EXPERIMENT 3: MAPPING AND LOCALIZATION

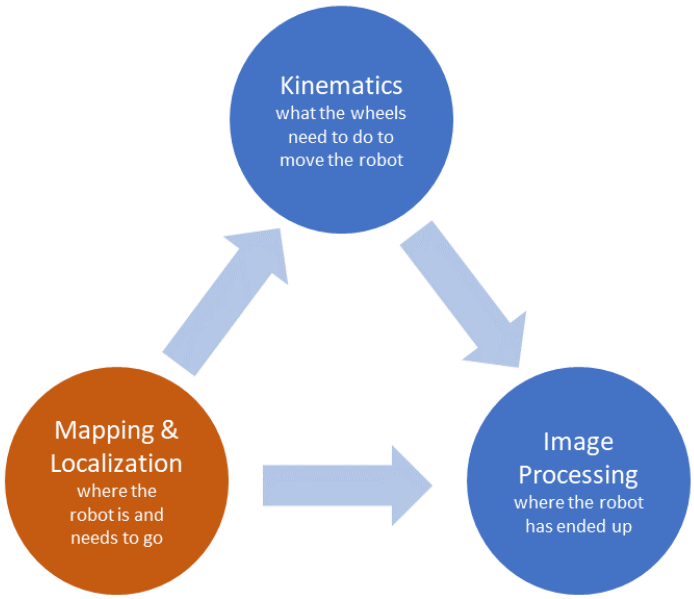
The purpose of this experiment is to create algorithms to map the environment around the Quanser QBot 3 Mobile Platform, and localize the robot inside that environment. The following topics will be studied in this experiment.

Topics Covered

- Occupancy Grid Mapping
- Particle Filtering

Prerequisites

- The QBot 3 has been setup and tested. See the QBot 3 Quick Start Guide for details.
- You have access to the QBot 3 User Manual.
- You are familiar with the basics of MATLAB® and SIMULINK®.



Mapping and localization is used to determine where the robot is

OCCUPANCY GRID MAPPING

In this lab you will learn how to use the on-board sensors of the Quanser QBot 3 Mobile Platform to autonomously build a map of the robotic environment through directed exploration.

Topics Covered

- Gathering and interpreting depth data from the Kinect sensor mounted on the QBot 3
- Autonomous 2D mapping of the surrounding environment QBot 3

1 Background

The Quanser QBot 3 Mobile Platform comes with a Intel RealSense sensor that has the ability to generate a depth map of the environment. This information, along with the location and orientation of the robot chassis, can be used for autonomous map building. To generate a 2D map, we will use the planar data received from the Intel RealSense depth image.

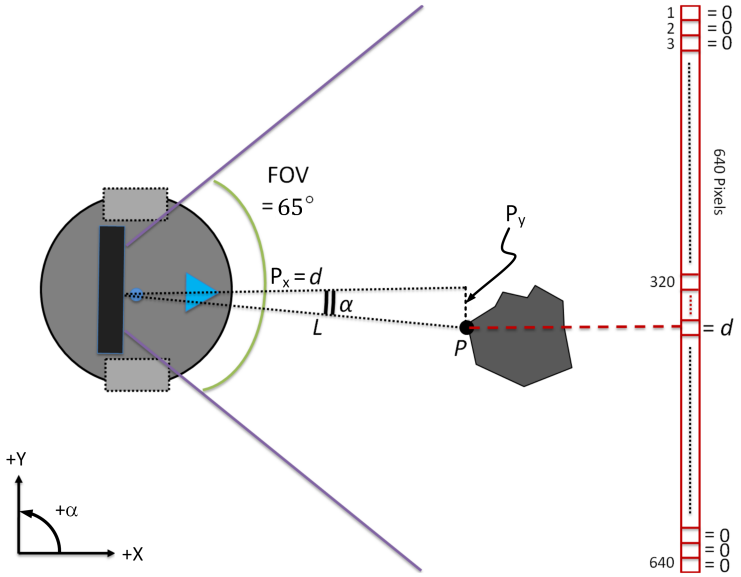


Figure 1.1: Depth data received from Intel RealSense sensor mounted on the Quanser QBot 3 Mobile Platform.

As is the case with any sensor, the Intel RealSense has some limitations:

- Range: The Intel RealSense sensor mounted on the QBot 3 has the ability to determine the distance to an object located between 0.5 m and 12 m. If an object is closer than 0.5 m or further than 12 m, the data is invalid and it will be zeroed down in the software. To map objects beyond this range, the robot should be moved accordingly, or other types of sensors can be used.
- Field of View: The horizontal field-of-view of the Intel RealSense is limited to 65 °. Therefore, to map the entire 360 °, the robot should be rotated accordingly.

Depth data received from the Intel RealSense sensor in QUARC is represented as a 480 $\times$ 640 depth image. For simplicity, we will only use one row of the depth image data for 2D mapping (each row includes 640 pixels) as illustrated in Figure 1.1.

The value of each pixel represents the distance (in millimeters) from the camera to the object, and should be mapped to the corresponding (x, y) point in the world coordinate frame. For example, assume we want to map the point *P*, shown in Figure 1.1, represented by the 400^{th} pixel of the centre row to the world coordinate frame. The value of this pixel received from the Intel RealSense sensor is d, which is the distance of the point *P* to the camera plane. Therefore $P_x = d$ (in millimeters). To calculate P_y the angle α is required as $P_y = L\sin(\alpha) = d\tan(\alpha)$.

The angle α can be calculated based on the distance from the desired point to the centre of the row (in pixels), and the horizontal FOV of the Intel RealSense sensor. In the example shown in Figure 1.1, α can be calculated as follows:

$$\alpha = (320 - 400) \times 65/640 = -7.12^{\circ}$$

where 320 refers to the central pixel of the sensor data. Therefore, we have $P_y = d\tan(11.6)$, or $P = (d, d \times \tan(-7.12))$ in the local coordinate frame of the QBot 3. Using odometric data, we can map the point *P* in the world

coordinate frame if we know the pose of the robot, (x, y, θ), where θ is the QBot 3 heading. Because odometric data initializes to zero when you start the robot, the origin of the map that you create is based on the initial location of the QBot 3.

As the robot moves around a 2D space, all of the points can be mapped to the world coordinate frame. These points can then be used to generate an occupancy grid map of that area collectively.

2 In-Lab Exercise

2.1 Autonomous Mapping

In this exercise, you will perform autonomous occupancy grid map generation for the QBot 3. Occupancy grid mapping is required for the robot to become aware of surrounding obstacles. The controller model for this exercise, shown in Figure 2.1, is called `QBot3_2D_Mapping_Manual.mdl`.

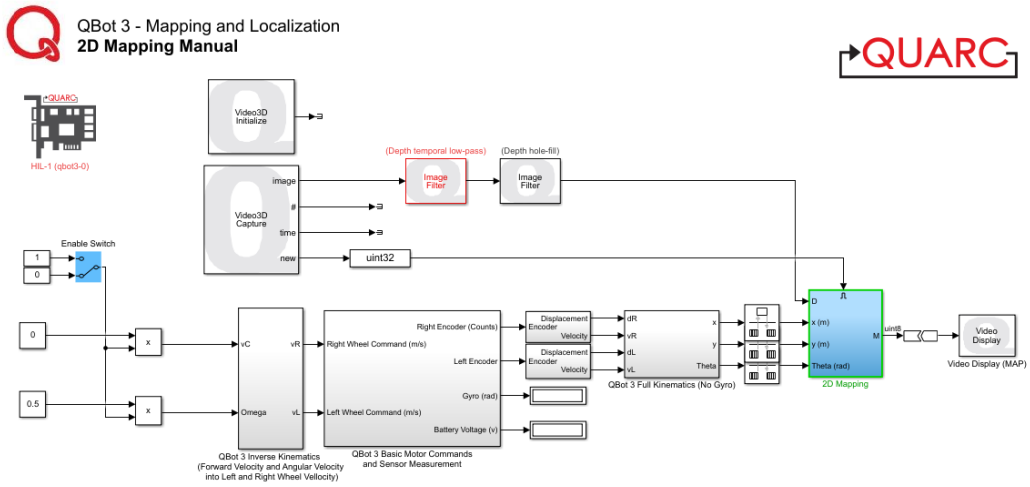


Figure 2.1: Snapshot of the controller model `QBot3_2D_Mapping_Manual.mdl`

The resulting vector is used in the QUARC models to match the depth data to real-world coordinates. Then open the `QBot3_2D_Mapping_Manual.mdl` model, compile it and go through the following exercises.

1. Set the robot linear velocity, v_c, and angular rate, ω, set-points to zero. Run the model, and look at the video display window showing the generated map. Put an object in front of the robot (make sure it is right in front of the Intel RealSense sensor, both horizontally and vertically) and 1 m away. Try to find the object in the map.

   Note: The map size is 300 pixels. The Max Room Dimension is set to 6 m by default. The implies a scale of 50 $\times$ (1 m will show as 50 units on the map). Adjust the Max Room Dimension number based on the largest length in your room (width or length).

   Note: The robot is shown on the generated map image window by a light gray circular shape as shown in Figure 2.3 with the forward direction indicated in the figure. You can change the scale of the robot on the map using *RobotScale* parameter in the To Display block diagram in the model.

2. What would happen if you used other rows of the depth image for mapping?

3. Move the object further away from the sensor (e.g. 1.5 m and 2 m) and follow the object on the map. Try moving the object closer to the sensor (but not closer than 0.5 m). Describe your observations and explain the behavior of the mapping data. Based on your observations, what do the white and gray areas and black dots represent on the map?

4. Stop the model.

5. Configure four objects surrounding the QBot 3 as in Figure 2.2. Now leaving v_c = 0, set ω to 0.3. Put the robot in a known initial configuration (facing the x axis on your map, shown in Figure 2.2), run the model and enable the manual switch once the robot is ready. Allow the robot to rotate twice and then disable the manual switch. Compare the generated map to the actual environment and examine any errors. The ideal generated map look similar to the one shown in Figure 2.3.

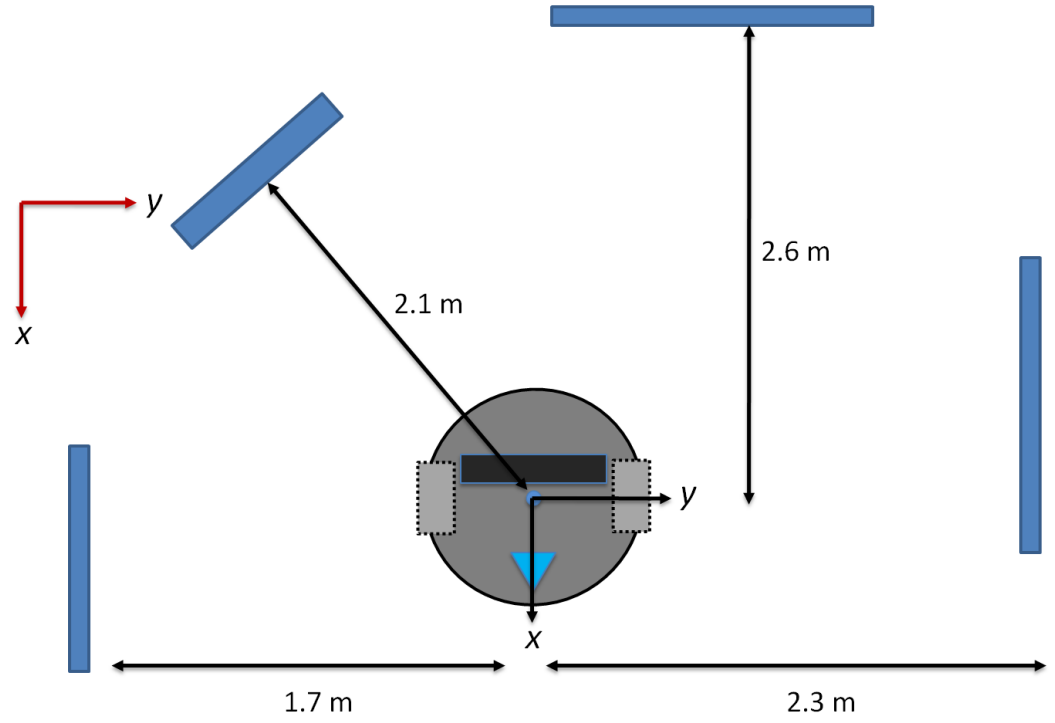


Figure 2.2: Configuration of the objects surrounding the QBot 3 for autonomous mapping.

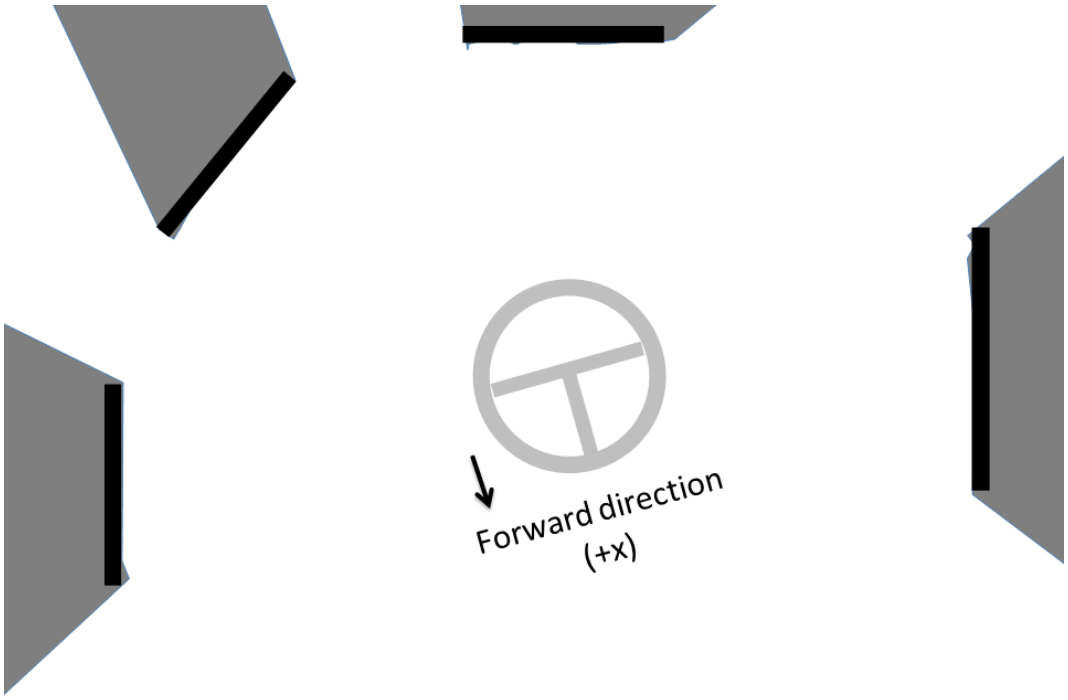


Figure 2.3: Generated map (zoomed in) and the current robot pose (position and orientation).

6. Click on the *Save* icon in the toolbar in the Video Display block. Save the file as `Map_1.bmp`. This will be used later.

7. What do you think are the potential sources of error in your measurements?
8. Explain how you could generate a 3D map of the environment (Vertical FOV of Intel RealSense is 40 °).
9. Now change v_c to 0.2, set ω to 0 and enable the manual switch. The robot starts moving forward. After about 0.5 m disable the manual switch.
10. Set v_c to 0, set ω to 0.3 and enable the manual switch so that the robot rotates for about 90 degrees, then disable the manual switch.
11. Run through steps 10 and 11 four more times.
12. What is the effect of these movements on the created map? Explain your observations.
13. Stop the model and turn off your robot.

ROBOT LOCALIZATION USING PARTICLE FILTERING

In this Lab you will learn how to use the depth and vision sensors of the Quanser QBot 3 Mobile Platform to accurately localize the robot.

Topics Covered

- Basic Knowledge of Particle Filtering
- Robot Localization using Particles

1 Background

Particle Filters are very versatile algorithms that can be used to "track" variables of interest as they evolve over time. For the purposes of robot localization, particle filtering is used to track the pose (position and orientation) of the QBot 3 in a given 2D map using the Microsoft Kinect sensor depth data as *sensory information*.

This algorithm first creates and randomly initializes multiple copies of the variable set, known as *particles*. For our experiment, each particle is essentially a copy of the QBot 3 including its position and orientation. Each particle is associated with a weight that signifies the accuracy of that specific particle's location. An estimate of the variable of interest can be obtained using the weighted sum of all particles. The particle filtering algorithm is iterative, with two fundamental phases as follows:

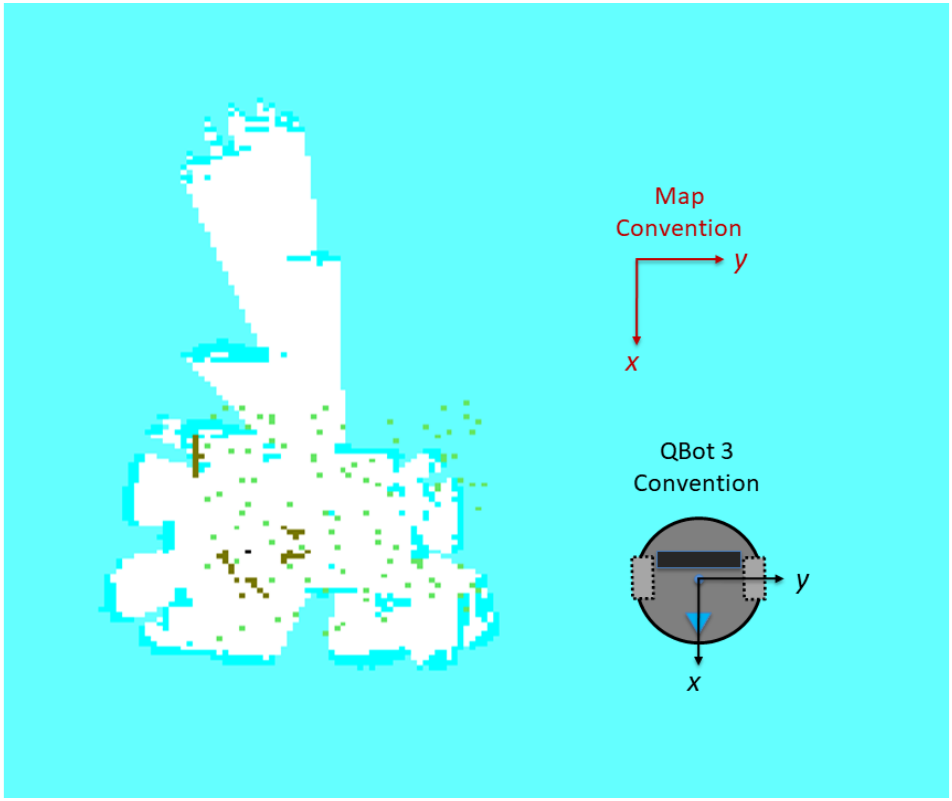


Figure 1.1: Example of the particle filtering algorithm used for localization of the QBot 3. the green dots represent the initial values for the particles, the red dots show the particles updated in real-time, and the black dot shows odometric data localization

- Prediction: In the prediction stage, the location of each particle is predicted according to the existing data and additional random noise to simulate the effect of sensory noise.
- Update: Then the weights of each particle are then updated based on the latest sensory information available.

At the end of the two stages the particles are evaluated, and the ones with small weights are eliminated. There are several particle filtering variations that utilize alternative methods to predict the particle locations and update the gains. In this experiment, we have kept the algorithm as simple as possible to convey the fundamental concept. The following describes the main steps in the algorithm used in this experiment:

1. Initialize N particles that contain position and orientation variables. Then randomly initialize the particles in a 2D map based on the size of the area around the robot.
2. Initialize the weights, W_i, where $i \in \{1, 2, .., N\} = 1/N$.

3. Move the QBot 3 robot by along both axis by Δ_x and Δ_y.
4. Apply the same motion (Δ_x and Δ_y) to all particles with added noise and update all particles.
5. Determine what features the robot would see (*sensory information*) if it had the pose of each particle in the given 2D map.
6. Compare the actual sensory data from QBot 3 with each particle's and determine the error.
7. Update the weights of all particles based on the error. The closer a particle's data is to the QBot 3 sensory information, the higher the weight.
8. Continue iterating through steps 3-7 until the particles converge on a single location.

Every time the robot moves to a new location, the particles will spread as they are actuated in a similar way, and errors are added to their location. Their locations will then converge after a few samples as the weights are adjusted. The sensory data received from Intel RealSense can be used either in the raw format (individual pixels) or in the processed form (features such as edges or corners).

Figure 1.1 shows an example of a particle filtering algorithm for localization of the QBot 3. Here the green dots represent the initial particles, and the red dots show the particles after the robot moves for a few seconds. The black dot is the location of the data based on pure encoder measurements. It is clear that the particles are converging to the actual location of the robot within the map.

2 In-Lab Exercise

In this exercise, you will become familiar with particle filtering, an iterative algorithm used for localization. The controller model for this exercise, shown in Figure 2.1, is called `QBot3_Particle_Filtering.mdl`.

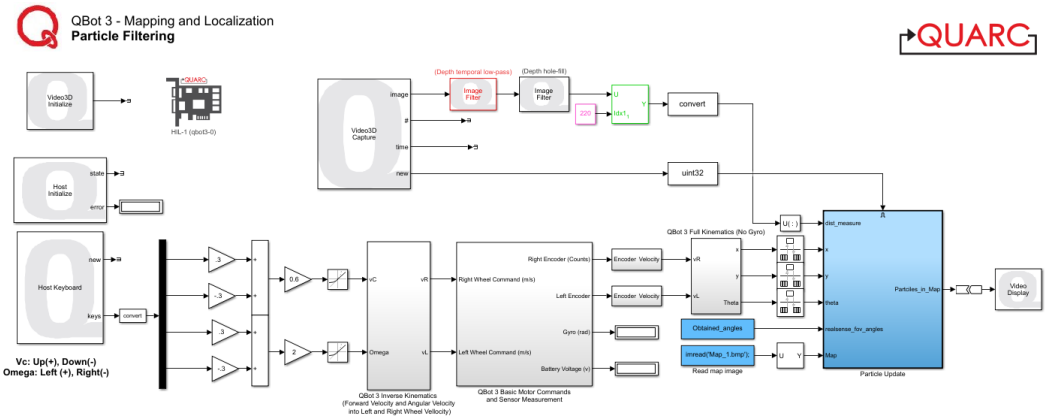


Figure 2.1: Snapshot of the controller model `QBot3_Particle_Filtering.mdl`

This experiment also loads the existing map you created in the previous experiment. Double-click the block in blue labeled Read map image and ensure that the constant value is set to `imread(`Map_1.jpg')`. This reads the image and converts it into a suitable format to be used by the model.

Make sure the robot's surroundings match the map you will be using. Put the QBot 3 in an initial location in the space, making sure the axes of the QBot 3 match of those of the map you have loaded as shown in Figure 1.1. Then go through the following steps.

1. Wait until the QBot 3 has fully initialized, and then enable the movement of the robot using the manual switch shown in Figure 2.1. This is an important step to insure that the sensor has fully initialized.
2. Make sure the QBot 3 is facing towards the x axis of the environment map as shown in Figure 1.1. The robot can be anywhere within your mapped region, but make sure it is no closer to any objects than 0.5 m so that you get valid non-zero data from the Intel RealSense sensor. To better understand this concept, try placing the robot about 0.5 m away from the origin of the map (the initial location of the QBot 3 when you created the map). In this case the odometric data will not represent the exact location of the QBot 3 (odometric data is always reset to zero when you start the robot, which is the origin of the map).
3. Compile and run `QBot3_Particle_Filtering.mdl`. Observe the initial particles in green, and their real-time location in red as well as the QBot 3 odometric location in black (which initially should be zero).
4. Wait for several seconds and observe the behavior of the particles (red dots).
5. Start rotating the robot slowly using the motor command slider gains and observe the particles as they update their location. Give the algorithm a few seconds to converge.
6. Now move the robot slowly within the map boundaries. Describe your observations.
7. Why do you think the particles tend to converge to multiple locations during the experiment?
8. Without blocking Intel RealSense sensor, approach the QBot 3 from behind and pick it up. Ask a friend to drive the robot so that the odometric data changes (by about 0.5 m) without the robot moving, then put the robot on the same location. Wait for the particles (red dots) to move and converge, and observe the black dot as well as the red dots. Explain the results of your observations.
9. Pick up the QBot 3 and manually move it 0.5 m closer to an obstacle. Wait for the particles to move and observe the black dot and the red dots. Compare your results to the previous case.

10. Double click on the Particle_Update block in the model and open up the particle_filter MATLAB function. Try to match the 8 steps described in Section 1 with the code.

11. Change the number of the particles in the function to 10, compile the model, and run through the above steps (steps 2 to 8). Observe the behavior of the algorithm with 10, 20, and 50 particles and compare your results to the previous cases.

12. Pick the most successful number of the particles from the previous step. Change the Gaussian noise distributions: *sigma_measure*, *sigma_move*, and *sigma_rotate*, that represent the noise of sensor measurements by multiplying them by 2 and 0.5. Compare your results to the previous cases.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com