Robotics Project: Scitos robot

Evgenia Vesterblom, Mikk Loomets

April 12, 2023

Abstract

Course project aims to introduce various essential topics of mobile robotics, such as feedback control for motion, robotic vision, sensor fusion, path planning and simultaneous localization and mapping (SLAM). Project tasks involve solving robotic problems in simulation and real robotics environment.

1 Introduction

Project involves the exploration and mapping for a cleaning robot Scitos. Aim is to map an indoor environment and navigate within it. Tasks involve sensor processing, mapping, path planning and task planning. Implementation is to be tested in simulation and real robotic platform. Task breakdown:

- W3: simple control implementation;
- W4: coordinate transform implementation and mapping overview;
- W5: Bayesian filter with log odds for occupancy grid mapping;
- W6: mapping improvements;
- W7: testing mapping on the real robotic platform and localization overview;
- W8-9: implementation of localization in simulation environment;

2 Tasks

2.1 Mapping Approach

Task W4. The idea is to use 2D occupancy grip for map with laser range-finder (LIDAR) sensor. Occupancy grids are used to represent a world as a discrete grid. The world is discretized into cells with the value of either being occupied or not. The LIDAR sensor data is going to be used to get the measurement in the world. Sensor measurement will give us information about probable obstacle in the world. This way we can assume that obstacles either exist or not in the particular and neighboring cells. With certain probability, it can be assumed that in between the measurement and sensor, no obstacles are detected. See Figure 1 how the map is divided and the cell, where the measurement is taken – is set to occupied. In the Figure 1, the more darker cells represent the probability of cell being occupied, the lighter cell represents the probability the cell being free.

- representation: each cell has a value that models occupancy;
- world is static until new information;
- pose in the world in know at measurement time;
- cell is independent;

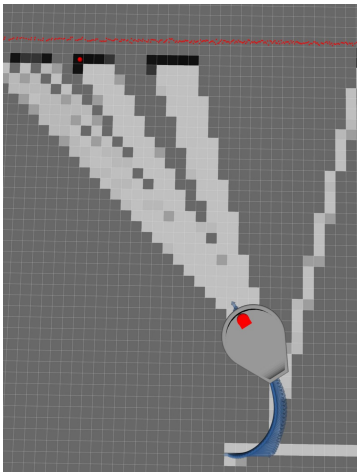

Figure 1: Cells in the space

2.2 Mapping Implementation

Task W5. To implement a mapping algorithm of choice and test it in a simulation framework using a virtual SCITOS robot including a laser-range finder as primary sensor.

Two implementations were made for occupancy grid occupancy mapping:

- using buffer method – running average of a cell value was updated continuously,
- using binary Bayesian filter with logarithmic odds form

The buffer method was simple implementation to handle the occupancy on a grid. Value of running average of a cell was taken, evaluated and updated. Next implementation used more sophisticated method to store occupancy grid data: the binary Bayesian filter with log odds.

Bayesian filtering is posterior probability distribution of a hypothesis based on a prior knowledge (previous value of a cell) and new observations. We can update the occupancy of the cell using log odds, because it is computationally efficient as it is needed to map value onto range of probabilities between 0 and 1 [Sta16]. Noise from sensor will not affect rapid changes in cell's state. The log odds notation computes the logarithm of the ratio of probabilities, where log odds ratio l is defined as

$$l(x) = \log\left(\frac{p}{1-p}\right)$$

where p represents the probability of an event occurring [Bur17]. To retrieve p from the log odds ratio following equation is used:

$$p = \frac{1}{1 + e^{-\log\left(\frac{p}{1-p}\right)}}$$

To implement cell value update in the map [223] [Sta16] the method results simply to addition of previous cell's value in log odds notion and new measurement:

$$\mathrm{l}(p) = \text{inverse sensor model} + \text{previous value} + \text{initial value} \tag{1}$$

where p is the probability of occupancy, and the other terms are of logarithmic scale defined as follows: inverse sensor model is the current sensor measurements transformed into expected probabilities of occupancy, see Figure 2; previous value is the previous occupancy grid estimates; and initial value is the initial value about the environment before any sensor measurements are taken (which in our case is $\log(0.5)$).

In other words, the equation 1 shows the probability of occupancy of the cell to the sensor measurements, previous occupancy estimates, and initial assumptions about the environment.

Function	Linear 100 point approach	DDA point approach
Point Processing	0.457s	0.009s

Table 1: Difference between static linear line and DDA points processing algorithms

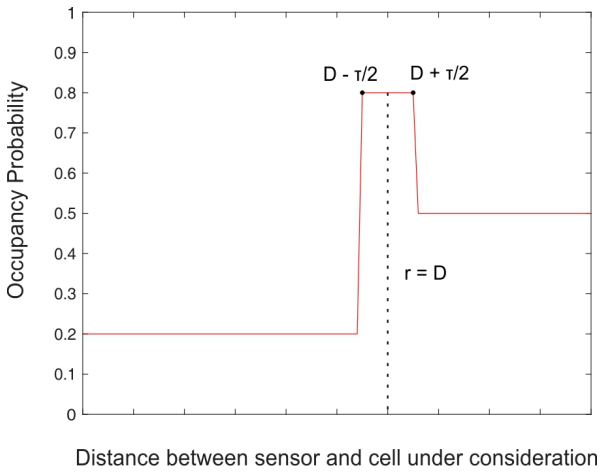

Figure 2: Inverse Sensor Model [223]

2.3 Mapping Improvements

Task W6. The improvements were made in several areas including:

- adjusting sensor correctly within the calculations;
- adding check of invalid sensor measurements;
- adding logic to update neighboring cells in respect with value τ marking them as occupied;
- improving line algorithm.

The changes that were added improved the quality of the generated map significantly. The resulting map itself became more accurate in comparison with previous one. The speed of generating the map increased due to optimizations in the line algorithm. Previous line algorithm used linear function with predefined set of 100 points on the line to update the free space between measurement and the obstacle. New implementation uses Digital Difference Analyzer (DDA) algorithm, which interpolates points based on the difference between the start and end points. Table 1 illustrates the difference in the computation in seconds per cycle.

Another algorithm for line points generation was proposed [ais23], but since the optimization using DDA was enough, the Bresenham algorithm can be implemented later in the mapping improvement scope.

Another change of setting neighboring cells occupied had positive effect on the map accuracy. The idea was to find a small value, with notation τ, which in the circle radius it is possible to find points related to the sensor's measurement. The value τ is experimental, it cannot be too small (otherwise the points in circle radius would point into the same cell). It cannot be too big, otherwise the result will be incorrect as the cells would be set initially as false-occupied. The factor τ can be taken into account from Figure 2, which should be adjusted according to the expected size of obstacles (e.g. thickness of walls) and the chosen resolution of the map. Demo can be seen here [Ves23]. Figure 3 shows the difference of mapping implementation improvements, the walls and space is much more accurate on the right side, where improvements have been added.

Figure 4 represents the map created using the mapping implementation of the room in Gazebo simulator setup visualized in RViz.

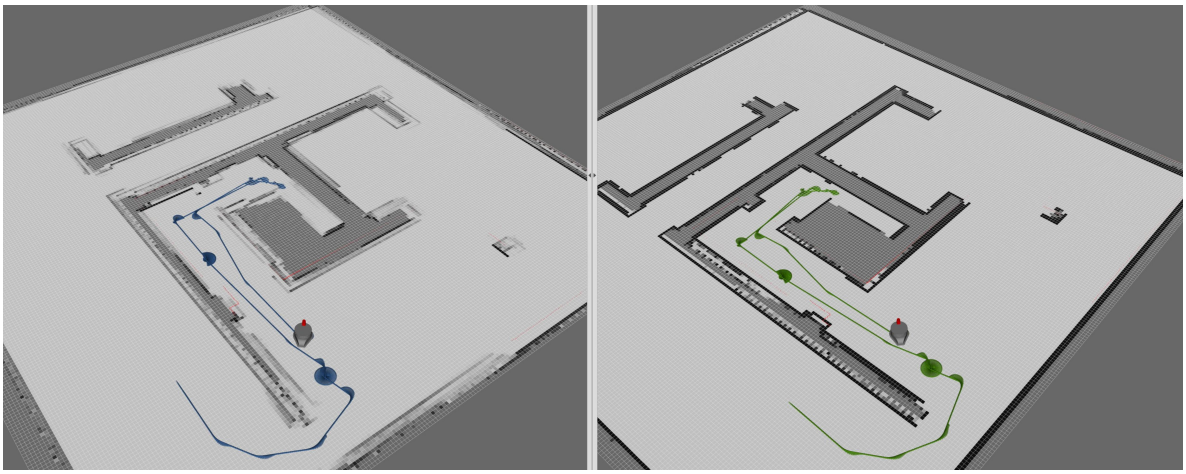

Figure 3: Mapping difference using improved mapping implementation [Ves23]

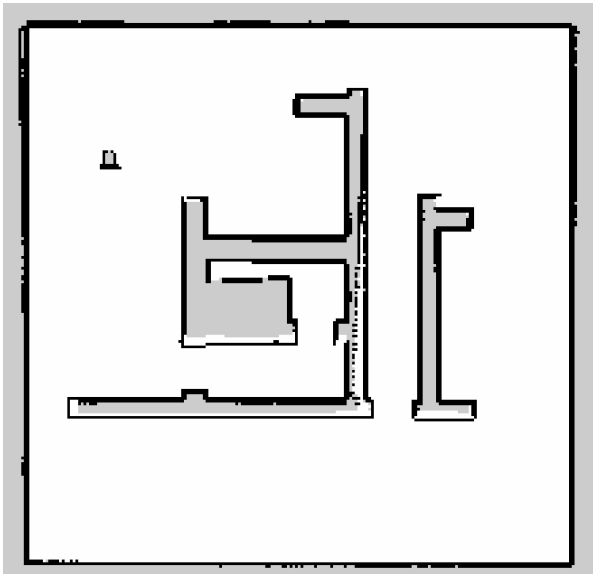

Figure 4: Map of the room

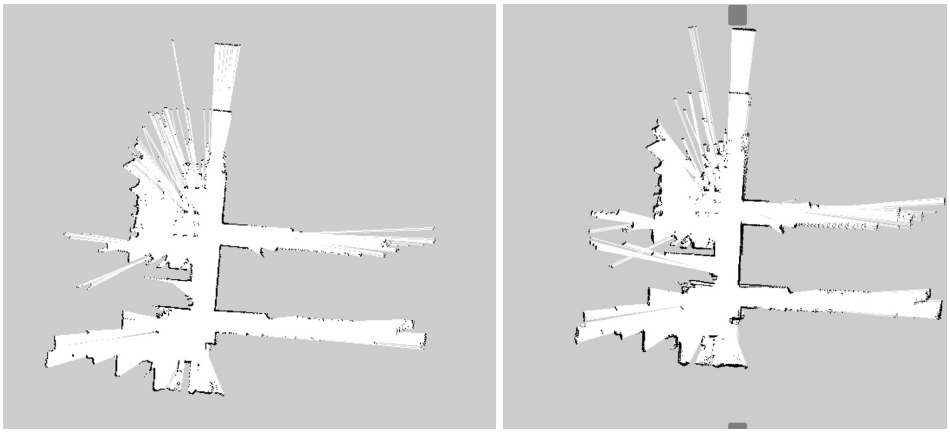

Figure 5: Map of the room using real robot (left GMapping, right Our Mapping)

2.4 Mapping Testing on Real Robot

Task W7. The main goal is to test mapping implementation and improve it based on real world data:

- line algorithm DDA implementation, significant change in cell calculation (100 points linear vs DDA);
- adjust function callback to not publish map too frequently;
- testing different resolutions for mapping;
- testing and improving neighbor cells occupancy settings;
- cancel measure when velocity too big;
- try improve on odom, on laser;

Main goal is to improve the mapping output. It is possible to do more mapping processing and reduce time to visualize the map output. Used mapping algorithm visualization with ROS gmapping tool – to improve own implementation. Real robot testing allowed to create rosbag data, so that when changes were done, the same data set could be used to improve the implementation.

Final mapping included timing checks so that it was possible to use most accurate odometry data. Few implementations were done, but the final version included using the buffer of odometry readings and matching corresponding laser reading – closest of last 10 odometry readings. By associating each laser reading with the closest odom reading, it is possible reduce the impact of odometry drift and improve the accuracy of the map.

The mapping visualization frequency was also reduced for RViz as it allowed more resources to be dedicated to mapping rather than visualization side. Figure 5 shows the final mapping output compared with Gmapping mapping algorithm.

2.5 Localization in Simulation Environment

In the previous week's homework, we investigated different localization approaches and decided to use the Monte Carlo Localization (MCL) algorithm for the implementation. MCL is a probabilistic algorithm that can estimate the pose (position and orientation) of a robot based on a known map, sensor data, and odometry. In our case – occupancy grid map; Lidar sensor data and odometry.

The basic idea of the MCL algorithm is to use a set of particles to represent possible poses of the robot: X, Y and heading angle. These particles are guesses of the robot's pose and are randomly distributed throughout the known map in the free area. As the robot moves, we update the set of

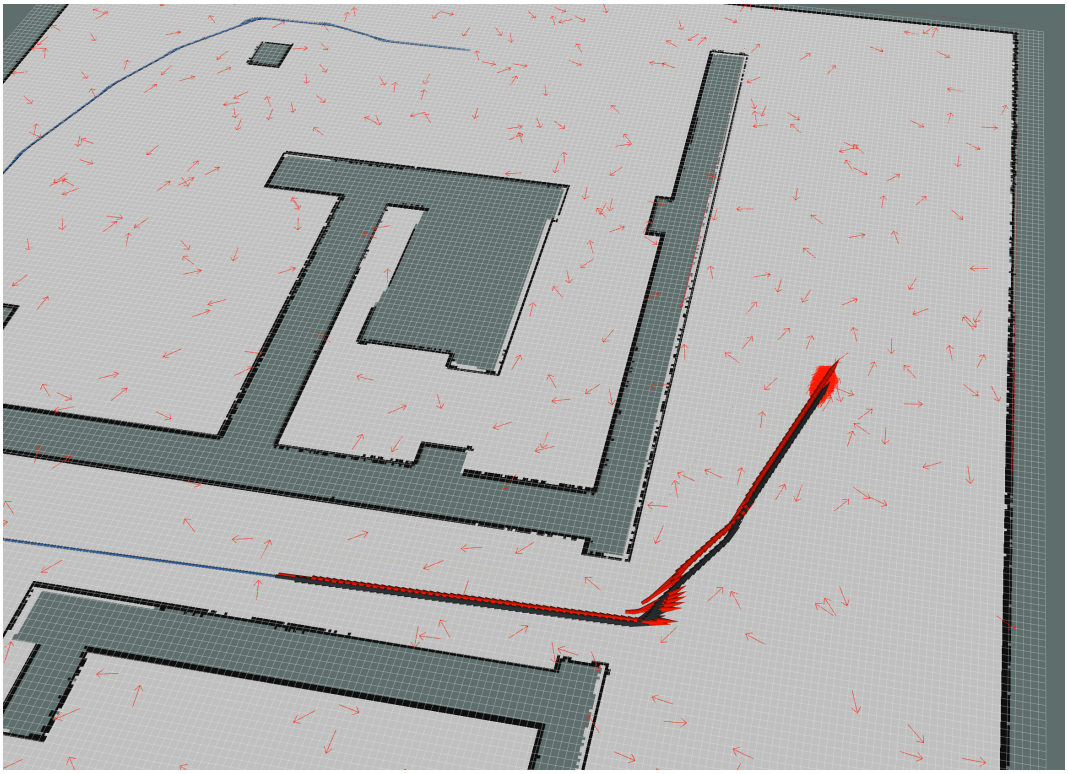

Figure 6: Localization implementation, odom, ground truth, best estimate and particles shown as red set

particles to maintain the guesses of the robot's pose.

The MCL algorithm consists of several steps, including prediction, measurement, re-sampling, and state estimation, we have modified our algorithm and simplified it

- Prediction step: we apply the motion model to each particle to estimate its new pose based on the odometry data;
- Measurement step: we use the laser range data to calculate the weight of each particle being in the correct pose;
- Re-sampling step: we regenerate higher weighted particles based on laser measurement data;
- Estimation step: we estimate the pose of the robot using the best weight of generated particles;

To use the MCL algorithm, we need laser scan data, odometry data, and the map context. The laser data is used for perception, the odometry data is used to track the robot's movement, and the map is used to know which areas are occupied and which are not. The output of the MCL algorithm is the best estimate of the robot's pose based on the input data.

2.5.1 Evaluate

For each particle that exists. For each lidar reading we received we calculate x and y coordiantes where the measurement landed. Then we check if that cell is occupied with probability of at least 80. If the cell is occupied we add 1 to that particles weight.

2.5.2 Re-sample

We generate new set of particles using the previous set. While the new set has less than 90% we choose random starting index and random weight (in range of 2x best particle weight). We move forward

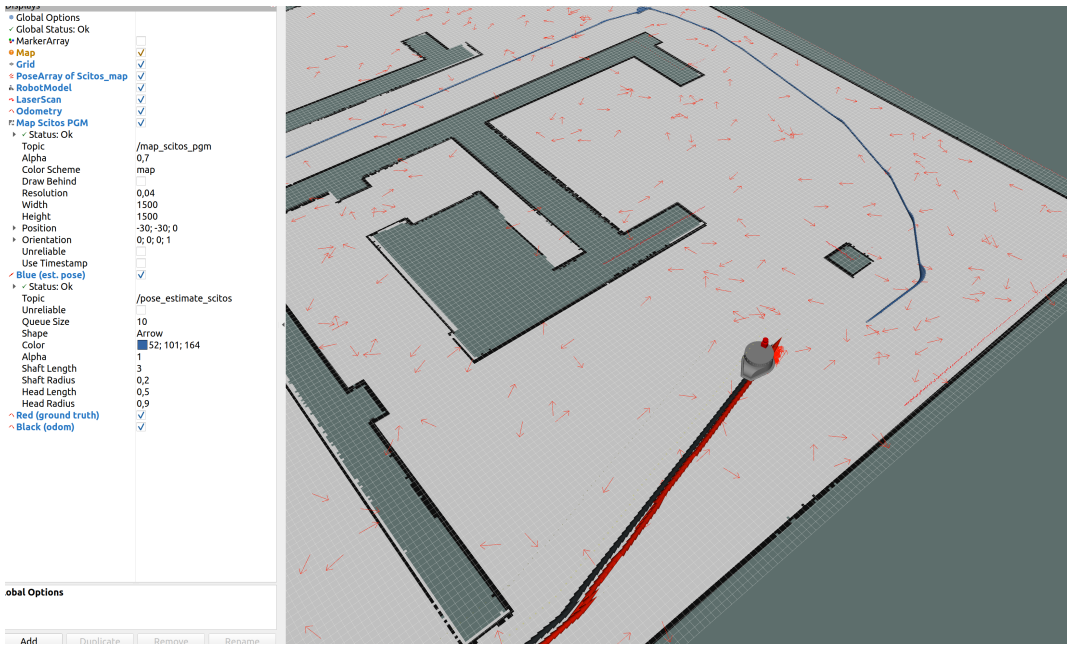

Figure 7: Our localization implementation

indexes while the current sum of weights is less than the randomly chosen weight. Once the weight sum grows larger than random weight we pick that particle, which means we generate a child to that particle. That child particle will be moved by calculated deltaX, deltaY and deltaYaw plus a random amount to each variable that will compensate for any errors in commands given. Finally that child particle will be added to the new set. After 90% of the particles have been generated the last 10% will be generated randomly all over the map.

Figures 6 and 7 show the outcome of the localization in simulated environment. During simulation there were not big visual differences in seen odometry vs ground-truth, but estimated pose differs from real location a bit, see 8. Figure 9 shows differences in the estimate x, y, heading vs ground-truth info x, y, heading. When running the simulation, depends on how much we drive, the errors we got, for example, is X: -0.154466, Y: 0.075006, Heading: 0.023458, info is shown in the command prompt

References

[223] IAS0060 Mapping 2. Project 1 (scitos): Bayesian statistics and occupancy probabilities, 2023.

[ais23] IAS0060 Mapping algorithm in simulation. Project 1 (scitos): Implement mapping algorithm in simulation, 2023.

[Bur17] Gian Diego Tipaldi Wolfram Burgard. Robot mapping - grid maps, 2017.

[Sta16] Cyrill Stachniss. Robot mapping - grid maps, 2016.

[Ves23] Evgenia Vesterblom. Project 1 (scitos): Mapping demo youtu.be/s0qfal7szxa, 2023.

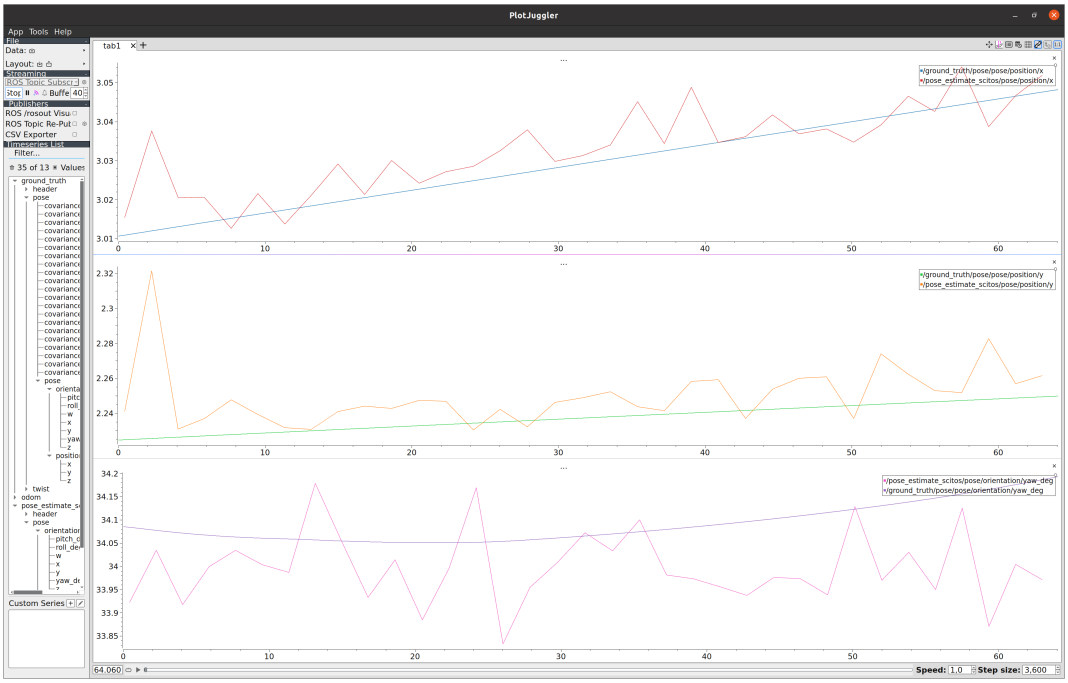

Figure 8: X, Y, Heading differences in estimation vs real position

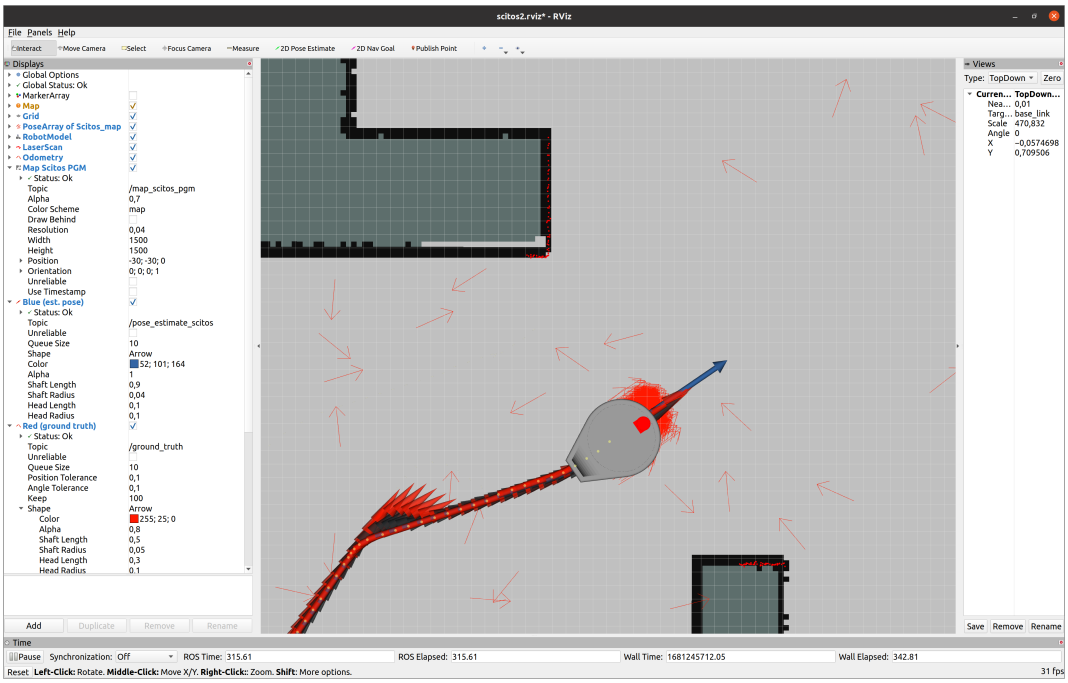

Figure 9: Top view, black mark - odom, red mark - ground-truth, blue mark - our estimated pose