Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
1–10
© The Author(s) 2017
DOI: 10.1177/0037549717710098
journals.sagepub.com/home/sim

An architecture for multi-robot localization and mapping in the Gazebo/Robot Operating System simulation environment

Erkan Uslu, Furkan Çakmak, Nihal Altuntaş, Salih Marangoz, Mehmet Fatih Amasyalı and Sırma Yavuz

Abstract
Robots are an important part of urban search and rescue tasks. World wide attention has been given to developing capable physical platforms that would be beneficial for rescue teams. It is evident that use of multi-robots increases the effectiveness of these systems. The Robot Operating System (ROS) is becoming a standard platform for the robotics research community for both physical robots and simulation environments. Gazebo, with connectivity to the ROS, is a three-dimensional simulation environment that is also becoming a standard. Several simultaneous localization and mapping algorithms are implemented in the ROS; however, there is no multi-robot mapping implementation. In this work, two multi-robot mapping algorithm implementations are presented, namely multi-robot gMapping and multi-robot Hector Mapping. The multi-robot implementations are tested in the Gazebo simulation environment. Also, in order to achieve a more realistic simulation, every incremental robot movement is modeled with rotational and translational noise.

Keywords
Robot Operating System, Gazebo, multi-robot, mapping, simultaneous localization and mapping, simulation, gMapping, Hector Mapping

1. Introduction

Mobile robotics have applications in various fields, such as the military, urban search and rescue (USAR), medical, space, entertainment, and domestic fields. As application areas of mobile robots have extended, they are expected to perform more complex tasks without any human input. Constructing a map of an unknown environment while keeping track of its own position within this map is a fundamental problem for an autonomous mobile robot and it is known as simultaneous localization and mapping (SLAM). Although there are several approaches to solve this problem, methods that can be tailored to the available resources and can be applied in real time are more popular. Two such algorithms are gMapping[1] and Hector Mapping.[2] These two algorithms, as well as some other single robot libraries, are implemented and are available within the ROS (Robot Operating System), which has become a niche platform. The ROS is licensed under an open source, BSD license and provides number of tools and libraries for developers. Yet, there are no multi-robot versions of any mapping algorithms implemented on the ROS. As multi-robot systems have started to play an important role in many robotic applications, the requirement of SLAM architectures has arisen to help the team of robots. For a multi-robot system working in a wide unknown area, having a good map of the environment is a main prerequisite for safe navigation and good exploration. With this motivation, multi-robot mapping architectures based on gMapping and Hector Mapping are proposed in this study. The proposed approaches are able to use several sensors to construct the map of the environment and are validated through experimental tests. Parallel optimization procedures are employed to minimize the

Computer Engineering Department, Yildiz Technical University, Turkey

Corresponding author:
Sırma Yavuz, Computer Engineering Department, Faculty of Electrical and Electronics Engineering, Yildiz Technical University, Istanbul, Esenler, 34220, Turkey.
Email: sirma@ce.yildiz.edu.tr

computational requirements. Moreover, the effect of integrated odometry in gMapping and adding odometry data to Hector Mapping are investigated. To achieve a more realistic odometry simulation, an existing model is reimplemented with rotational and translational noise.

Section 2 gives brief literature information about mapping algorithms and the simulation environment. In Section 3, the bases for incremental SLAM is given and the construction of gMapping and Hector Mapping over incremental mapping is explained. Section 4 explains the odometry noise model introduced into the simulation environment and defines the improvements made to original gMapping and Hector Mapping. In Section 5, multi-robot architectures for both multi-robot gMapping and multi-robot Hector Mapping are given. Section 6 gives the experimental setup and experimental results and Section 7 concludes the study.

2. Related work

Some tasks, like USAR where time is crucial, particularly focus on multi-robot systems. For this type of system, all robots try to explore the different parts of the environment they do not know. Having multiple robots is only of benefit when there is a good exploration strategy and a good SLAM ability for this strategy to work. Generally, the initial positions of the robots are assumed to be unknown and each robot generates its own map as it explores the area. Usually, a center having an optimal exploration strategy directs the robots toward a goal. For an operation center to apply an optimal exploration strategy, it has to collect the individual maps of the robots and merge them. Map merging where the initial positions of the robots are unknown is a difficult problem and has been widely studied in the literature.[3,4] On the other hand, for almost all of the real life applications including USAR tasks, initial positions of the robots are known and the communication architectures are quite reliable. As a result, map merging becomes less of a problem and more centric methodologies requiring less computational resources can be applied. Such systems can treat each robot as a mobile sensor, sensing the environment and sending their measurements to a center, as proposed here. The center only deals with multiple robot measurements in order to generate and keep one global map. Based on this map and the exploration strategy, navigation commands are generated and sent to the robots.

Graph-based single robot SLAM approaches consists of a front-end that constructs graph nodes and constraints, and a back-end that optimizes the graph. Nodes contain robot poses and measurements, whereas constraints are the distances between nodes.[5] Although back-end optimization mostly is carried out offline, there are on-line graph-based SLAM approaches too. The graph-based SLAM approach can be generalized to multi-robot mapping, given that constraints are defined for each robot map pair.[4,6] Howeverm, to be able to define constraints across robot maps excessive map data interchange is necessary.

In this study, multi-robot versions of the most popular grid-based mapping algorithms, gMapping[1] and Hector Mapping,[2] are proposed.

3. Incremental mapping

SLAM approaches calculate a posterior over current pose x_t, along with the map m, as in Equation (1), where u is the odometry measurement and z is the sensor measurement. The method is referred to as incremental mapping, since it produces an updated map with every new sensor input. The method runs in constant time independent of the size of the map:

$$p(x_{1:t}, m|z_{1:t}, u_{0:t-1}) \tag{1}$$

Based on Murphy et al.,[7] Equation (1) can be approximated as in Equation (2):

$$\cong p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}|z_{1:t}, u_{0:t-1}) \tag{2}$$

The first part of Equation (2) is called *RegisterScan*, with known poses and laser measurements and it is analytically solvable for the map building process. According to incremental mapping, it is also possible to approximate the second expression of Equation (2) by only using the current map m_{t-1} and the latest sensor data z_t. This assumption is given in Equation (3):

$$\cong p(x_t|x_{t-1}, z_t, u_{t-1}, m_{t-1}) \tag{3}$$

Incremental mapping aims to find x_t that maximizes Equation (3). To calculate the map m_{t-1}, *RegisterScan* can be used. The optimization of the Equation (3) is realized in two steps where First the robot pose is estimated as given in Equation (4) using odometry u_{t-1}, then the best robot pose estimation is calculated by using Equation (5):

$$\hat{x} = x_{t-1} \oplus u_{t-1} \tag{4}$$

$$x_t = \underset{x}{\operatorname{argmax}}(p(x|\hat{x}, z_t, m_{t-1})) \tag{5}$$

There are several incremental mapping mapping algorithms, using different methods for the optimization step defined in Equation (5). The optimization methods used in gMapping and Hector Mapping algorithms are based on hill climbing and Gauss–Newton, respectively.

3.1. gMapping

gMapping[1] is a widely used SLAM algorithm in robotic research. gMapping is a method that uses only the laser distance measurement sensor and passes to the mapping

Algorithm 1 gMapping

```
1:  procedure GMAPPING
2:  Input:
3:  $z_t$ the laser scan at tth time step
4:  $u_{t-1}$ the odometry measurement of previous time step
5:  Output:
6:  $S_t$ the sample set at tth time step
7:  $m_t$ the map at tth time step
8:      $S_0 \leftarrow \varnothing$
9:      for $t \leftarrow 1$ to $t_{end}$ do
10:         for all $s_{t-1}^{(i)} \in S_{t-1}$ do
11:             $\left\langle x_{t-1}^{(i)}, w_{t-1}^{(i)}, m_{t-1}^{(i)} \right\rangle \leftarrow s_{t-1}^{(i)}$
12:             $x_t^{'(i)} \sim P\left(x_t | x_{t-1}^{(i)}, u_{t-1}\right)$
13:             $x_t^{(i)} \leftarrow \underset{x}{argmax}\, P\left(x \mid m_{t-1}^{(i)}, z_t, x_t^{'(i)}\right)$
14:             $w_t^{(i)} \leftarrow updateWeight\left(w_{t-1}^{(i)}, m_{t-1}^{(i)}, z_t\right)$
15:             $m_t^{(i)} \leftarrow integrateScan\left(m_{t-1}^{(i)}, x_t^{(i)}, z_t\right)$
16:             $s_t^{(i)} \leftarrow \left\langle x_t^{(i)}, w_t^{(i)}, m_t^{(i)} \right\rangle$
17:             $S_t \leftarrow S_t \bigcup s_t^{(i)}$
18:         if required then
19:             $S_t \leftarrow resample(S_t)$
```

steps after selecting the optimum starting point with the help of odometry information.

gMapping is based on the Rao–Blackwellized Particle Filter (RBPF),[8] a multi-particle grid-based method. Each particle holds its own belief in the previous position of the robot and builds its own map. Each incoming laser scan updates the beliefs of the particles. The beliefs of the particles are based on the robot position and orientation. In addition, gMapping uses its own laser scan mapping method in the optimization step. The gMapping algorithm implemented in the ROS and used in the scope of this study is given in Algorithm 1. The set of all particles is given with S_t and each particle's reliability is given as a weight value $w_t^{(i)}$. Overall weight values sum up to one. $x_t^{(i)}$ represents the belief of the robot position and $m_t^{(i)}$ represents the map belief of particle i. z_t represents laser measurement and u_t shows the odometry data. gMApping consists of six important steps, as shown below.

- **Measurement acquisition:** a new laser scan is taken.
- **Scan matching:** performed by the pairing of previous and current laser scans.
- **Sampling:** with the help of previously taken particles $x_{t-1}^{(i)}$, suggested distribution $\pi(x_t|z_{1:t}, u_{0:t})$ is calculated to perform the best particles $x_t^{(i)}$.
- **Weighting:** each particle is given a weight $w^{(i)}$ as calculated in Equation (6):

$$w^{(i)} = \frac{p\left(x_t^{(i)}|z_{1:t}, u_{0:t}\right)}{\pi\left(x_t^{(i)}|z_{1:t}, u_{0:t}\right)} \tag{6}$$

- **Re-sampling:** particles whose weight is below a certain threshold value are redrawn from heavy particles.
- **Mapping:** each position example $x_{t-1}^{(i)}$ and all observations $p\left(m_t^{(i)}|x_{1:t}^{(i)}, z_{1:t}\right)$ based on map $m_t^{(i)}$ are calculated.

The gMapping algorithm produces maps in the occupancy grid type, which is supported by the ROS. This type of fusing data from different sensor sources can be held in a high-resolution grid. However, running gMapping in large-scale areas, where the number of grids increases, is considerably more costly than running it in small areas. The occupancy rate of a grid $p(x,y)$ can be found, as the number of grid points over the grid in total, by Equation (7):

$$p(x,y) = \frac{\#full}{\#full + \#empty} \tag{7}$$

3.2. Hector Mapping

Hector Mapping is a fast SLAM method that also uses grid maps.[2] The method highly relies on the fast scanning rates provided by modern Light Detection and Ranging (LIDAR) systems to produce accurate localization and mapping. The scan matcher used by the method is suitable for structured indoor environments. Since the method does not use any other odometry information, it does not require loop closure to build a good map. On the other hand, since the method does not handle the data association explicitly, it is very sensitive to lower scan rates. The success of the Gauss–Newton method used by the algorithm is highly relative where the error function and the initial guess directly effect the convergence of the algorithm.[9]

Hector Mapping method is given with Algorithm 2 . z_t shows laser data, i_t represents IMU (inertial measurement unit) data and L is the number of map resolution levels. x_t^ℓ is the position estimate at map level ℓ, whereas m_t^ℓ is the map constructed for resolution level ℓ.

The scan matcher approach used by Hector Mapping integrates two-dimensional (2D) SLAM from laser scans and three-dimensional (3D) navigation from the IMU. The scan matcher provides the full state information, including all translational and rotational displacements. and does not calculate pose graph optimization. It provides the full six-degree-of-freedom (6DOF) state, which includes a robot platform's translation and rotation. To obtain coherent 3D results from the IMU and other sensors, a navigation filter is used.

Algorithm 2 Hector Mapping

```
1: procedure HECTORMAPPING
2: Input:
3: $z_t$ the laser scan at $t^{th}$ time step
4: $i_t$ the IMU data at $t^{th}$ time step
5: Output:
6: $m_t^L$ the L-level map at $t^{th}$ time step
7:   for $\ell \leftarrow 1$ to $L$ do
8:     $x_0^\ell \leftarrow initPose()$
9:   for $t \leftarrow 1$ to $t_{end}$ do
10:    $pc_t \leftarrow 3dTransformLaser(z_t,i_t)$
11:    $z_t^* \leftarrow filterPointCloud_{z\_coord}(pc_t)$
12:    for $\ell \leftarrow 1$ to $L$ do
13:      if $\ell = 1$ then
14:        $\hat{x}_t^\ell \leftarrow initPose(x_{t-1}^L)$
15:      else
16:        $\hat{x}_t^\ell \leftarrow initPose(x_t^{\ell-1})$
17:      $g_{t-1}^\ell \leftarrow calcOccupancyGradient(m_{t-1}^\ell)$
18:      repeat
20:        $\hat{x}_t^\ell \leftarrow gaussNewton(g_{t-1}^\ell,\hat{x}_t^\ell,z_t^*)$
21:      until $optimisationcriteriamet$ $x_t^\ell \leftarrow \hat{x}_t^\ell$
22:      if $distance(x_{t-1}^\ell,x_t^\ell) > threshold$ then
23:        $m_t^\ell \leftarrow mapUpdate(m_{t-1}^\ell,x_t^\ell,z_t^*)$
```

The Gauss–Newton approach, optimizing the alignment of laser beam endpoints with the current map, is employed to calculate the best match between the map and laser scans. The general SLAM solution given in Equation (1) is optimized according to the error measure defined by Equation (8):

$$\sum_{i=1}^{r}[1 - M(S_i(x + \Delta x))]^2 \rightarrow 0 \tag{8}$$

In Equation (8), r represents the number of laser scan endpoints. The global coordinates of those laser endpoints $S_i(x)$ are calculated according to a transformation x and $M(S_i(x))$ is the map value at the coordinate of $S_i(x)$.

Afterwards, to optimize Equation (8), the update step is applied as in Equation (9):

$$\Delta x = H^{-1}\sum_{i=1}^{r}A^T[1 - M(S_i(x))] \tag{9}$$

where $H = \sum_{i=1}^{r} A^T A$
and

$$A = \nabla M(S_i(x))\frac{\partial S_i(x)}{\partial x} \tag{10}$$

To achieve a faster matching scheme, Hector Mapping uses a multi-resolution map. The idea is similar to the image pyramid approaches in computer vision, where each map has half of the previous map's resolution. Since all of these maps are consistent with each other, this method does not require down-sampling and saves computational resources. On the other hand, keeping all of the maps in memory and updating them during pose estimation increases the memory requirements, as well as the computational requirements. To estimate the pose, the scan alignment procedure is applied starting from the lowest resolution map without using any odometry information.

4. Initial improvements over the underlying simulation environment and algorithms

The multi-robot SLAM architectures developed in this study are based on existing single robot versions of the gMapping and Hector Mapping algorithms. Before utilizing these algorithms, some improvements are made to improve their accuracy and to reduce computational complexity. Also an odometry noise model is implemented and used with the Gazebo simulation environment to achieve more realistic results.

4.1. Improvements over the simulation environment

The position values are generated without noise in the Gazebo plugin for the robots, which is used in RRSL (RoboCup Rescue Simulation League) 2016. This situation does not fit a realistic simulation environment. With this motivation, we implemented the odometry noise model[10] for the *skid_steering* Gazebo plugin. According to this model, the relative difference between two posses from $\bar{x},\bar{y},\bar{\theta}$ to $\bar{x}',\bar{y}',\bar{\theta}'$ is represented by a concatenation of a rotation δ_{rot1}, a translation δ_{trans}, and another rotation δ_{rot2}. The odometry information can be calculated by Equation (11):

$$\begin{aligned} \delta_{rot1} &= atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\ \delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\ \delta_{rot2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot1} \end{aligned} \tag{11}$$

The errors are added to the calculated ground-truth odometry according to Equation (12). In this equation, ε_b is drawn from $\mathcal{N}(0,b)$ normal distribution:

$$\begin{aligned} \hat{\delta}_{rot1} &= \delta_{rot1} + \varepsilon_{\alpha_1|\delta_{rot1}| + \alpha_2|\delta_{trans}|} \\ \hat{\delta}_{trans} &= \delta_{trans} + \varepsilon_{\alpha_3|\delta_{trans}| + \alpha_4|\delta_{rot1} + \delta_{rot2}|} \\ \hat{\delta}_{rot2} &= \delta_{rot2} + \varepsilon_{\alpha_1|\delta_{rot2}| + \alpha_2|\delta_{trans}|} \end{aligned} \tag{12}$$

In Equation (12), α_1 is the standard deviation of rotation noise for unit rotation, α_2 is the standard deviation of rotation noise for unit translation, α_3 is the standard deviation of translation noise for unit translation, and α_4 is the standard deviation of translation noise for unit rotation.

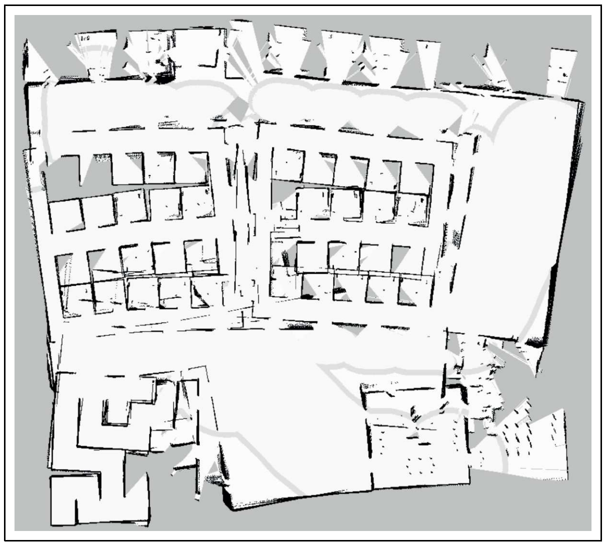

Figure 1. Multi-robot mapping results solely based on the noise added odometry.

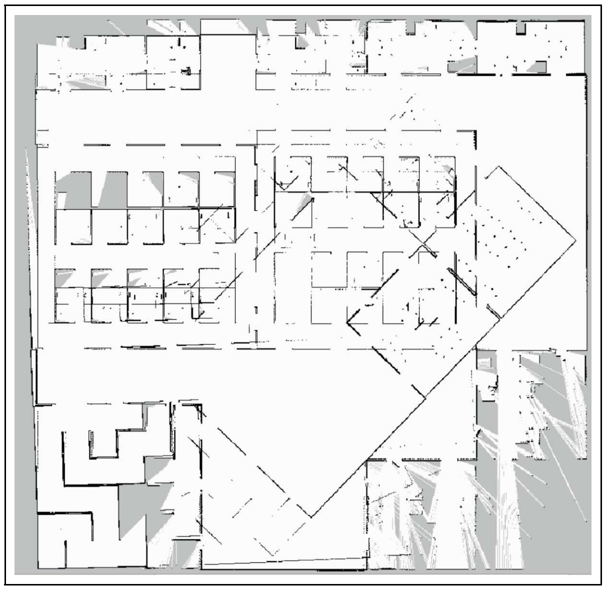

Figure 2. Multi-robot Hector Mapping without odometry data.

The noisy position value $\langle x', y', \theta' \rangle$ at time $t+1$ is obtained from $\langle x, y, \theta \rangle$ noisy position values at time t with Equation (13):

$$\begin{aligned} x' &= x + \hat{\delta}_{trans} cos\left(\theta + \hat{\delta}_{rot1}\right) \\ y' &= y + \hat{\delta}_{trans} sin\left(\theta + \hat{\delta}_{rot1}\right) \\ \theta' &= \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \end{aligned} \tag{13}$$

In Figure 1, the effect of the introduced noise model can be seen.

4.2. *Single particle gMapping*

gMapping needs an external odometry data and laser sensor data for mapping, whereas Hector Mapping solely depends on laser sensor data. One improvement over standard gMapping with the implementation given in this study is providing a fake odometry based on the Canonical Scan Matcher (CSM)[11] running between consecutive laser scans.

USAR missions require consistent map building to be used in exploration tasks; however, standard gMapping utilizes particle-based mapping and, as the best representing particle can change over time, the map can change drastically too, resulting in disturbed exploration. Keeping this in mind, gMapping is used in this implementation with one particle only.

4.3. *Hector Mapping with odometry*

Hector Mapping chooses not to utilize odometry data, with the idea that odometry data is noisy or not available in some situations. This is reflected in the best robot pose estimation given in Equation (4) by redefining it as $\hat{x} = x_{t-1}$.

Hector Mapping utilizes the Gauss–Newton method in the optimization step, which is quadratic and has some convergence issues.[12]

In Hector Mapping, the initial point of the optimization step is taken as the last optimized robot position.[2] This may result in the initial point of the mapping diverging from the global minimum if the robot displacement increases between sequential laser scans. As a result, the optimization procedure will not converge, causing the map to fail. In Figure 2, the effect of mapping without odometry data on Hector Mapping is given. If current odometry is more accurate than zero odometry, using Equation (4) as the initial point is more reliable.

5. Multi-robot simultaneous localization and mapping architectures

The multi-robot version of the SLAM problem can be defined with Equation (14), where n is the number of robots. Robots register their laser scans to the same map m:

$$p(x^j_{1:t}, m | z^j_{1:t}, u^j_{0:t-1}), j = 1 \ldots n \tag{14}$$

For each robot, an independent optimization step is applied as in Equation (15):

$$x^j_t = \underset{x^j}{\operatorname{argmax}}\left(p\left(x^j | \hat{x}^j, z^j_t, m_{t-1}\right)\right), j = 1 \ldots n \tag{15}$$

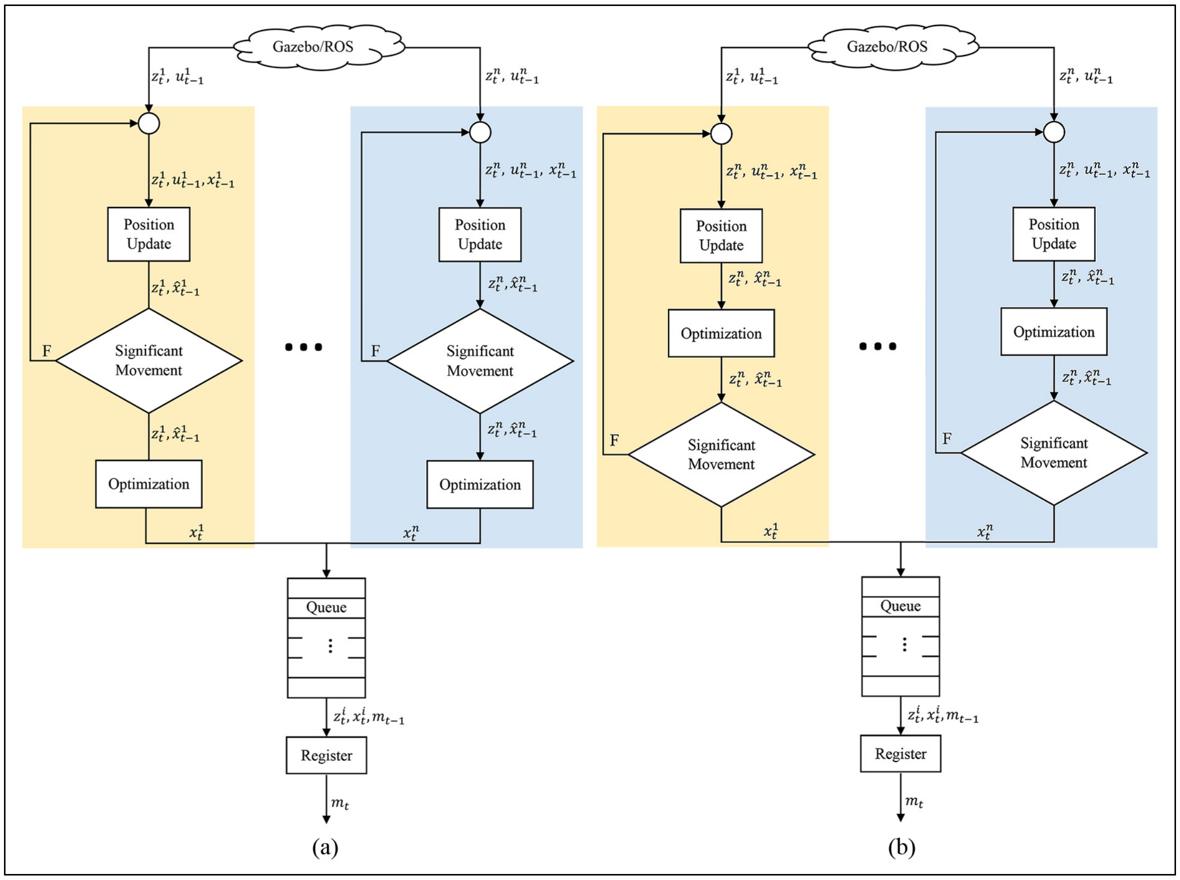

Figure 3. Multi-robot (a) gMapping, (b) Hector Mapping architecture. ROS: Robot Operating System.

5.1. Multi-robot gMapping

One possible solution of multi-robot map building is to run incremental mapping serially for each robot. However, with this method each robot should wait for the other robots' mapping steps, and this may increases the volume of unprocessed data. When we examined the whole process in detail, we figured out that the robot position update and optimization steps only read shared memory (map data). However, the register step writes to a shared memory. With this observation, the position update and optimization steps for each robot are parallelized.

The proposed architecture for multi-robot gMapping is given in Figure 3(a). Laser scans z and odometry data u are provided by Gazebo/ROS simulation for each robot, as shown in Figure 3(a). The proposed architecture consists of parallel optimization steps for robot pose estimation processes and a separate thread as the *RegisterScan* step for map construction.

In Figure 3(a), the *PositionUpdate* block corresponds to Equation (4), the *Optimization* block corresponds to Equation (5), and the *Register* block corresponds to the first part of Equation (2).

5.2. Multi-robot Hector Mapping

The multi-robot Hector Mapping architecture is given with Figure 3(b). As in multi-robot gMapping, z laser scans and u odometry data for each robot are obtained from Gazebo/ROS simulation. For robot pose estimation, parallel optimization steps and a separate thread, the *RegisterScan* step for map construction, are employed in the proposed architecture. The *Optimization* step produces a queue of optimized robot locations in an unordered fashion and the *RegisterScan* step retrieves from the first in, first out (FIFO) queue. In Figure 3(b), the *PositionUpdate* block corresponds to Equation (4), the *Optimization* block corresponds to Equation (5), and the *Register* block corresponds to Equation (2).

6. Experimental results

In this section, after giving the experimental setups and results for both multi-robot gMapping and Hector Mapping, an overall assessment will be made for the designed methods.

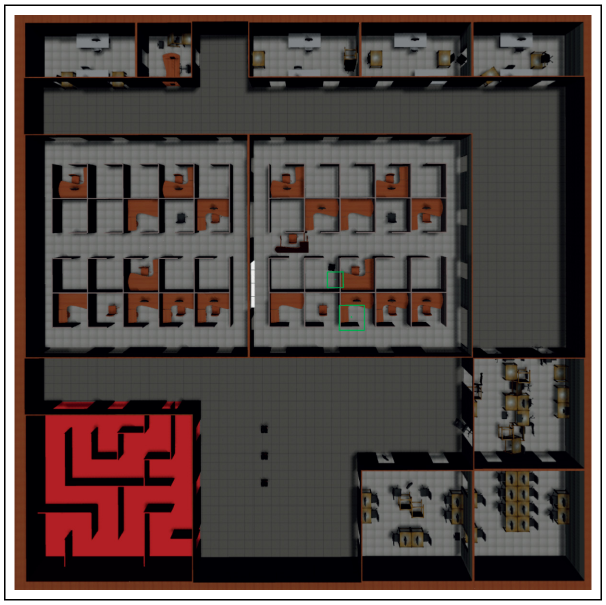

Figure 4. Gazebo simulation world.

6.1. Experimental Setup

The Gazebo simulation environment is initialized with one of the preliminary rescue mission maps of the RoboCup Virtual Robot Rescue Competitions that was held in 2012. This simulation world model consists of an office environment with several connecting wide/long corridors and a maze. The Gazebo environment can be seen in Figure 4 with the stated world file.

In the ROS environment, local coordinate frames are given with respect to a root coordinate frame. This structure is called the ROS Transformation Tree (TF-Tree). The TF-Tree for multi-robot mapping is given in Figure 5. Figure 5 shows that each robot has its own odometry data and laser sensor data to construct a commonly shared map.

The 2012 RRSL preliminary map is used to show the effect of this noise model. In Figure 1, the map is constructed without any SLAM algorithm. In the experiment, three robots were run with noisy position values.

For a realistic simulation, laser sensor readings are modeled with an additive Gaussian noise where $\sigma = 0.005$. Odometry data also contains a realistically modeled noise. Each movement can be broken down into three sub-actions, which are a rotation followed by a translation that is followed by another rotation. Odometry noise is added with respect to rotation to rotation (α_1), translation to rotation (α_2), translation to translation (α_3), and rotation to translation (α_4) error parameters.[10] The bag file is recorded with $\alpha_1 = 0.0002$, $\alpha_2 = 0.0001$, $\alpha_3 = 0.002$, $\alpha_4 = 0.001$ odometry noise parameter values. Gazebo enables the control simulation and ground-truth odometry data of the P3AT model by means of so-called Gazebo plugins. The original P3AT Gazebo plugin[13] does not have the aforementioned odometry noise model.

The multi-robot mapping problem with a relaxed constraint on approximate knowledge of initial start points for the robots enables sequential laser scan registration on the map, contrary to the general map merging solution. Approximate knowledge of the initial start points for the robots can be known for real world rescue tasks.

6.2. Multi-robot gMapping results

The maps belonging to single robot runs are given, together with simultaneous multi-robot gMapping results, in Figures 6(a) and 7(a), respectively.

Figure 7(a) shows the feasibility of our proposed multi-robot mapping. The optimization effect of simultaneous multi-robot mapping can be easily seen when Figures 1 and 7(a) are compared.

Figure 6(a) is obtained by running the proposed implementation for only one robot data. In other words, our algorithm can be effectively used with single or multiple robots.

6.3. Multi-robot Hector Mapping results

Hector Mapping uses multi-level grid cells for multi-level optimization to register laser scans to a map. As can be seen in Figure 2, multi-robot Hector Mapping without odometry data causes significant deformation in the map. In this experiment, in particular, the robot starting from the bottom left causes significant deformation in the joint map, and deformations for other robots are also prominent. Those errors are caused from the optimization step of Hector Mapping starting from an initial point, such as $\hat{x} = x_{t-1}$.

In Figure 6(b), single robot Hector Mapping results are given for comparison purposes. In this figure, one can also interpret overlapping paths for the robots.

With the use of odometry data as an initial start point for optimization, it becomes possible for Hector Mapping to converge more easily.

The result for multi-robot Hector Mapping that utilizes odometry data can be seen in Figure 7(b). From the results it can be seen that the usage of odometry can compensate for errors encountered with no odometry situation.

6.4. Results

In our detailed profiling analysis, we have seen that CPU consumption of the *Register* block is at least 10 times less than whole process for a single robot. In other words, our approach parallelizes 90% of each robot's processes.

Let A be the number of optimized scans per unit time with the single robot run. If the number of robots (N) is not higher than the CPU core number, the total number of optimized scans is almost $N \times A$, when robots are simultaneously running with our method.

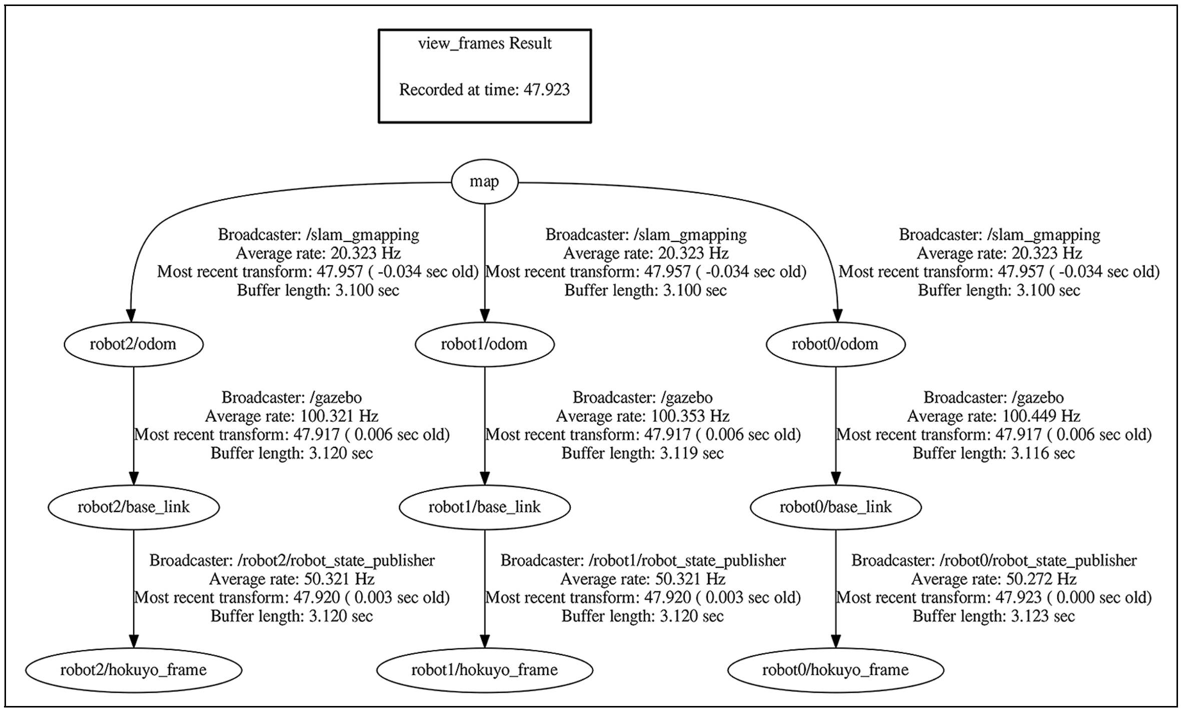

Figure 5. Robot Operating System Transformation graph for simultaneously operated multi-robot Gazebo simulation.

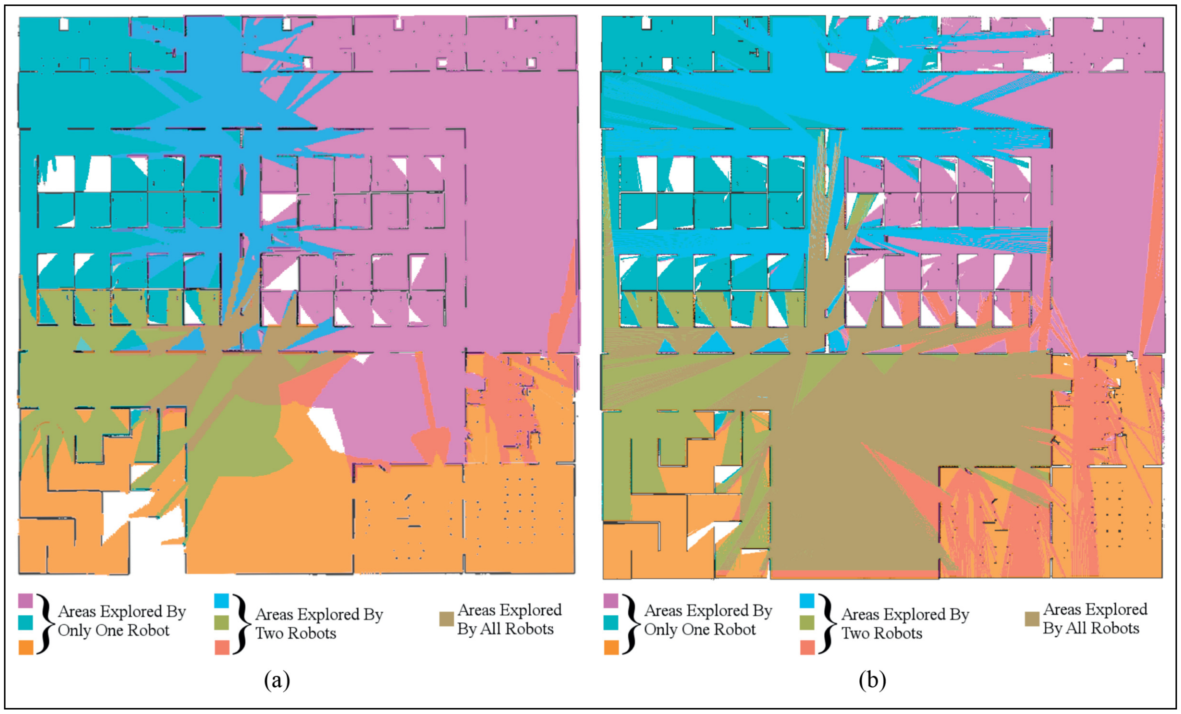
Areas Explored By Only One Robot
Areas Explored By Two Robots
Areas Explored By All Robots
(a)
Areas Explored By Only One Robot
Areas Explored By Two Robots
Areas Explored By All Robots
(b)

Figure 6. Single robot mapping results (a) gMapping, (b) Hector Mapping.

In our detailed analysis on gMapping with single robot, we have seen that 266 scans are processed and optimized for Robot0, 319 scans are processed and optimized for Robot1, and 187 scans are processed and optimized for Robot2. On the other hand, the proposed method with the *RegisterScan* step run as another thread processed and

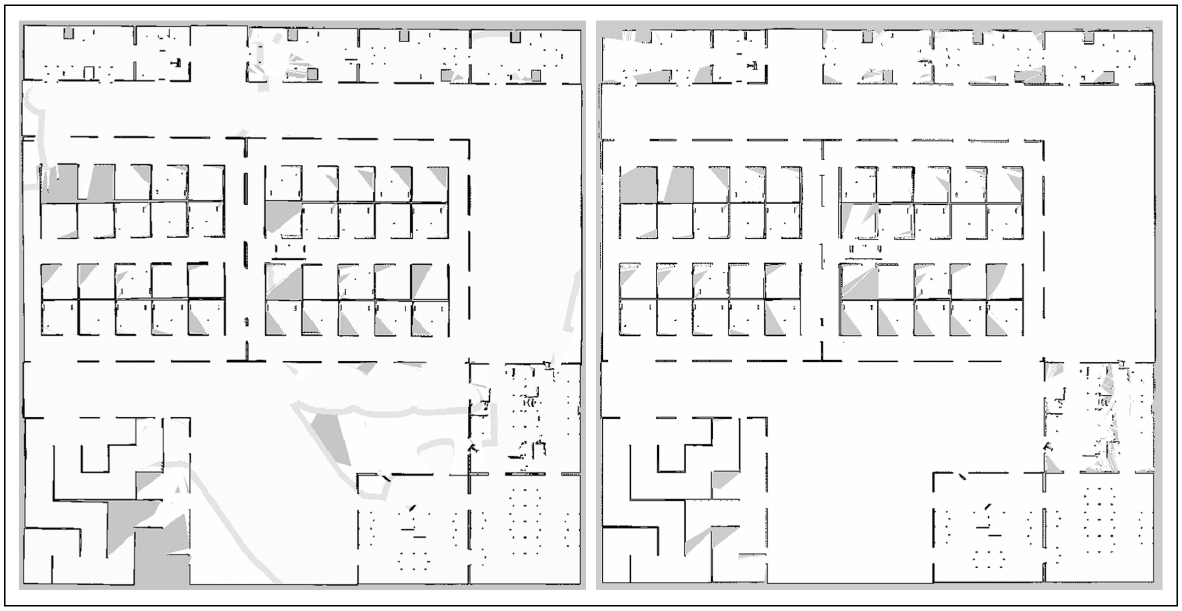

Figure 7. Multi-robot mapping results (left) gMapping, (right) Hector Mapping.

optimized 831 scans for Robot0, 1164 scans for Robot1, and 863 scans for Robot2 in 100 seconds for the same conditions. That meansthe original gMapping processed and optimized a total of 772 scans when it ran on three different nodes; however, the proposed method with the *RegisterScan* step run as another thread optimizes a total of 2858 scans. In other words, the proposed method can boost the performance of gMapping up to 3.7 times.

In our analysis, we have also seen that the original Hector Mapping processed 2858 scans for Robot0, 2934 scans for Robot1, and 2773 scans for Robot2 in 100 seconds with our bag files. However, the proposed method with the strictly serial *RegisterScan* step processed 2407 scans for Robot0, 2443 scans for Robot1, and 2371 scans for Robot2 in 100 seconds for the same conditions. That means the original Hector Mapping processed a total of 8563 scans when it ran on three different nodes. However, the proposed method with the strictly serial *RegisterScan* step processed a total of 7221 scans when it run on only the single node. The proposed method can parallelize 84% of the entire system.

Moreover, by running the *RegisterScan* step as another thread, where it is not required to wait for robot optimizations in order (say first Robot0, then Robot1, … etc.), the multi-robot Hector Mapping registration frequency exceeded the original Hector Mapping registration frequency. Mapping in only one grid level for both the original and the proposed methods optimized all scans. With two-level grid cells, the original Hector Mapping optimized 22 scans per second on average, whereas the proposed method with the separate thread *RegisterScan* step optimized 24 scans per second on average.

The difference in the number of optimized scans between multi-robot gMapping and multi-robot Hector Mapping mainly arises from the structure of the individual methods. In gMapping, scans are optimized if and only if a significant difference in the robot position is seen, whereas in Hector Mapping each scan is optimized but only the ones that result in a significant difference in the robot position are registered to the map.

The constructed multi-robot mapping results are also compared based on the average grid-based distance from ground-truth map. In this comparison, only occupied grids are taken into account and not the cleared areas. Occupied grids on the ground-truth map are paired in a 1-to-*n* approach with the multi-robot mapping results. The Euclid distance between the pairs is calculated for each grid and, based on these distances, the difference map is constructed. Then the average grid distance of the multi-robot map to the ground-truth is is calculated as the summation of grid distances in the difference map over the number of total grids. By definition, smaller average grid distances implies a more successful multi-robot mapping result. The average grid distance of gMapping is calculated as 0.0319 compared to the ground-truth map, and the average grid distance of Hector Mapping is calculated as 0.0258.

7. Conclusion

An architecture that allows multi-robot mapping for both gMapping and Hector Mapping algorithms has been developed in this study. The developed method adapts single particle mapping methods to multi-robot mapping. As gMapping is a multi-particle probabilistic method, a single particle structure was created for multi-robot mapping. Multi-robot gMapping is built on this structure and Hector Mapping is built on single particle architecture and no additional processing has been performed to transform it into multi-robot mapping. To develop a multi-robot version of the algorithm, update and optimization steps are parallelized for each of the maps and a queue structure maintaining optimized positions is implemented.

One disadvantage of the method may be considered as not having an explicit map merging strategy. When robots starts from different positions by sensing the same landmark, a small amount of error is introduced into the initial pose estimation. This will eventually reflect to the map as well. On the other hand, since it is usually possible to determine the starting positions of the robots, keeping computational complexity low by ignoring such specific cases is preferable.

Our group, the Yildiz Team, was awarded first place at RRSL in 2016 using the multi-robot gMapping version of these developed architectures.

The presented multi-robot mapping architectures are planned to be used along with a novel multi-robot exploration strategy to provide full autonomy.

Funding

This work was supported by the Turkish Scientific and Technical Research Council (TUBITAK-EEEAG-113E212 Project and TUBITAK-BIDEB-2211/E Program) and the Yildiz Technical University Scientific Research Projects Coordination Department (Project Numbers: 2015-04-01-GEP01 and 2015-04-01-KAP01).

References

1. Grisetti G, Stachniss C and Burgard W. Improved techniques for grid mapping with rao-blackwellized particle filters. Piscataway, NJ: *IEEE Trans Robot* 2007; 23: 34–46.
2. Kohlbrecher S, Meyer J, von Stryk O, et al. A flexible and scalable slam system with full 3d motion estimation. In: *proceedings of the IEEE international symposium on safety, security and rescue robotics (SSRR)*, Kyoto, Japan, 1-5 November 2011, pp.155–160. Piscataway, NJ: IEEE.
3. Martins JAS. *MRSLAM-multi-robot simultaneous localization and mapping*. Coimbra: University of Coimbra, 2013.
4. Lazaro MT, Paz LM, Pinies P, et al. Multi-robot slam using condensed measurements. In: Amato N *2013 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 3-8 November 2013, pp.1069–1076. Piscataway, NJ: IEEE.
5. Grisetti G, Kummerle R, Stachniss C, et al. A tutorial on graph-based slam. *IEEE Intell Transp Syst Mag* 2010; 2: 31–43.
6. Reid R, Cann A, Meiklejohn C, et al. Cooperative multi-robot navigation, exploration, mapping and object detection with ros. In: *2013 IEEE intelligent vehicles symposium (IV)*, Gold Coast Ciety, Australia, 23-26 June 2013, pp.1083–1088. Piscataway, NJ: IEEE.
7. Murphy KP, et al. Bayesian map learning in dynamic environments. In: Solla SA, T.K. Leen TK, and Müller k (eds.) *NIPS*, Denver, CO, 29 November - 4 December 1999, pp.1015–1021. Cambridge, MA: MIT Press.
8. Doucet A, De Freitas N, Murphy K, et al. Rao-blackwellised particle filtering for dynamic bayesian networks. In: Boutilier C and Goldszmidt M (eds.) *proceedings of the sixteenth conference on uncertainty in artificial intelligence*, Stanford, CA, 30 June - 3 July 2000, pp.176–183. Burlington, MA: Morgan Kaufmann Publishers Inc.
9. Balcilar M, Yavuz S, Amasyali MF, et al. R-slam: resilient localization and mapping in challenging environments. *Robot Auton Syst* 2017; 87: 66–80.
10. Thrun S, Burgard W and Fox D. *Probabilistic robotics*. Cambridge, MA: MIT Press, 2005.
11. Censi A. An ICP variant using a point-to-line metric. In: *robotics and automation, Pasadena, CA, 19-23 May 2008 (ICRA 2008). IEEE international conference on*, pp.19–25. Piscataway, NJ: IEEE.
12. Björck A. *Numerical methods for least squares problems*. Siam, 1996.
13. Koenig N. Pioneer 3-AT demo, https://github.com/nkoenig/pioneer3at_demo 2016 (accessed 13 April 2016).

Author biographies

Erkan Uslu is a research assistant in the Computer Engineering Department at Yildiz Technical University. He also received his MSc and PhD degrees from the same university.

Furkan Çakmak is a PhD candidate and a research assistant in the Computer Engineering Department at Yildiz Technical University.

Nihal Altuntaş is a PhD candidate and a research assistant in the Computer Engineering Department at Yildiz Technical University.

Salih Marangoz is an undergraduate student in the Computer Engineering Department at Yildiz Technical University.

Mehmet Fatih Amasyalı received the MSc and PhD degrees from Yildiz Technical University. He is also currently an associate professor at the same university.

Sırma Yavuz is an associate professor of Computer Engineering at Yildiz Technical University. She received her BE, MSc, and PhD from the same university. She is also a founder of the Probabilistic Robotics Group.