Lab on localization with ROS

Olivier Aycard

Professor

Grenoble INP - PHELMA

GIPSA Lab

https://www.gipsa-lab.grenoble-inp.fr/user/olivier.aycard

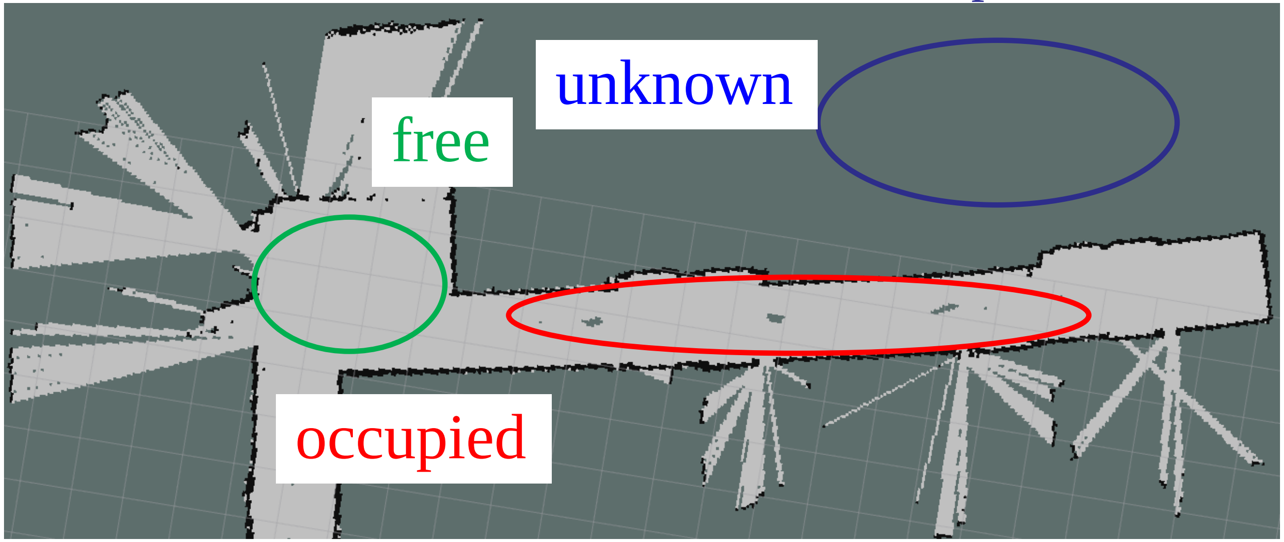
olivier.aycard@grenoble-inp.fr

Outline

1. Requirements for localization
2. Implementation of localization
3. Tests of both nodes

Requirements for localization: a map (1/3)

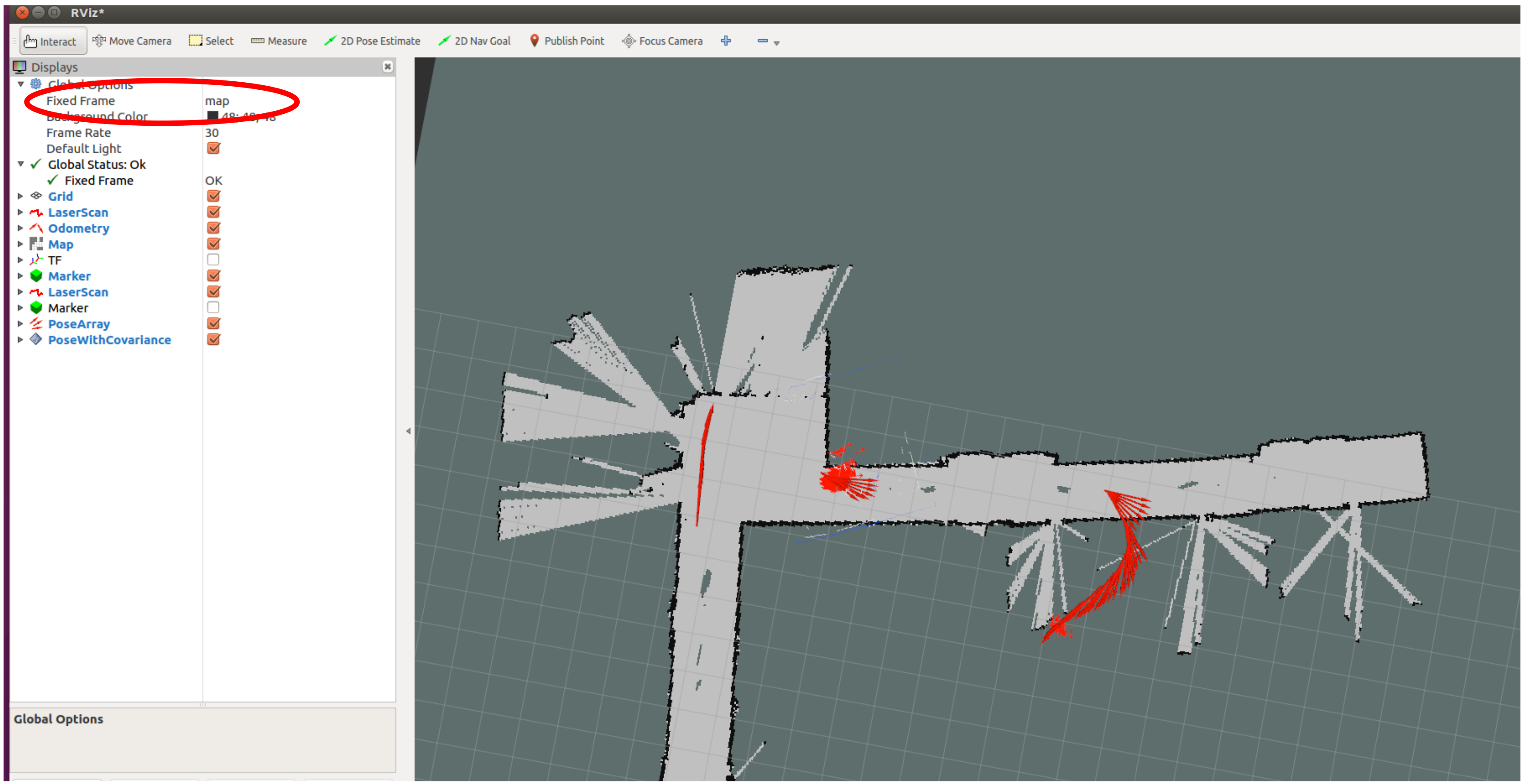
- A map in ROS is an occupancy grid (OG);
- A map has 2 components:
  - A yaml file containing a description of the map;
  - A png file containing the map: it can be edited/modified with an image editor
- To load a map in ros:
  - Rosrun map_server map_server “name_of_the_map”.yaml
  - You must be in the folder where the map is located



Requirements for localization: a map (2/3)

- To display a map and the localization process in rviz, you should modify/add 2 things in rviz

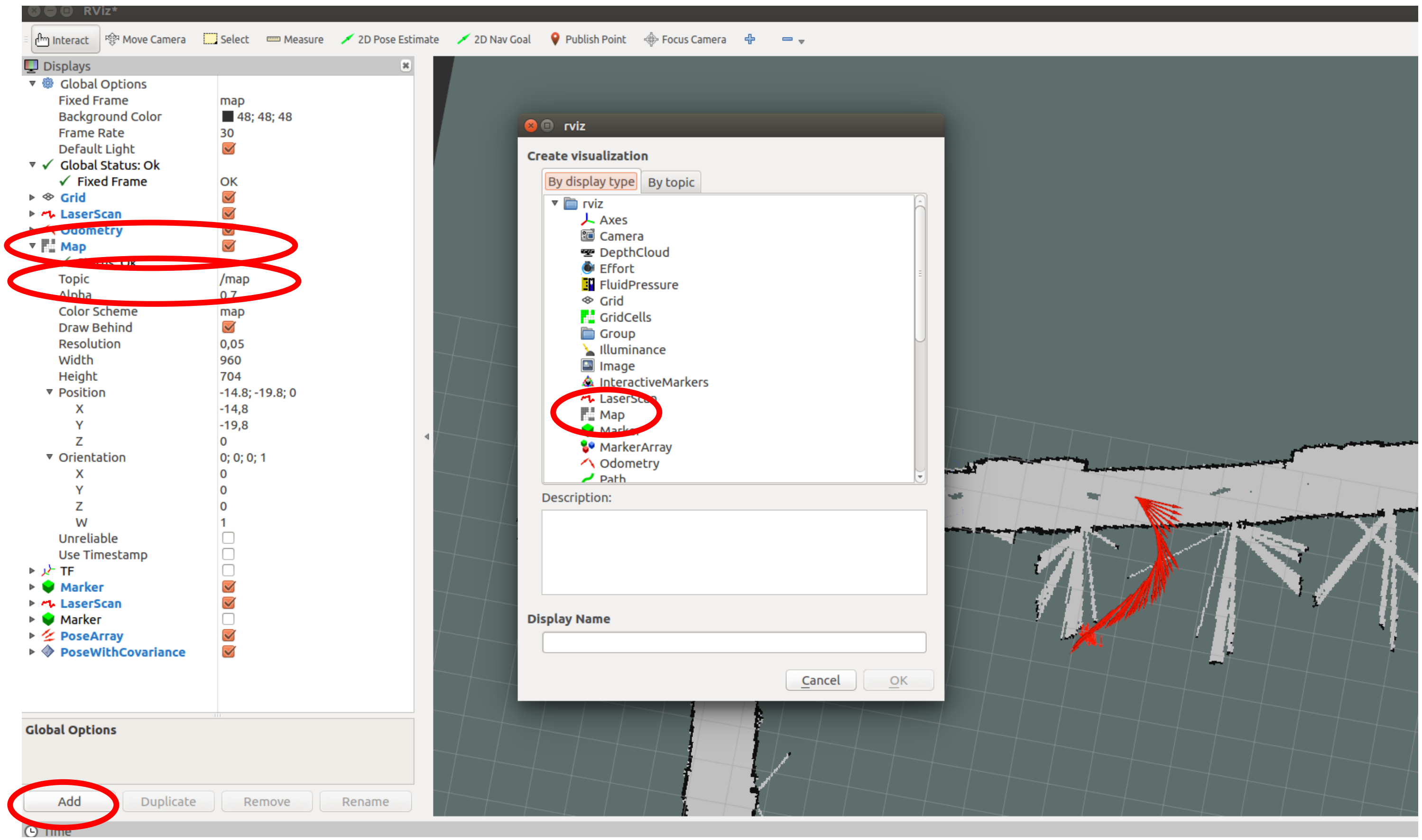
1. Change the global frame to map instead of laser



Requirements for localization: a map (3/3)

- To display a map and the localization process in rviz, you should modify/add 2 things in rviz

2. Add the map to rviz

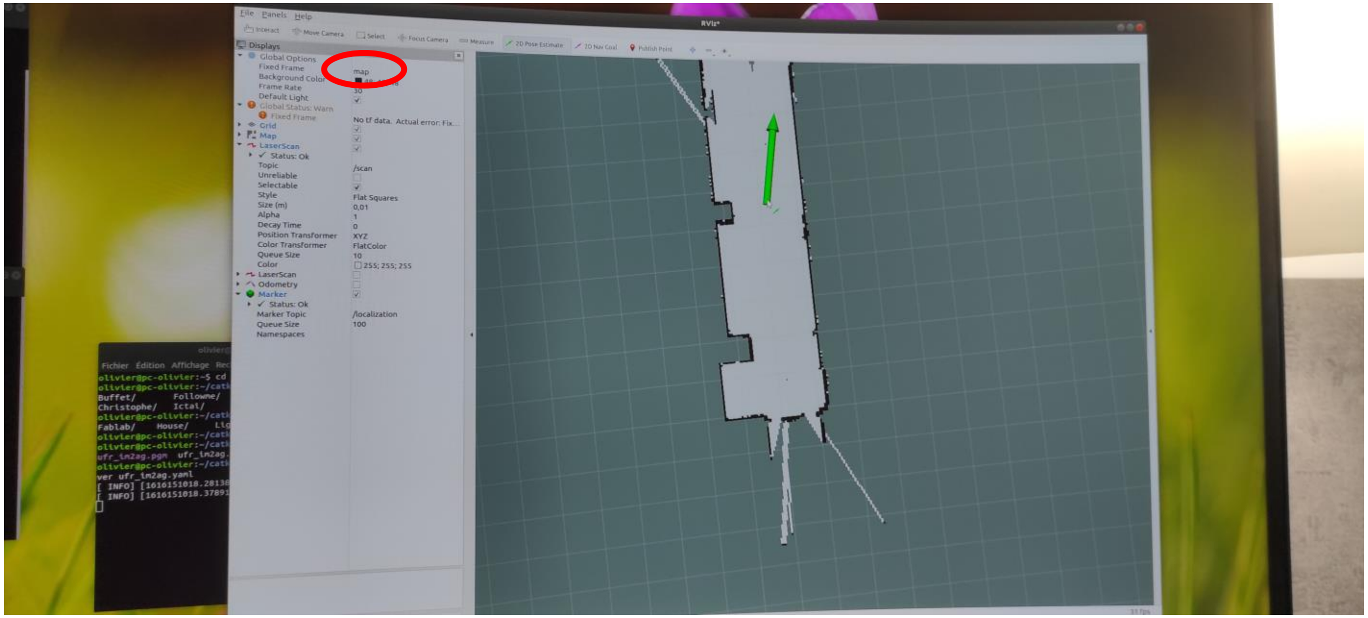


Outline

1. Requirements for localization
2. Implementation of localization
   1. Sensor_model_node
   2. Localization_node
3. Tests of both nodes

Implementation of sensor_model_node (1/4)

- We will implement a method to compute the matching between the current laser data and the map
  1. We provide a rough initial position;
     In rviz, use the "2D pose estimate" and choose a position in the map;
     The green arrow materializes the orientation of the robot.



Implementation of sensor_model_node (2/4)

- We will implement a method to compute the matching between the current laser data and the map
  2. If we suppose that the mobile robot is located at this initial position
     We compute for each hit of the laser its position in the map;
     If it matches to an occupied cell;
     Then we draw it in green color and increase the score_current;
     Else we draw it in red color

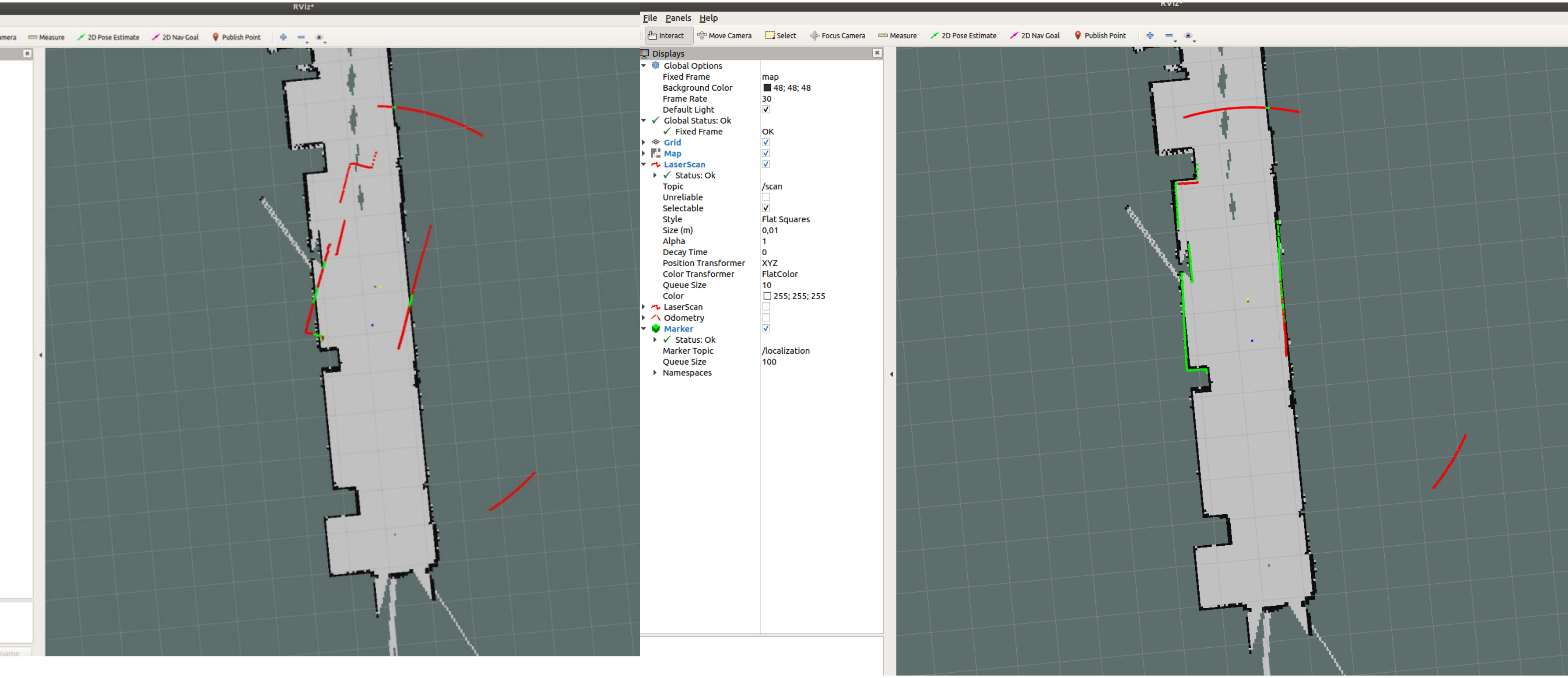
- See the lecture on localization for details of implementation

Olivier.Aycard@imag.fr

Implementation of sensor_model_node (3/4)

1. Edit *sensor_model_node.cpp* in *~/catkin_ws/src/localization*;
   - You should have a look on the source file;
2. Edit and modify *localization.cpp* in *~/catkin_ws/src/localization*;
   1. You should implement the function « sensor_model »
3. Edit localization.h to see the data structure and prototypes of functions
4. Check the results in a terminal and rviz

Implementation of sensor_model_node (4/4)



Outline

1. Requirements for localization
2. Implementation of localization
    1. Sensor_model_node
    2. Localization_node
3. Tests of both nodes

Implementation of localization_node (1/3)

- We will implement a local localization method
  1. We provide a rough initial position;
  2. We find the best position in the neighboring of this initial position;
  3. Each time, the mobile robot has a travelled a given distance or rotated of a given orientation, we predict its position with odometry;
  4. We find the best position in the neighboring of the predicted position.

- See the lecture on localization for details of implementation

Implementation of localization_node (2/3)

- We will implement a local localization method
  1. We provide a rough initial position;
  2. We find the best position in the neighboring of this initial position;
     Implement the method "initialize_localization" to compute the score of each position in the neighboring of the initial position and store the highest one;
     The neighboring is defined by all the positions at less than 1 meter and all possible orientations;
     You can test it before starting the next step.

Implementation of localization_node (3/3)

- We will implement a local localization method

3. Each time, the mobile robot has a travelled a given distance or rotated of a given orientation, we predict its position with odometry;
4. We find the best position in the neighboring of the predicted position.

In the method “estimate_position” compute the score of each position in the neighboring of the initial position and store the highest one;
The neighboring is defined by all the positions at less than 0,5 meter and 30 degres of the initial position.

- See the lecture on localization for details of implementation

Implementation of localization_node (3/4)

1. Edit *localization_node.cpp* in *~/catkin_ws/src/localization*;
   - You should have a look on the source file;
2. Edit and modify *localization.cpp* in *~/catkin_ws/src/localization*;
   - You should implement the 3 functions « initialize_localization », « estimate_position » and « find_best_position »
3. Edit localization.h to see the data structure and prototypes of functions
4. Check the results in a terminal and rviz

Outline

1. Requirements for localization
2. Implementation of localization
   1. Compute_score_node
   2. Localization_node
3. Tests of both nodes

Tests of both nodes

- You will test your two nodes with the map of ufr im2ag
- 2 options to perform your tests:
  1. You can do your tests with the provided rosbag to do it without a mobile robot;
     Find the rough initial position of the mobile robot in the rosbag file;
     Run on your rosbag to move the mobile robot;
     Pause your rosbag when the mobile robot is relocalizing.
  2. You can do your tests with a mobile robot;
     Position the mobile robot in the map of ufr im2ag;
     Move it slowly with the teleoperation_node and stop when it is performing relocalization.