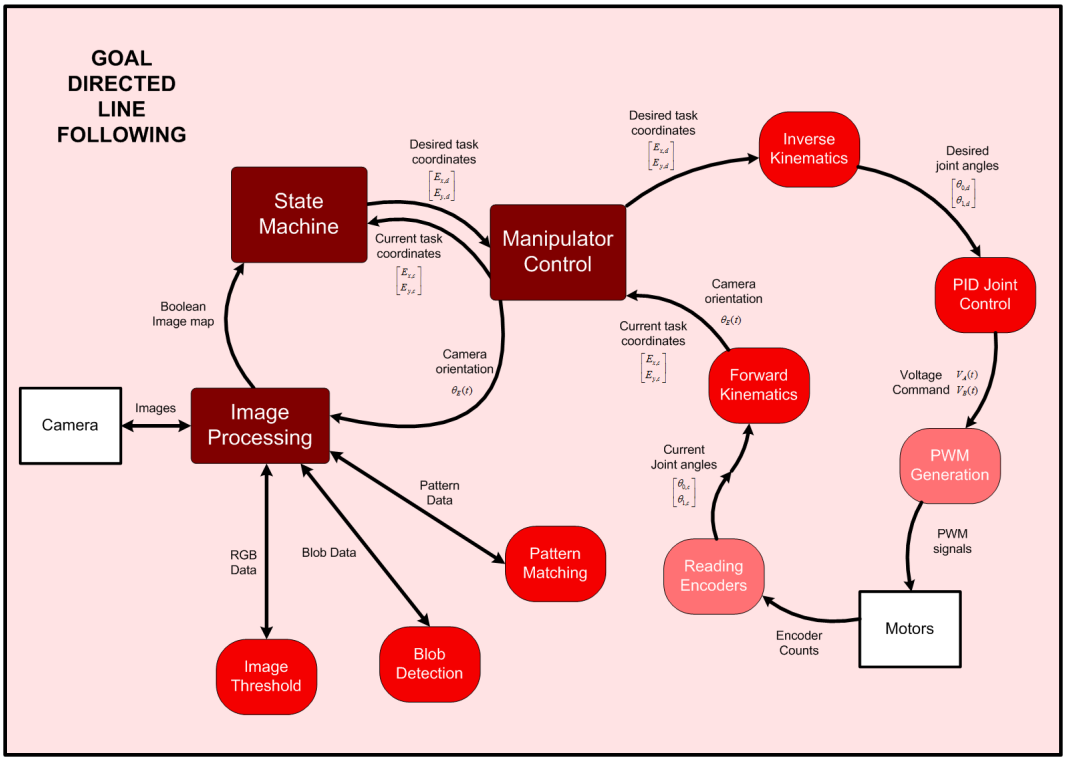
Goal-Directed Line Following

Topics Covered

- Goal directed line following using image processing, manipulator control and state machines
- Goal directed line following with the QNET Mechatronic Systems.

Prerequisites

- The QNET Mechatronic Systems is set up according to the Quick Start Guide.
- Manipulator Control laboratory experiment
- Image Processing laboratory experiment
- State Machines laboratory experiment



1 Background

Goal directed line following brings image processing, manipulator control and state machines together, to have the QNET Mechatronic Systems' manipulator move from point A to point B autonomously, while only travelling along roads. The following functionalities must be met,

1. The system should wait for the user to enter the final destination. This is the first point clicked on the silk image.
2. If required, the user can add additional way-points. These are added by CTRL-clicking on the silk image. Note that the system must follow these way-points in the order that they are clicked, and finally arrive at the final destination. All these points are added in a queue. When ready, the user presses GO!.
3. The system gets to the point at the top of the queue, by only following roads.
4. When the manipulator moves within range of a desired point, it eliminates the top point in the queue, and repeats step 3.
5. When the manipulator reaches the final destination point, the VI ends.

It is important that the 3 systems communicate information back and forth as shown in Figure 1.1.

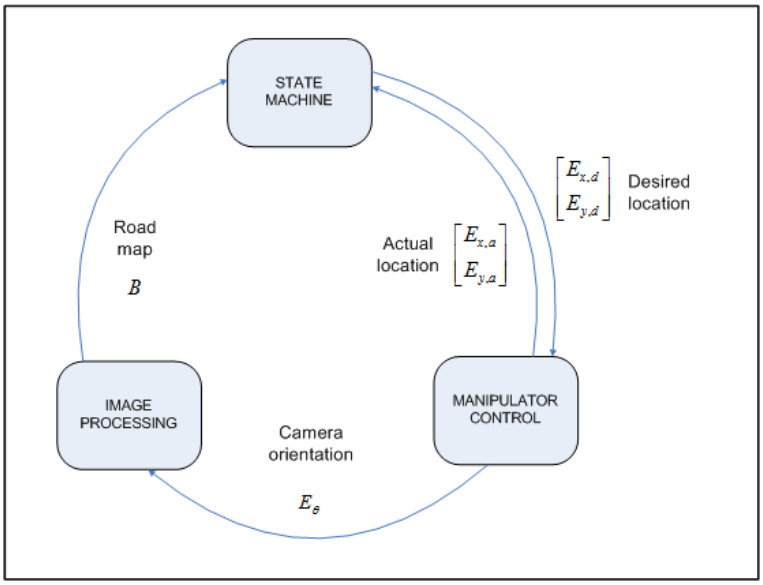


Figure 1.1: Communication between the 3 systems in goal-directed line following

1.1 Manipulator Control

The manipulator control loop is very similar to the one you developed in Manipulator Control laboratory experiment. It receives the desired end-effector task coordinates from the state machine. It uses inverse kinematics (see Inverse Kinematics laboratory experiment) to transform this into the desired joint coordinates. It uses a PID compensator (see PID Position Control laboratory experiment) to drive the manipulator to the joint coordinates. It transforms the actual joint coordinates into the actual task coordinates using forward kinematics (see Forward Kinematics laboratory experiment), and passes this information to the state machine. It also uses forward kinematics to pass the camera orientation to the image processing loop. This is shown in Figure 1.2.

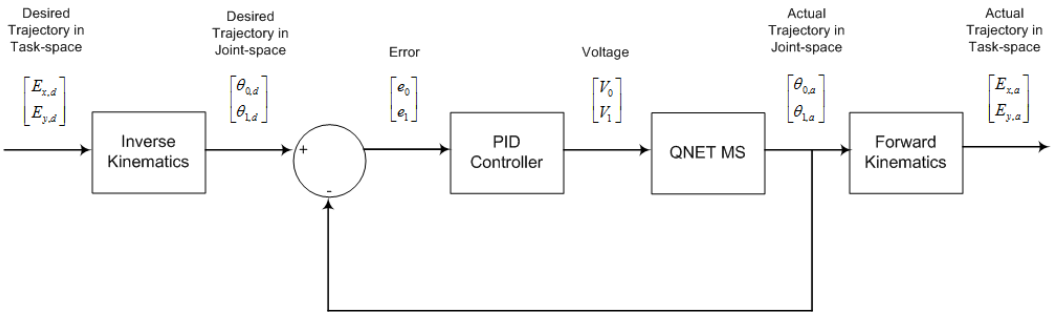


Figure 1.2: Manipulator control loop

1.2 Image Processing

The image processing loop receives a gray-scale image from QNET Mechatronic Systems's camera, and rotates it using information on the camera's orientation from the manipulator control loop. It uses thresholding to convert the image into a binary format (see Image Threshold laboratory experiment), and resamples the image to create a boolean map. The boolean map is a smaller binary image, but instead of a number at each pixel location, it has a boolean variable reading TRUE (road present) or FALSE (not a road). It passes this boolean map to the state machine. This is shown in Figure 1.3.

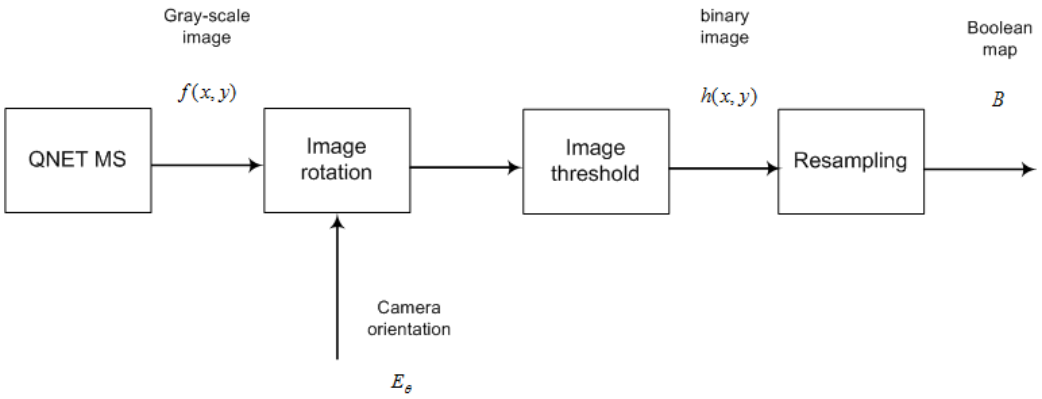


Figure 1.3: Image Processing loop

1.3 State Machine

The state machine is similar to the one presented in the State Machines laboratory experiment. It is designed using 5 states, as shown in Figure 1.4.

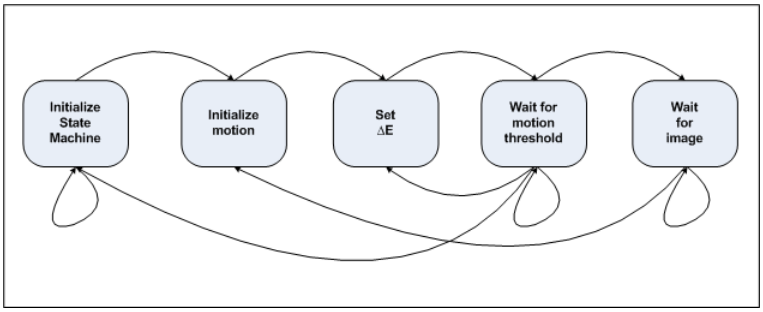


Figure 1.4: State Machine

In the Initialize State Machine state, the system waits for the user to enter the points they would like the system to go through. The first point is entered by clicking on the silk image, and is considered as the final destination. The manipulator goes to this point last, because it is always kept at the end of the queue.

Additional way-points can be added in this state as well. These are inserted at the top of the queue, but maintained in the order they are CTRL-clicked in. Thus, the 2nd point clicked on the image will remain at position 1, the third point clicked will remain at position 2, and so on. When the user presses GO! the system moves to the next state.

The Initialize Motion state sets a priority list for directions in which the boolean map B should be checked for TRUE values, based on the current location of the manipulator (from the manipulator control loop) and the desired point from the top of the queue.

The Set Delta E state selects a point from the boolean map B (from the image processing loop) that is TRUE in order of the direction priorities, and converts the distance between the corresponding pixels, to a distance that the manipulator has to traverse in centimetres (refer to the Pattern Matching laboratory experiment to recall the vector kinematics). This distance is passed on to the manipulator control loop.

The Wait for Motion Threshold checks until the current manipulator location (from the manipulator control loop) is close enough to the desired setpoint location, and if it is, goes back to Set Delta E, for the next point. When all the points are exhausted, the systems waits in Wait for Image till a new image is available. This ensures that the latest image corresponds to one taken at the current location. With the new image, the system goes back to Initialize Motion, and repeats the cycle, until the point in the queue is reached.

When a point from the top of the queue is reached, it is removed from the queue, and the process repeats for the next point. When all the points are removed, the system terminates and ends the VI. A sample flow has been shown in Figure 1.5.

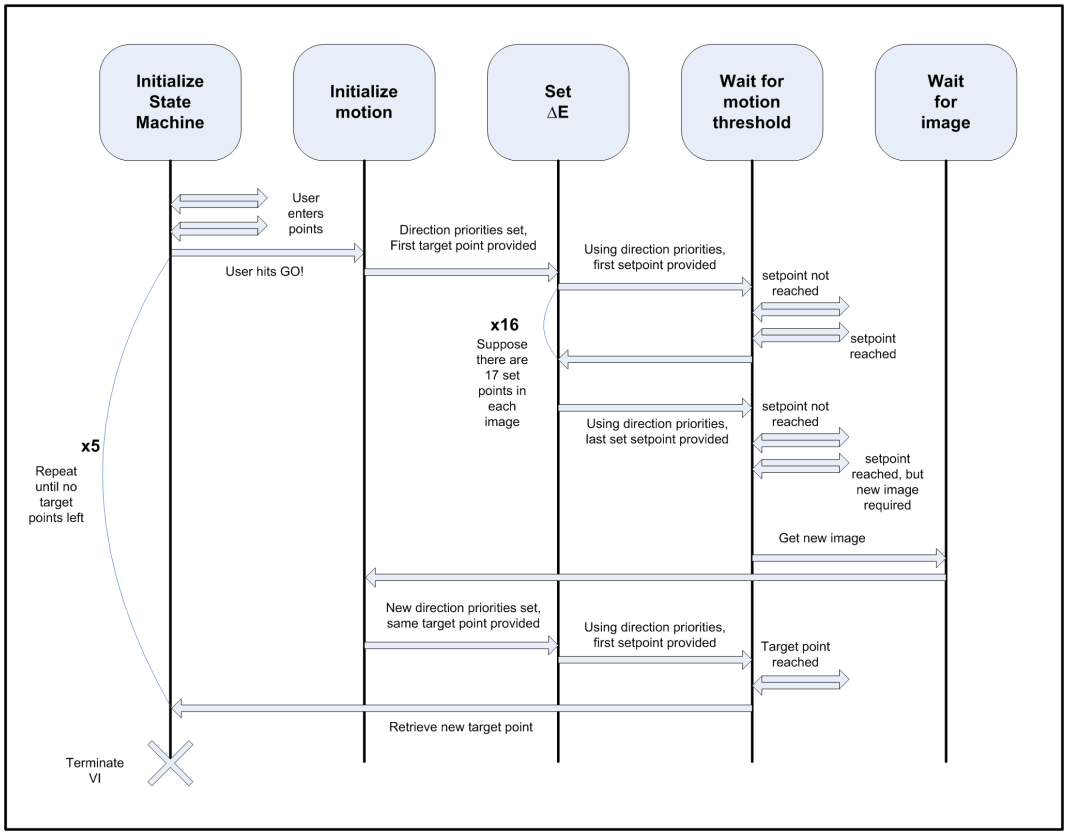


Figure 1.5: State machine flow example

2 In-Lab Exercises

2.1 Connecting System Loops

1. Open `Mechatronic Systems.lvproj`, and under `Quanser ELVIS RIO | Subsystems`, open `Goal directed line follower.vi`. Open the block diagram and scroll to the Manipulator Control loop. Connect the Desired task coordinates node to the Task coordinates (cm) input of the Inverse Kinematics subVI. Connect the Task coordinates (cm) output of the Forward Kinematics subVI to the Current task coordinates node. Also connect the Camera orientation (deg) output of the Forward Kinematics subVI to the Camera orientation node.

2. Scroll to the Image Processing loop. Connect the Camera orientation node to the Camera orientation (deg) input of the Image Processing subVI. Also connect the Real-time boolean path map output of the Image Processing subVI to the Boolean map node.

3. Scroll to the State Machine loop, and browse to the Initialize State Machine state. Connect the Current task coordinates node to the Current location node of the state machine. Connect the Boolean map node to the Boolean map node in the state machine. Lastly, connect the Desired location node in the state machine to the Desired task coordinates node.

2.2 Goal Directed Line Following

1. Run the VI. When the Calibration bar is full, click on the Silk image at the point shown in Figure 2.1. This is the final destination point. Click the GO! button. Does the system get to the final destination point? Stop the VI and give your answer based on the System State.

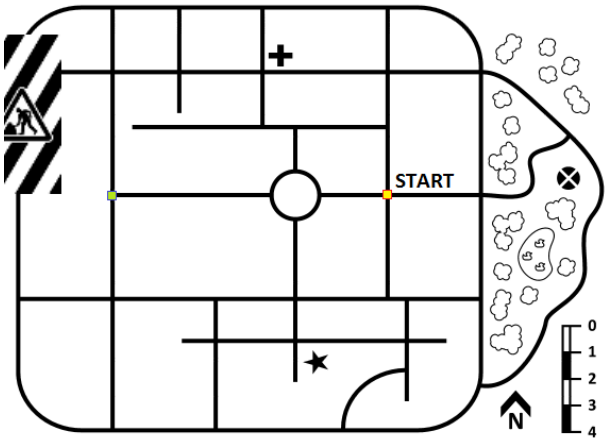


Figure 2.1: Final destination on the front panel Silk image

2. In the block diagram, browse to the Waiting for Motion Threshold state in the state machine. Take a look at the loop which checks if you are close to the desired point (flag 2). Note that the Threshold input to the sub-VI is set to 0.01 cm. Thus, the state machine waits until the manipulator is within 0.01 cm of the desired position. Why is this problematic?

3. The square boolean map is constructed by resampling every other pixel from the rectangular 160 px by 128 px source image, and the centimetre to pixel conversion is 0.0275 cm/px. What is the minimum distance between the current and desired task coordinates? Change the Threshold input to the sub-VI in the previous step, from 0.01 cm to 0.2 cm and comment on this choice, based on the fact that the roads are 0.1 cm thick. Run the VI with the same final destination as in the previous step. Has the performance improved?

2.3 Using Way-points for Navigation

1. Run the VI. Select the final destination as shown in Figure 2.2. How many paths can the manipulator take to reach this point? Click on the GO! button. Does the manipulator reach the final destination? Stop the VI and comment on the result.

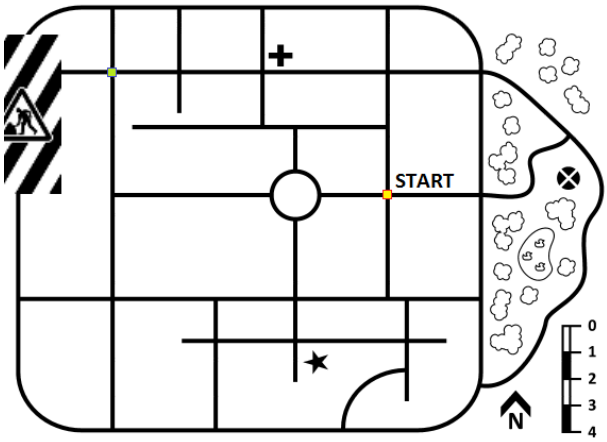


Figure 2.2: Final destination on the front panel Silk image

2. Run the VI. Select the final destination, point 1, as shown in Figure 2.3. Add two way-points by CTRL-clicking at point 2 first, and then point 3. Note that the manipulator will first get to point 2, then point 3 and finally to point 1, because the system will get to the final destination after following the way-points in the order they are clicked in. Does the manipulator reach the final destination? Stop the VI and comment on the result.

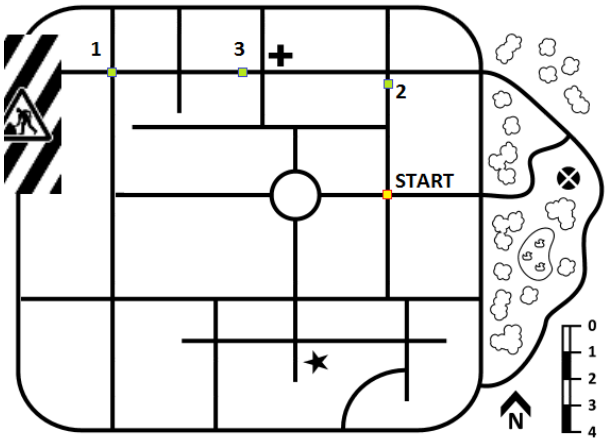


Figure 2.3: Final destination and way-points on the front panel Silk image

Way-points improve the system's performance and should always be placed to avoid ambiguity in the path to be followed. By default, the state machine provides a direction priority that is based on the distance between the current location and the final destination. Thus, if the destination point is further east than it is north, it will always take an east roadway, even if it leads to a dead-end because the system does not think ahead. It decides the route on a need-to basis.

2.4 Construction Detour

Recall from the Forward Kinematics laboratory experiment that the QNET Mechatronic Systems should not be able to achieve any desired points in the region shown in Figure 2.4. If a target point makes the manipulator go through

this route, it should request a detour to the final destination point. Thus, the state machine should erase all way-points that it still has left to follow, maintain it's final destination, and request new way-points from the user in the Initialize State Machine state. The goal in this exercise is to modify the state machine to account for this functionality.

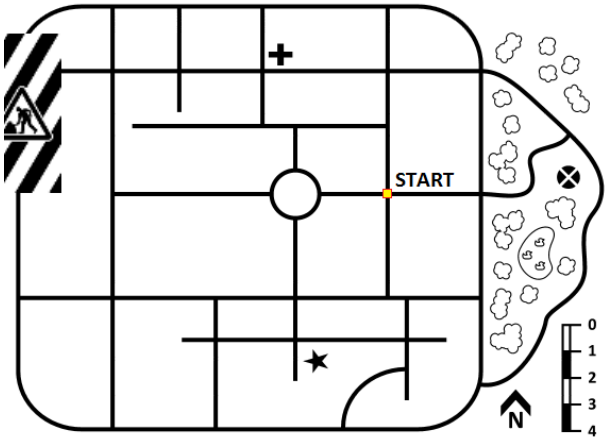


Figure 2.4: Construction zone on the QNET Mechatronic Systems silk

1. The first step involves detecting that the manipulator is in the construction zone. In the block diagram, browse to the Initialize Motion state in the state machine. Enter the sub-VI with a construction symbol on it. Update the code to that shown in Figure 2.5. This includes a detection if the manipulator is within a 2 cm by 4 cm region centred at (0.15 cm, 5 cm), next to the north-west traffic light. If it is, the next state should be New route request, instead of Set Delta E, which is handled by another sub-VI labelled Quanser next state. This construction sub-VI is also located in the Wait for motion threshold state of the state machine, but will get updated automatically.

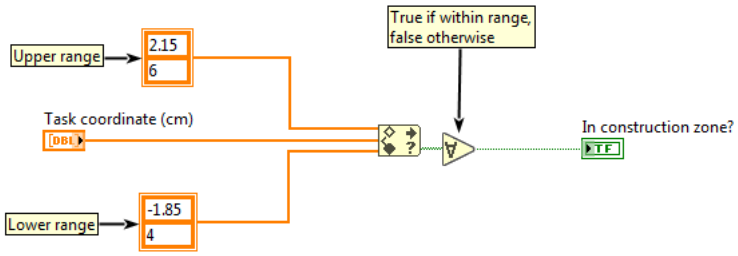


Figure 2.5: Detecting the construction zone in the Initialize Motion state

2. The next steps include emptying the target point queue except the final destination point, clearing the corresponding points from the Silk image, asking the manipulator to move out the construction zone, and then proceeding to the Initialize State Machine state to reset operations. These steps have already been completed for you. Run this VI and assign the system a final destination and way-points as shown in Figure 2.6. When it asks for new points after the construction zone routine, assign it the points shown in Figure 2.7. Note that point 1 in the latter already exists as it is the final destination. Does the line follower meet the required functionality?

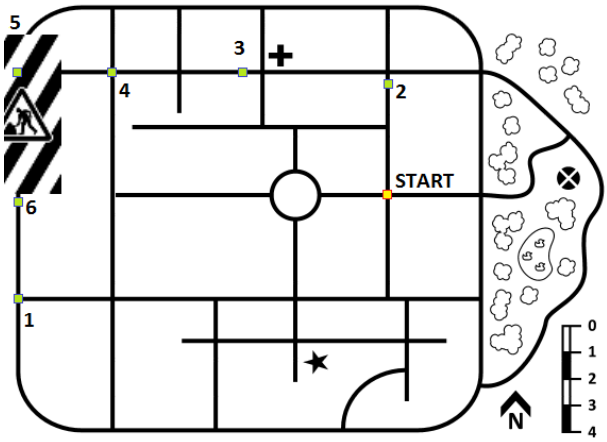


Figure 2.6: Way-points through the construction zone

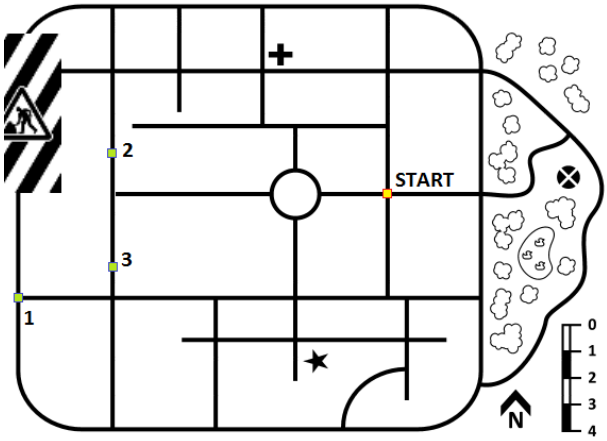


Figure 2.7: way-points around the construction zone

2.5 Thinking Further

1. In this lab, the construction zone was detected manually through kinematics. Can pattern matching be used for this purpose? Is there any disadvantage associated with this?
2. List some ways to improve the line-following algorithm.
3. List two similarities between this experiment and a real-life situation such as a self-driving car? List some assumptions in this experiment that do not necessarily hold in case of a self-driving car.

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