EXPERIMENT 4: PATH PLANNING

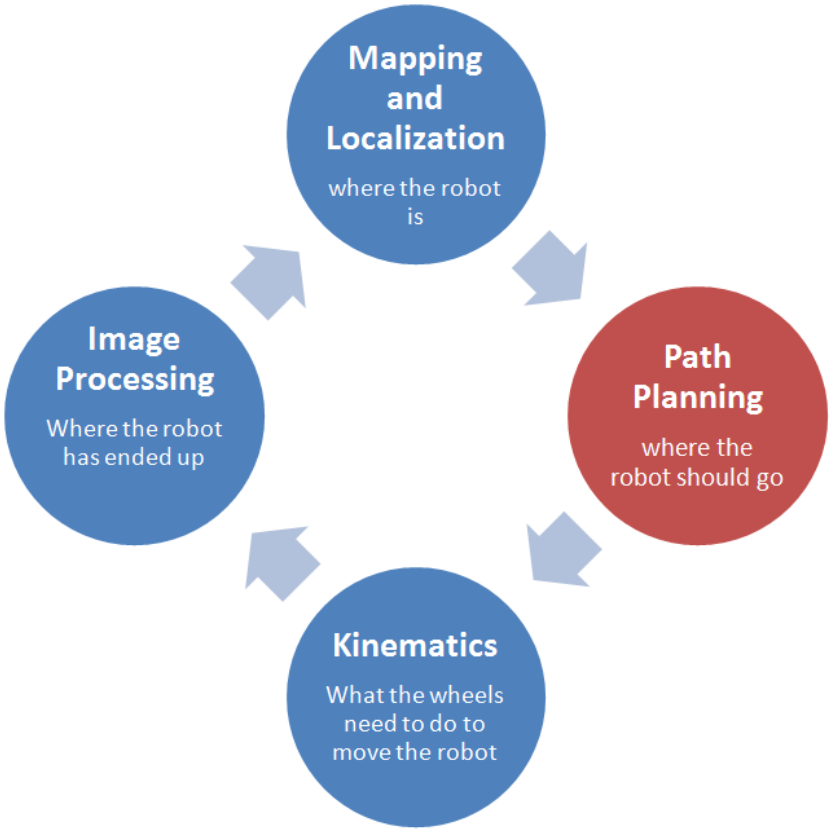
The purpose of this experiment is to create algorithms for the QBot 2 to navigate safely and avoid obstacles. The following topics will be studied in this experiment.

Topics Covered

- Different motion planning approaches including deliberate, reactive and hybrid methods.
- Different path planning algorithms such as A* and potential fields.

Prerequisites

- The QBot 2 has been setup and tested. See the QBot 2 Quick Start Guide for details.
- You have access to the QBot 2 User Manual.
- You are familiar with the basics of Matlab® and Simulink®.



Path planning is used to determine where the robot should go

PATH PLANNING

Motion planing plays a key role in autonomous mobile robot navigation. It chiefly involves interpreting the goals specified for the robot, along with the current mapping and navigational data in order to create a path to a target location. In this lab, you will learn how to design and implement path planning algorithms in order to navigate and avoid obstacles in a mapped environment. You will learn two important algorithms for path planning.

Topics Covered

- Potential fields
- A*

You will also learn how to implement motion control for QBot 2 using closed-loop control.

1 Background

The Quanser QBot 2 Mobile Platform comes with a Microsoft Kinect 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. The 2D map generated using this data will be processed to locate various obstacles in the map and create the "occupancy grid map" of the environment. In this lab we only focus on the path planning portion of this process.

Generally, global path planning is performed in robot's *configuration space*, where the robot is considered as a point object and the obstacle dimensions are increased by the maximum centroid-to-periphery dimension of the robot (in the case of the QBot 2, by the 18 cm radius of the body). Path planning in the configuration space provides a safety margin between the way points and the obstacle grids. An example of an occupancy grid map is shown in Figure 1.1 where the workspace is represented in pixels.

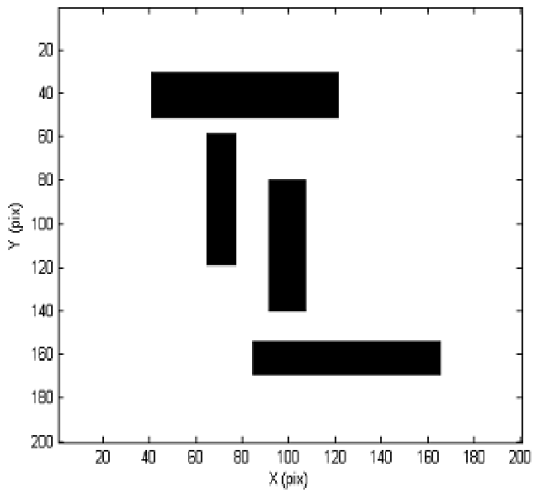


Figure 1.1: An example of an occupancy grid map where "pix" represents the pixels in the map.

Global path planning methods search for the connected grid components between the robot and target. There exist several techniques for global path planning; this laboratory will only describe the following two methods: Potential Field, and A* Diagram.

1.1 Potential Field

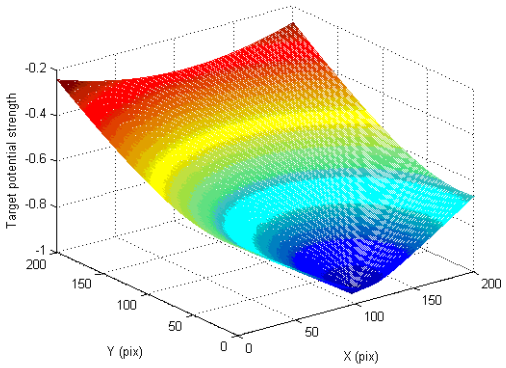
In this approach, the target exhibits an attractive potential field (U_{target}) and the obstacles exhibit repulsive potential fields (U_{obs_i}). The resultant potential field is produced by summing the attractive and repulsive potential fields as follows:

$$U = U_{target} + \sum U_{obs_i} \tag{1.1}$$

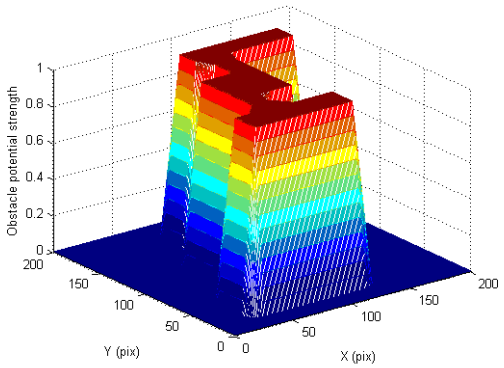
The net potential field is used to produce an artificial force equivalent to the gradient of the potential field with a negative sign as follows

$$F = -\nabla U \tag{1.2}$$

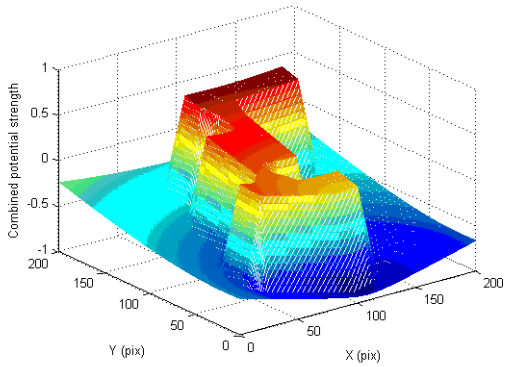
The motion is then executed by following the direction of the created force. To understand this, the potential fields are shown in Figure 1.2.



(a) Target potential field



(b) Obstacle potential field



(c) Combined potential field

Figure 1.2: Visualized potential fields for path planning

The target and obstacle potential field can be defined based on the distances to the target and the obstacles as follows

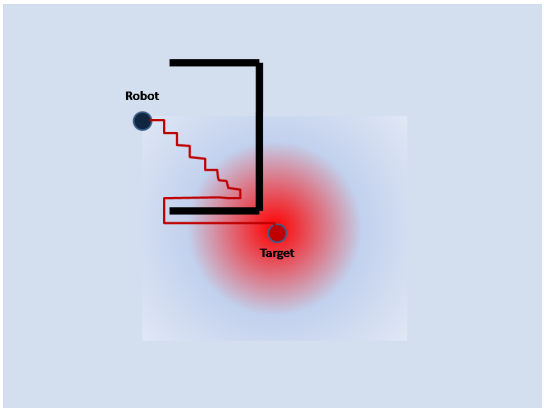
$$U_{target} = \frac{dist_{tar}(x,y)}{dist_{tar,th}} - 1 \tag{1.3}$$

$$U_{obs_i} = \begin{cases} 0 & \text{if } dist_{obs_i} > dis_{obs_i,th} \\ 1 - \frac{dist_{obs_i}(x,y)}{dist_{obs_i,th}} - 1 & \text{otherwise} \end{cases} \tag{1.4}$$

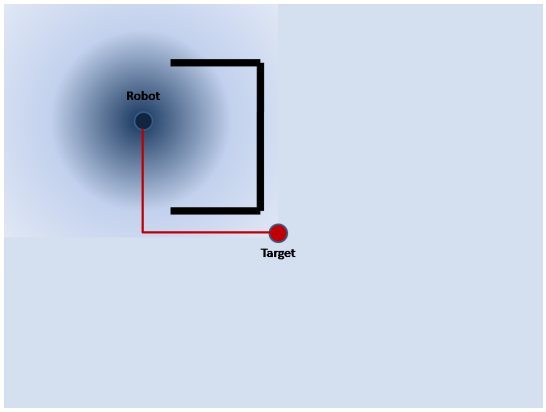
where $dist_{tar}(x,y)$ is the distance to the target (x,y) location of the workspace, $dist_{tar,th}$ is a normalization factor set to the diagonal distance of the workspace, $dist_{obs_i}$ is the distance to the i^{th} obstacle point from (x,y) location of the workspace and $dis_{obs_i,th}$ is the radial boundary of the obstacle potential field. A common problem of the potential field method is that the robot is trapped in a back-and-forth oscillatory motion when attractive and repulsive forces work in a straight line. To overcome this problem, a wall-following approach can be deployed where the robot follows the normal vector to the repulsive force. Tuning the threshold values will also help get better performance out of the algorithm.

1.2 A* algorithm

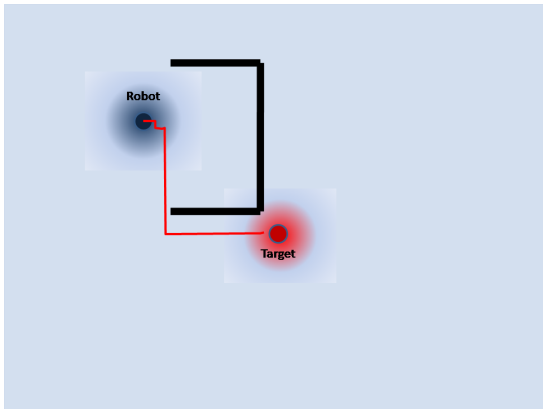
A* is the one of the most popular choices for path planning, because it's fairly simple and flexible. This algorithm considers the map as a two-dimensional grid with each tile being a node or a vertex. A* is based on graph search methods and works by visiting vertices (nodes) in a graph starting with the current position of the robot all the way to the goal. The key to the algorithm is identifying the appropriate successor node at each step.



(a) A* using path cost, g(n), exclusively



(b) A* using heuristic cost, h(n), exclusively



(c) A* complete

Figure 1.3: A* algorithm components and complete algorithm.

Given the information regarding the goal node, the current node, and the obstacle nodes, we can make an educated guess to find the best next node and add it to the list. A* uses a heuristic algorithm to guide the search while ensuring that it will compute a path with minimum cost. A* calculates the distance (also called the cost) between the current location in the workspace, or the current node, and the target location. It then evaluates all the adjacent nodes that are open (i.e., not an obstacle nor already visited) for the expected distance or the heuristic estimated cost from them to the target, also called the heuristic cost, h(n). It also determines the cost to move from the current node to the next node, called the path cost, g(n). Therefore, the total cost to get to the target node, f(n) = h(n) + g(n), is calculated for each successor node and the node with the smallest cost is chosen as the next point.

Using either the path cost g(n) or heuristic cost h(n) exclusively will result on non-optimized paths as shown in Figure 1.3. Figure 1.3a and Figure 1.3b show the result of using only g(n) and h(n) respectively. Together, an optimized path from the current node to the target node can be achieved as shown in Figure 1.3c.

2 In-Lab Exercise

This exercise consists of the following four steps

1. Creating an occupancy grid map around an obstacle using “Qbot2_2D_Mapping_Keyboard.mdl”.
2. Processing the created map to detect the obstacle using “Process_Map.m” MATLAB code.
3. Running different path planning algorithms using “Path_Planning_Map.m” MATLAB code.
4. Performing robot motion using “Qbot2_Path_Planning’_Motion_Control.mdl” model.

2.1 Setting up the environment and creating an occupancy grid map

You will have to first perform an occupancy grid map generation so that the QBot 2 is aware of its surrounding obstacles. The controller model for the mapping, shown in Figure 2.1, is called “QBot 2_2D_Mapping_Keyboard.mdl”.

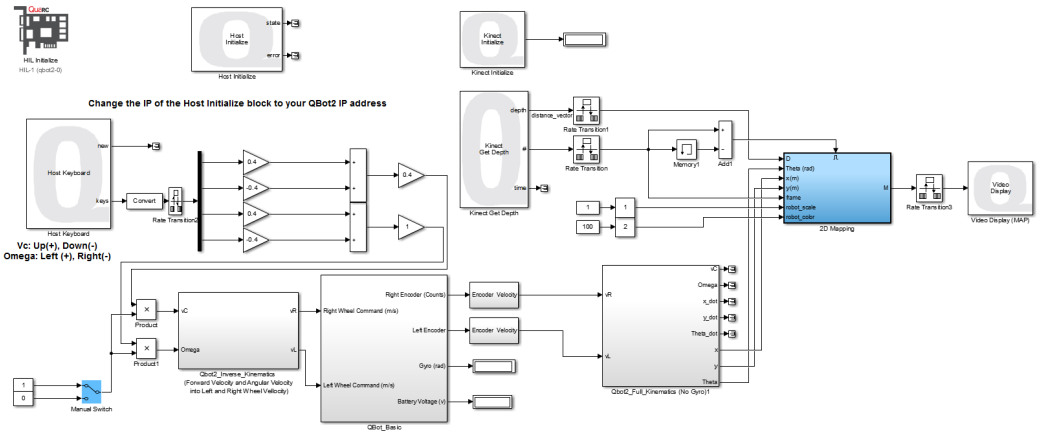


Figure 2.1: Snapshot of the controller model “QBot 2_2D_Mapping_Keyboard.mdl”

Before performing the experiment, make sure the Kinect sensor is flat so that it is parallel with the ground plane and not tilted. The resulting vector is used in the QUARC models to match the depth data to real-world coordinates. Then open the “QBot 2_2D_Mapping_Keyboard.mdl” model; make sure you change the IP address to your robot’s IP address in QUARC-Preferences menu option.

1. Find a zero pose for your QBot 2 where you have at least a 2.5m$\times$2.5m free space around it. Mark this initial position and orientation of the the QBot 2 as reference for the next steps. Find a box of at least 0.5m$\times$0.5m$\times$0.5m and place it about 1m in front of the robot as in Figure 2.2. Put the robot in a known initial configuration (Pose 1), shown in Figure 2.2), run the model and enable the manual switch once the robot is ready. Use the keyboard (up and down for linear motion; left and right for rotation) to move the robot around the obstacle, to poses 2-4, and then move it back to the initial start point. At each pose try to rotate the robot 360 degrees around itself so it can find all the free space around the obstacles.

   Note: Make sure the robot always stays at least 1 meter away from the obstacle as in Figure 2.2. Do not drive the robot closer to the obstacle.

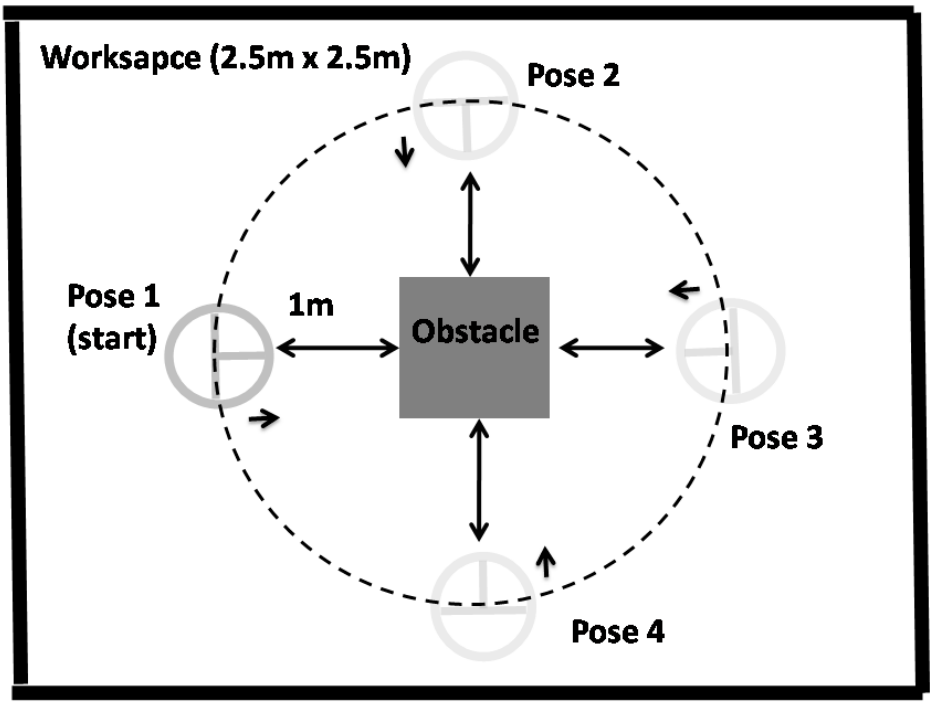


Figure 2.2: Configuration of the robot and obstacle for occupancy grid mapping.

2. When the robot is back to the initial pose, you should clearly see the box in the created map similar to Figure 2.3.

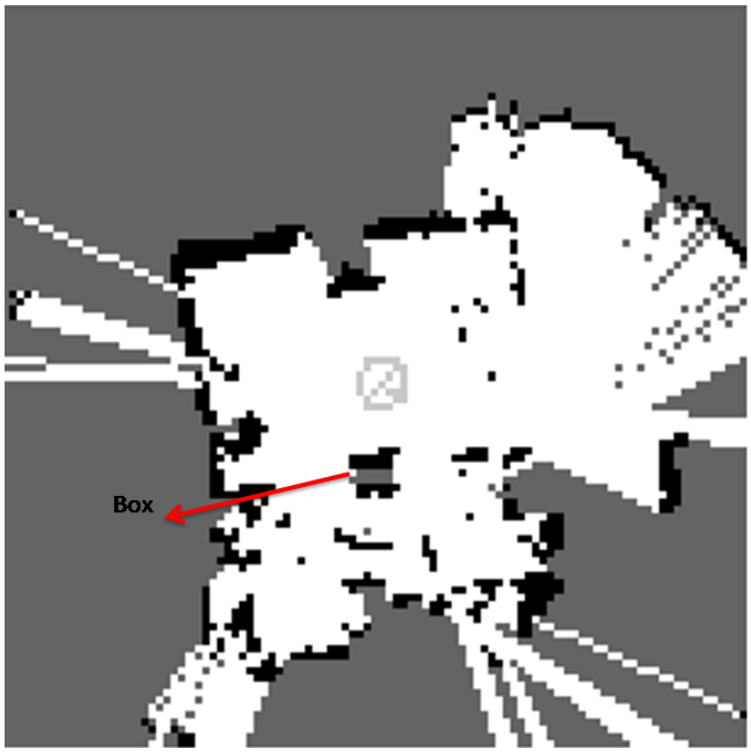


Figure 2.3: Configuration of the robot and obstacle for occupancy grid mapping required for path planning.

3. Right click on the map figure and select save to workspace. Save it as “Map_obs”. Go to the MATLAB workspace, find this item, right click on it and save it as “Map_obs.mat” in the same folder as the code is for future reference.

2.2 Path planning and motion execution

The next step is to process the map and find the coordinate of the obstacle in the map. This step ignores errors and imperfections and at the same time adds margins to the found obstacle. For the purpose of this lab, this pre-process MATLAB code (called Process_Map.m) is designed to use blob analysis and ignores larger blobs which represent for background areas.

1. Run the MATLAB function Process_Map.m. It creates a simplified occupancy grid map (look for x and y axes directions - x axis shows the direction of the initial pose of the robot) as in Figure 2.4 (only black rectangles at this point) and make sure it can find the obstacle and display it. It then asks you to click on the robot initial position as well as the target position. You should select (0,0) for the initial position since you start off from the centre of the map.

   Note: The resulting simplified map may appear to be rotated 90 degrees counter clockwise compared to the initial map you saved before; this is is because of the definition of initial axes direction.

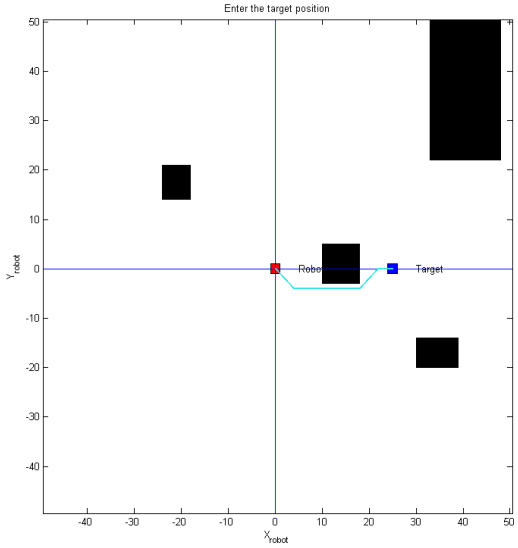


Figure 2.4: Path, start and target points for the robot shown in the simplified occupancy grid map using the A* algorithm.

2. Without closing this figure, open MATLAB function called “Path_Planning_MAP.m” and look for ”Method options:” in the code (line 10). Uncomment the “method = POTENTIAL_FIELD;” line and make sure the rest of the methods are commented out.

3. Run this MATLAB code and observe the path from start to the end (as in Figure 2.4).

2.3 Motion Execution

After the path is found, manually move the robot back to the starting point and make sure you maintain the initial orientation as marked in the first step. The x axis is the direction of the QBot 2 at this pose and the y axis can be found using the right hand rule (on the left side of the robot when standing behind it and looking forward) . By looking at the path plotted in the previous step (note the x and y axis values), try to estimate the motion of the robot (note

the scale of 10: 1m shows as 10 in the figure). If your estimated motion is not colliding with the obstacle, then go through the following path to execute the motion.

1. If the path in the resulting figure completely avoids the obstacles in the simplified map, now you can execute the motion. For this, Open the model called “Qbot2_Path_Planning_Motion_Control.mdl” shown in Figure 2.5.

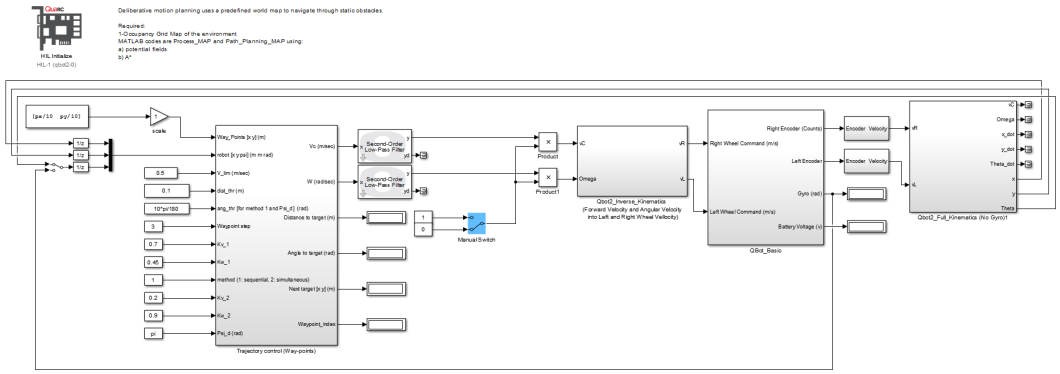


Figure 2.5: Snapshot of the controller model “QBot 2_Path_Planning_Motion_Control.mdl”

This model utilizes the path created in the previous step and controls the robot to move along this path. This model gets the target way points generated by the path planning algorithm as vectors and sends them one by one to the robot. Then it will compare the way points with the current position of the robot and uses a proportional controller in a feedback loop to control the pose of the robot. Take a moment and explore the model, particularly the feedback and the trajectory control block. Then go through the following steps to execute this motion.

2. Make sure that the Manual Switch is disabled. Compile this model and run it.
3. Enable the manual switch and wait for the robot to move. Observe the behavior of the robot and compare the motion with the path that is found. Once the robot arrives at the target, disable the manual switch.
4. Go back to the first step on Subsection 2.2 and this time use the A* method to find the path. Run through all the steps above and observe robot motion.
5. Open up the “potential_fields.m” as well as the “A_Star.m” MATLAB function, explain the algorithms used in both, and compare it with the discussion in the Background section.

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