COMPUTING SCIENCE 399 – Topics in Artificial Intelligence, Machine Learning and Mobile Robotics
Assignment 4 – Reactive Robotics and PID controller

Due: April 1st 2014:

- **Demo - during the lab,**
- **Report and Code – at midnight**

Weight: 10% of the final mark
Type: Electronic
Work: Team Assignment
Objectives:

- Implement a P, PI,PD and PID controller
- Tune P,PI,PD,PID controller
- Implement a behavior control model

Submission information:

- Submit a ZIP file and a report file (.pdf) electronically using BlackBoard/eClass.
- The ZIP file has to contain your implementations and any data you either measured or generated. The implementation has to be well documented. In the ZIP file use a proper directory arrangement and write a readme.txt file about its structure.

NOTE:*Properly* acknowledge (add a note and/or hyperlink and/or comment) any help or resource you used.

1. Line follower

It is recommended to use a wide tape. The purpose is to get to the end of the line as quickly as possible.

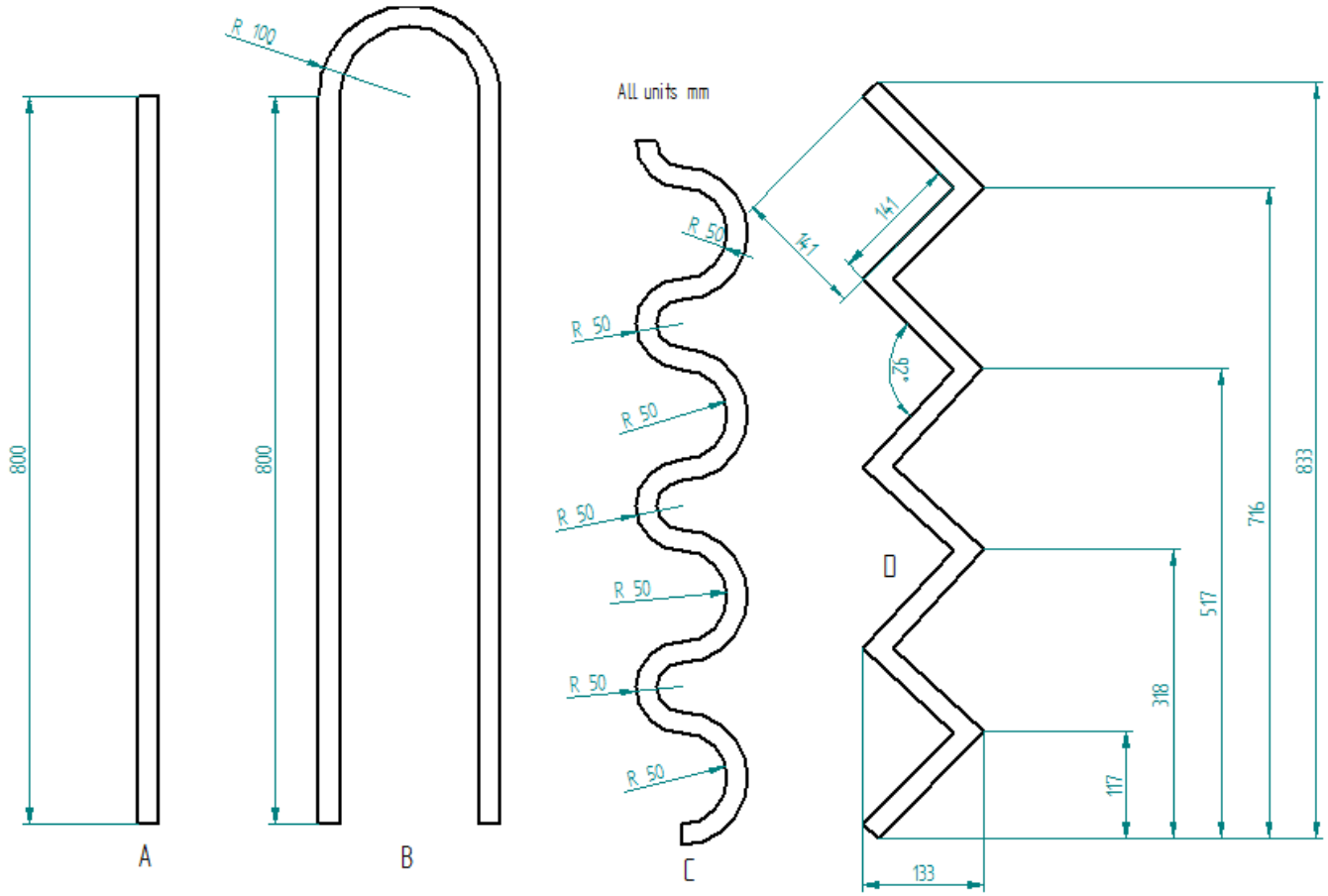

Figure 1. Line Follower

1.1. Add one or two light sensors to your robot.
1.2. Design a Bang-Bang, P, PD, PI, PID controller that makes your robot follow the line.
1.3. Report : **(REPORT)**

- Values of the *K* parameters after tuning (where applicable).
- Tuning method (where applicable).
- Time to completion.
- Describe the controller behavior.

Hint: http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html

1.4. Demo the best controller for each line configuration. **(DEMO)**

2. Click and go.

Design a system that allows your robot to follow a virtual line. Note: We are assuming that the camera lens and robot navigation plane are parallel and the lens distortion is negligible.

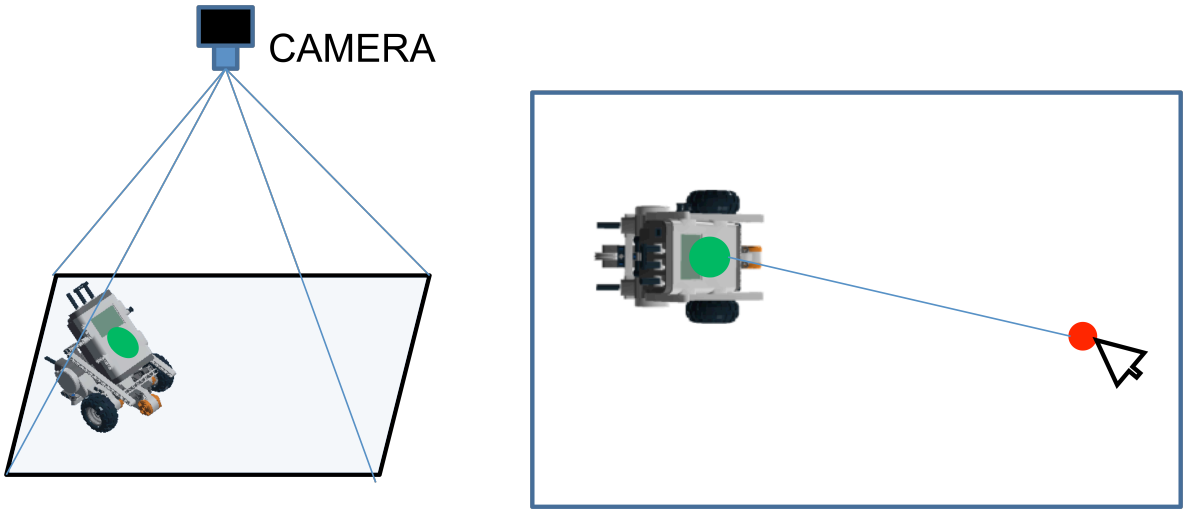

Figure 2. Robot Setup

2.1. Based on the Python Camshift tracker code, write a program to draw a line between the selected point and the tracker location.
2.2. Based on your experience from point 1 **choose** a P, PI, PD or PID controller that will make the robot follow the line. **Hint: What is the minimum distance between a point and a line?**
2.3. Give details of your controller: error, and *K* parameters. **(REPORT)**
2.4. Make the robot follow the virtual line. **(DEMO)**
2.5. Modify your implementation to allow multiple path points, see Figure 3. **(DEMO)**

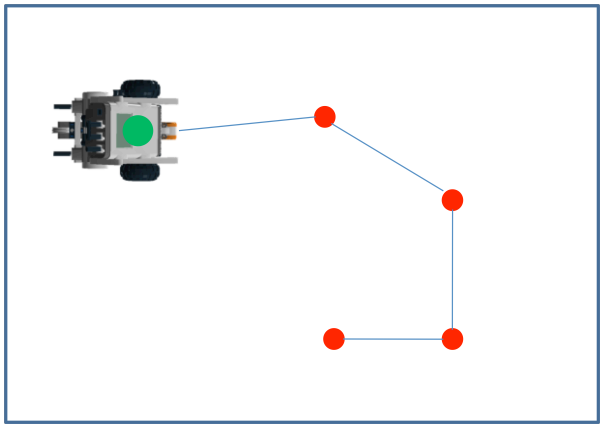

Figure 3. Multiple points

3. Get to goal and avoid obstacles.

Based on the robot world shown in Figure 4, design a **behavior based controller** that leads the robot to the goal as fast as possible. Attach at least 2 sensors to your robot.
Note: The NXT supports only 4 sensors and the tachometer built in the motors. Be wise locating and choosing your sensors. Obstacles are randomly located between the two black lines as shown in Figure 4. Consider all possible scenarios. For example: the robot may start with different orientations and at different positions relative to the start line.
Hint: http://www.lejos.org/nxt/nxj/tutorial/Behaviors/BehaviorProgramming.htm

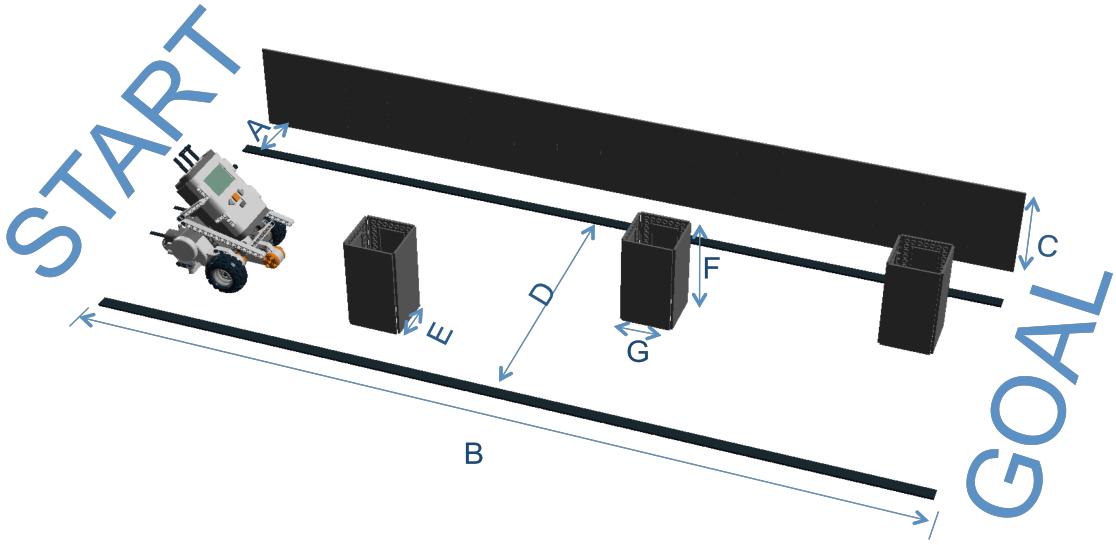

Figure 4. Robot World

A=10cm, B=170cm, C=12cm, D=60cm, E=10cm, F=12cm, G=10cm

3.1. Explain in detail your design, including the behaviors. **(REPORT)**
3.2. Make the robot get to goal. **(DEMO)**

Marking scheme:

Exercise	**Mark**
1	40
2	30
3	30
TOTAL	**100**

The mark for each point will be based on: report, demo and code.