 のように自作マニピュレータが倒れてしまうため注意する．

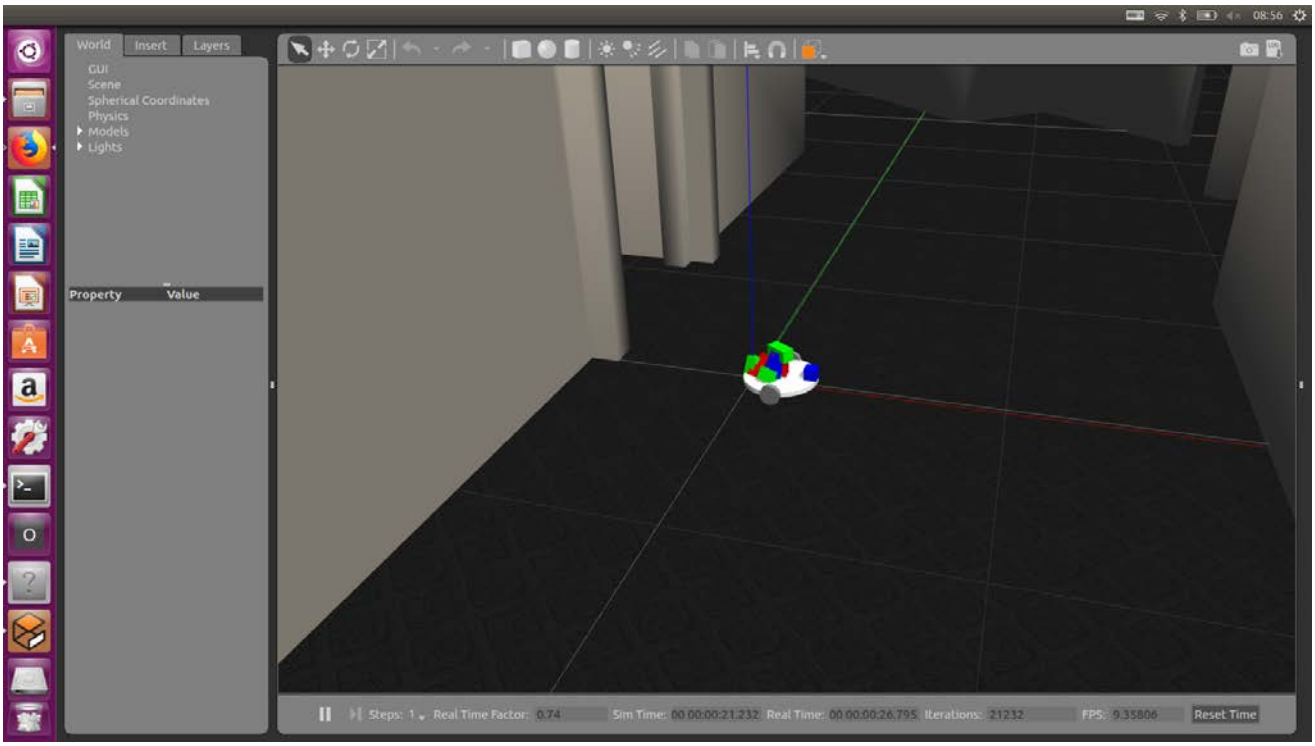

Fig. 8　AMCL[2]

Fig. 7 のようにロボットが表示されたら，自作マニピュレータにコマンドラインで目的角度を指定してみる．

下記のコマンドを実行する．

```
$ rostopic pub -1 /arm/Link2_2_Link3_position_controller/command std_msgs/Float64 “data: 1.0”
$ rostopic pub -1 /arm/Link4_2_Link5_position_controller/command std_msgs/Float64 “data: -1.0”
$ rostopic pub -1 /arm/Link6_2_Link7_position_controller/command std_msgs/Float64 “data: -1.0”
$ rostopic pub -1 /arm/Link8_2_Link9_position_controller/command std_msgs/Float64 “data: 1.0”
```

上記のコマンドを実行すると，Fig. 9 のようにマニピュレータの姿勢が変化する．

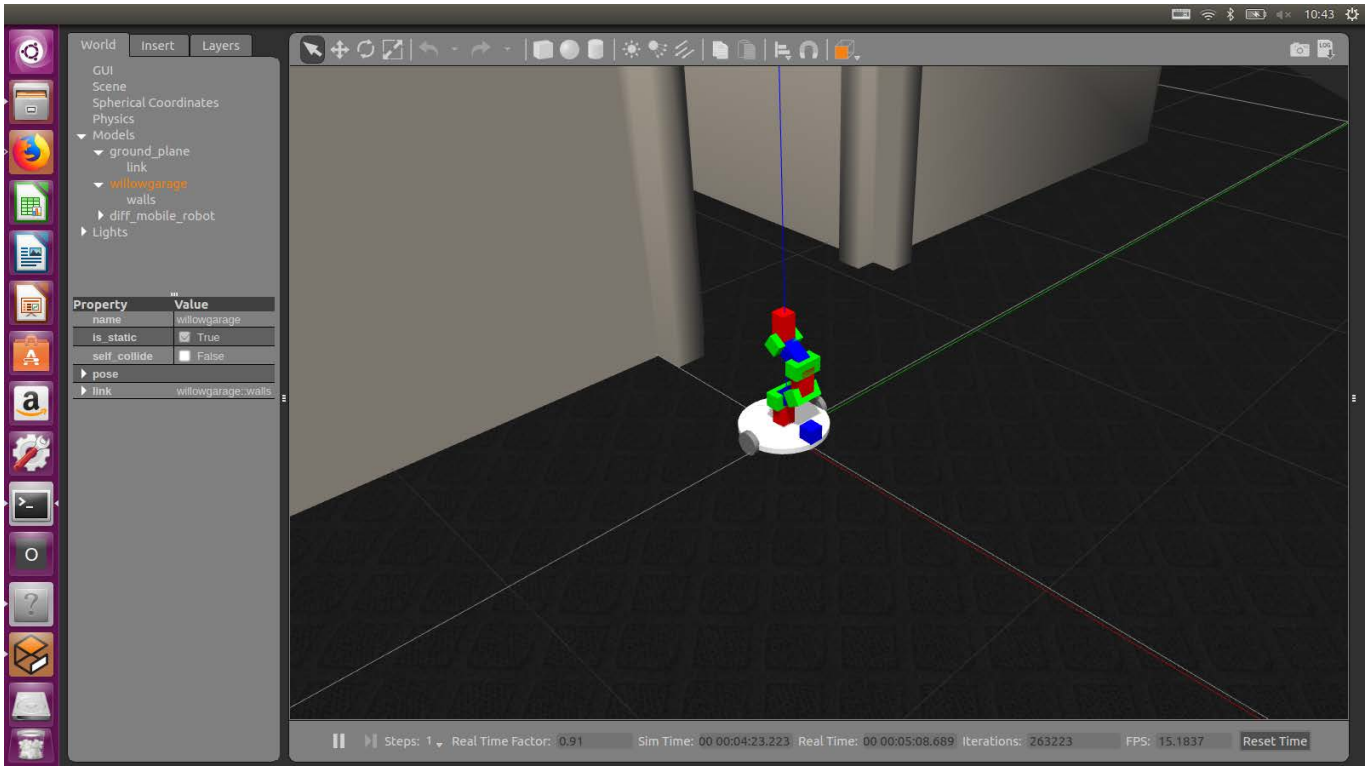

Fig. 9　AMCL[3]

2.2. AMCL によるロボットの自己位置推定

- 使用するファイル：
  ①arm_control.launch
  ②diff_mobile_gazebo.launch
  ③amcl.launch
- ディレクトリ：
  ①catkin_ws/src/arm_control/launch/
  ②catkin_ws/src/diff_mobile_robot/launch/
  ③catkin_ws/src/diff_mobile_robot/launch/

下記のコマンドを実行する．

```
$ roslaunch arm_control arm_control.launch　※端末 1 つ目で立ち上げる：自作マニピュレータコントロール用
$ roslaunch diff_mobile_robot diff_mobile_gazebo.launch　端末 2 つ目で実行：シミュレータ起動用
```

※シミュレータを起動する前に立ち上げておく．

上記のコマンドを実行すると，Fig. 10 のように自作マニピュレータが付いた差動 2 輪移動ロボットが表示され

る.

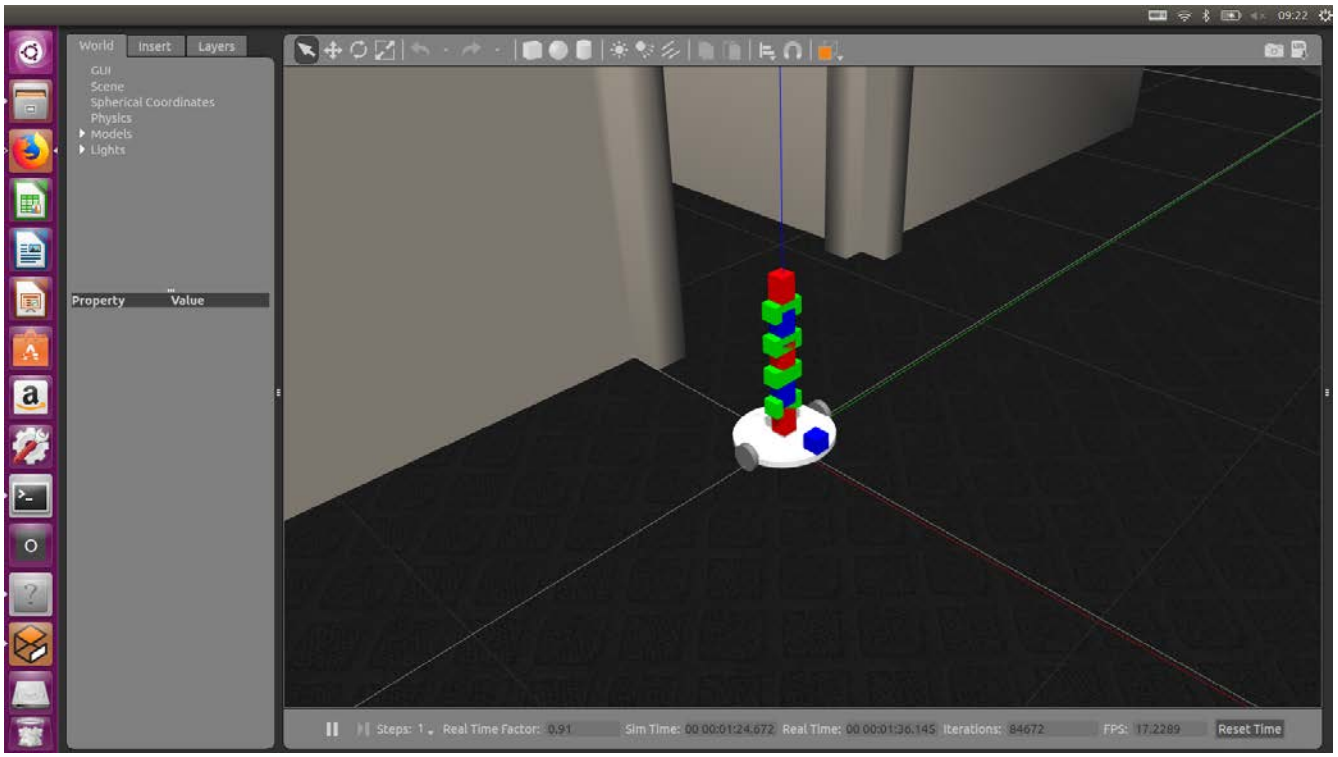

Fig. 10　AMCL[4]

下記のコマンドを実行する.

$ roslaunch diff_mobile_robot amcl.launch　端末 3 つ目で実行：ナビゲーション用

$ rviz rviz　端末 4 つ目で実行：移動場所指示用

上記のコマンドを実行したら, Fig. 11, Fig. 12 に従って Fig. 13 のようにマニピュレータとマップを表示させる.

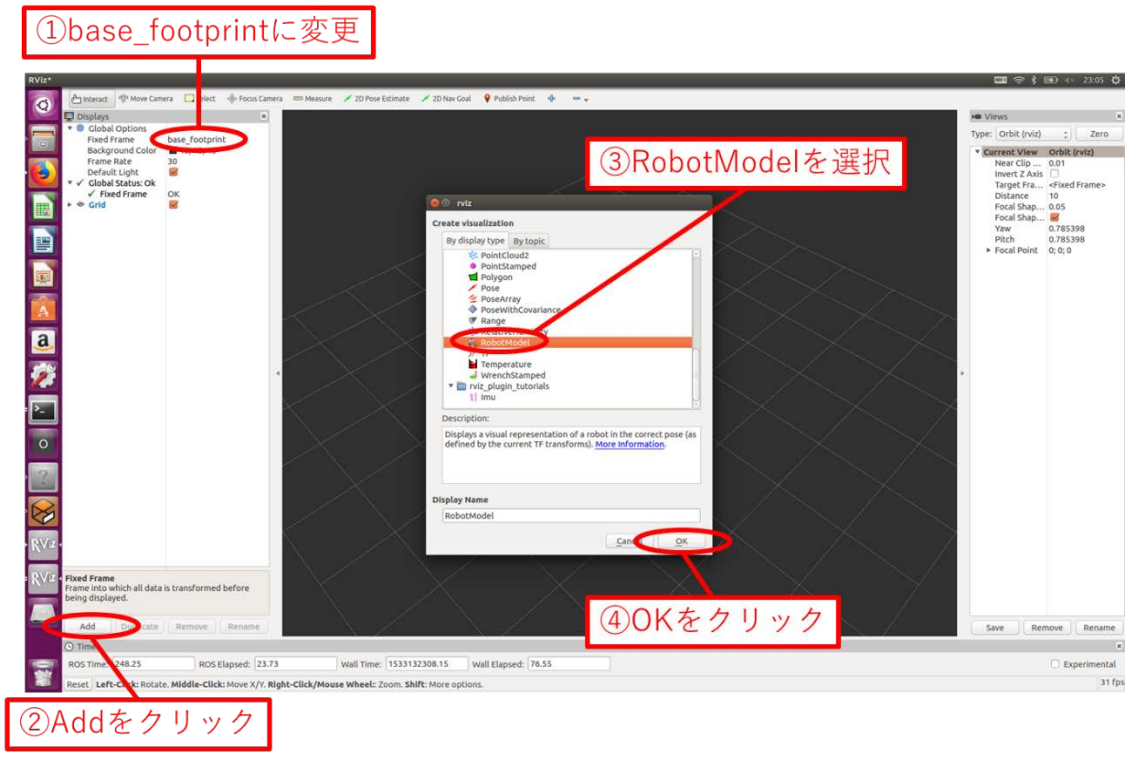

Fig. 11　AMCL[5]

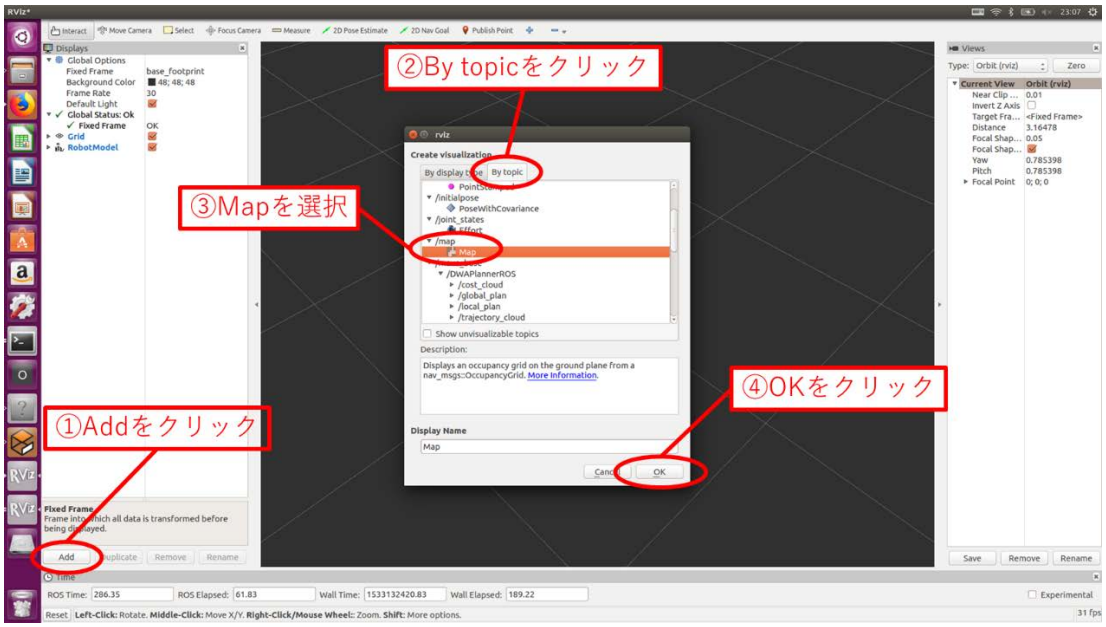

Fig. 12　AMCL[6]

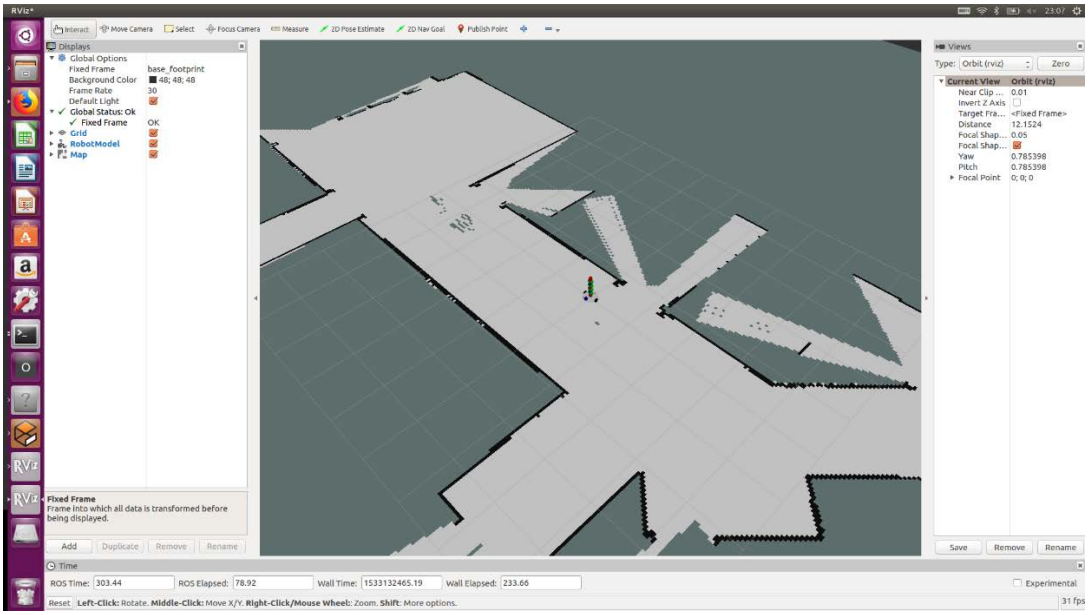

Fig. 13　AMCL[7]

Fig. 14 のように目的地を Rviz 上で指示すると，Fig.15 のように Gazebo 上のロボットが Rviz で指示した目的地に向かって移動していることがわかる．

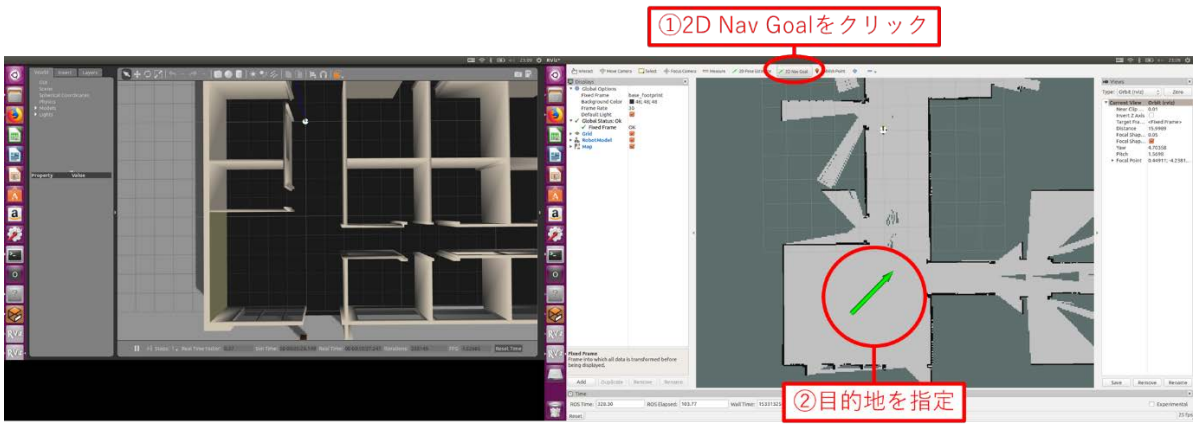

Fig. 14　AMCL[8]

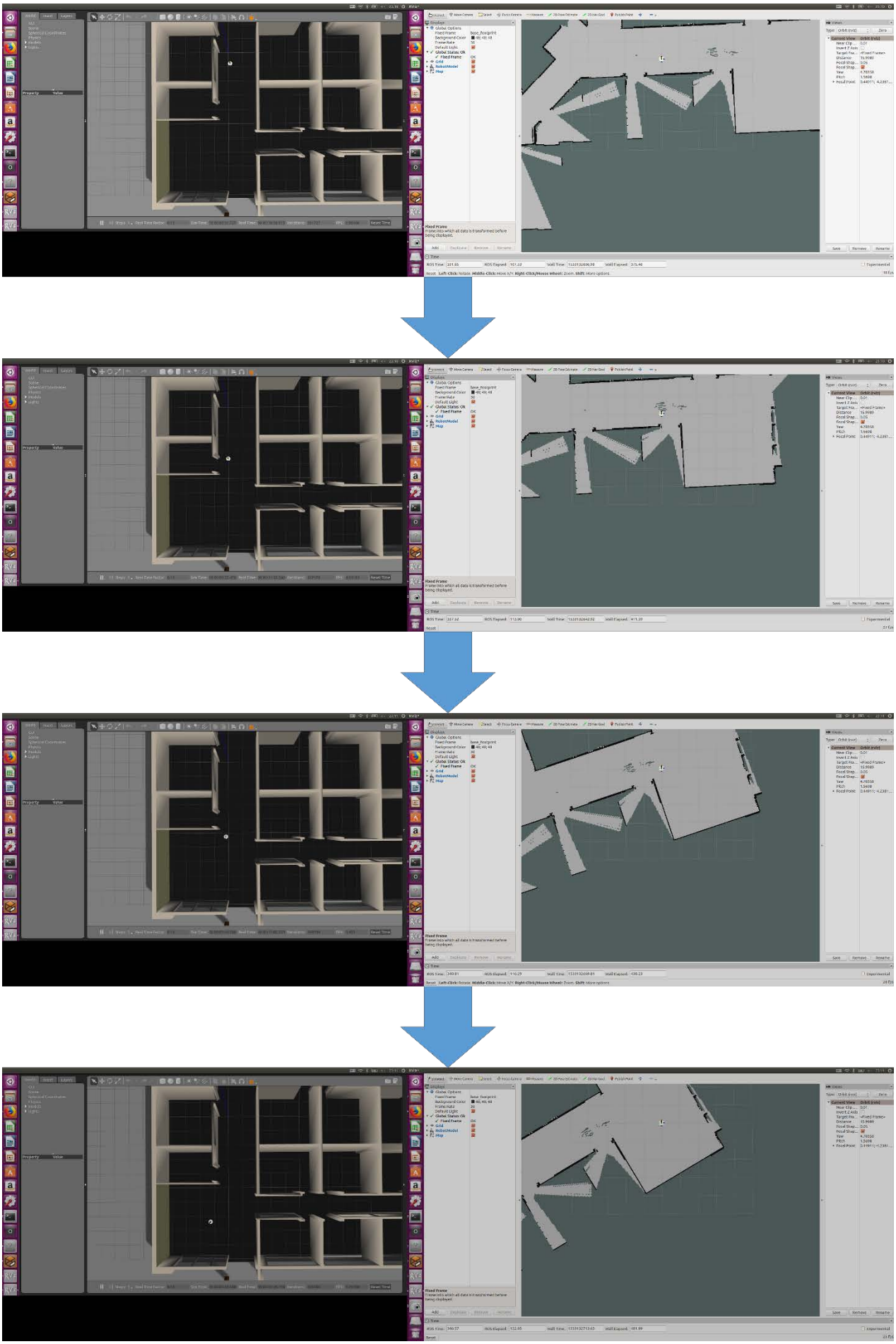

Fig. 15 AMCL[9]