

Robotics&Automation

ENGT4314

Lab Assignment - Reactive Driving

Lab Assignment 1

Individual Assignment

Due: week 3

Implement a robot driving system that implements a simple reactive control.

Required Skills: - C/C++ programming
- EyeSim simulator *or* Player/Stage/Gazebo simulator

Links: - <http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html> (Windows)
- <http://playerstage.sourceforge.net/> (Linux)

On the mobile robot simulation system:

- Implement a robot program that continuously reads three distance sensors (PSDs pointing to the left, front, and right) in the main loop and displays their distance values as three lines on the robot's LCD.
- If all three distances are above a threshold (sufficient free space), then the robot is to drive slowly forward.
- If there is insufficient free space in front of the robot, then the robot will slowly drive backwards at a fixed angle.
- If there is insufficient free space to the left (or right) of the robot, then the robot will slowly turn to the right (or left, respectively).
- In order to cope with sensor noise, for each sensor record the two previous readings together with the current one. For each sensor, use the median value of the three data readings for processing (this method is known as "mid-value-select).

Demonstrate your solution:

- By changing the robot's position in the settings menu

Robotics&Automation

ENGT4314

Lab Assignment - Remote Control

Lab Assignment 2

Individual Assignment

Due: week 4

Implement a remote control system that drives the robot from a PC. Use fltk to design a graphical user interface.

Required Skills:

- C/C++ programming
- RoBIOS robotics library
- fltk graphics user interface (fast light toolkit), <http://www.fltk.org/>

On EyeSim:

- Implement a program that reacts to pressing one of the four buttons by:
Driving forward, Rotating left, Rotating right, Stop.
Each driving motion is started by the corresponding button press and will continue until another button has been pressed.
Test the functionality of this step by pressing buttons on the robot's console.
- Extend the program so that it reads the file `command.txt` in an endless loop. The file will contain only a single letter:
F (drive forward), **B** (drive backward), **L**, (rotate left), **R** (rotate right), **S** (stop)
Perform the corresponding driving motion on the robot and continue until a different command is being read from the file.
Test the functionality of this step by editing the file `command.com` with a text editor.

On the PC:

- Implement a GUI using fltk with buttons for:
Driving forward, Driving backward, Rotate left, Rotate right, Stop.
Pressing of one of these buttons will write the corresponding driving command (F, B, L, R, S) to file `command.txt`.

Demonstrate your solution:

- By remote-controlling the robot via your PC's GUI.
-

BONUS POINT:

Implement an additional command **P** (read front PSD sensor), which takes a single sensor reading and reports this back for display on the PC's GUI.

Robotics&Automation

ENGT4314

Lab Assignment - Manipulator

Lab Assignment 3

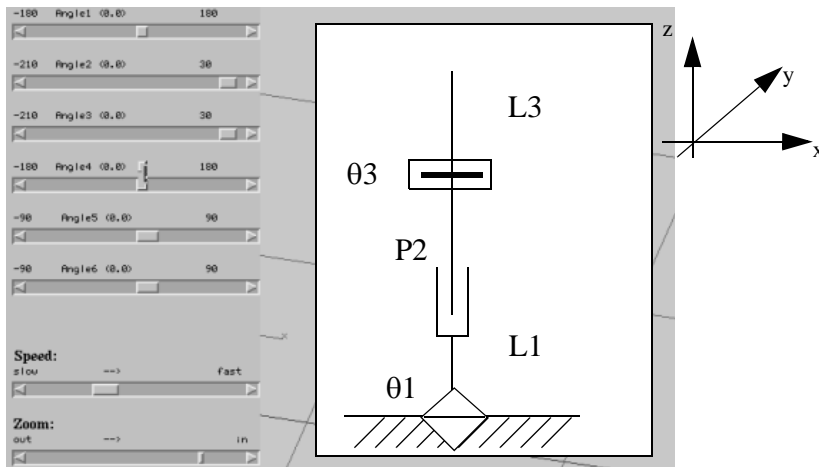
Individual Assignment (2 weeks)

Due: week 6

Implement a simple manipulator simulation system using 3D polygon graphics in C/C++.

Manipulator definition: (axes x, y are on ground plane; z axis is pointing up)

- 3 link manipulator with 2 rotary joints and 1 prismatic joint
- Base - **Rot(z)** - Link L1 - **Trans(z)** - **Rot(x)** - Link L3 -
- In the zero position all three links are aligned in a straight line, pointing up
- All links are box-shaped of identical size
- There is no end-effector.



- Use fltk (fast light toolkit), <http://www.fltk.org/>, to implement three slide rulers to set the two joint angles in the range [-90 .. +90] and the prismatic joint in range [0..100].
- For every slider change, recalculate the robot's new target position, then move the joints to the new position using a fixed joint speed.

Note: For getting started, use the fltk example “cube.cxx” for drawing a box (either shaded or as a wireframe) at a given 3D pose.

Robotics&Automation

ENGT4314

Lab Assignment - Navigation

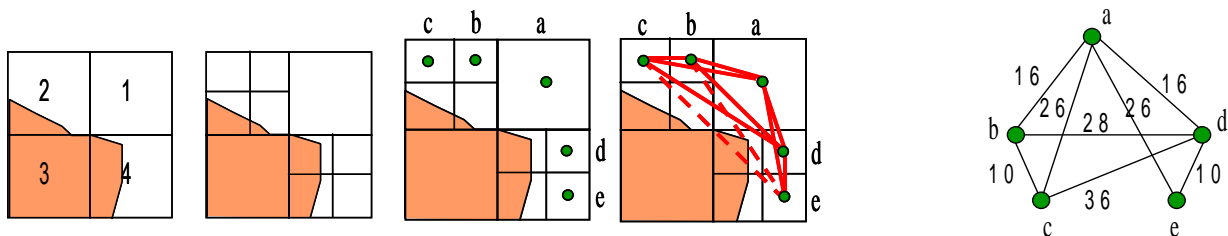
Lab Assignment 4 *Group Assignment (2 students, 2 weeks)* Due: week 8

For a given 2D environment in pixel format (specified as a PGM graphics file) of max. 128x128 pixels, implement the **Quadtree subdivision** algorithm to calculate a nodal distance graph for the free space in the environment.

Part 1:

For all remaining free nodes, calculate the coordinates of the respective square center (see example). Print all results in table form to a file, listing each node with its x,y-coordinates and its respective distances to all other nodes (see sample output below; use **fprintf** for file output).

Example: _



Note that distances can be calculated from the coordinates using the following formula:

$$dist(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

However, for each line it needs to be checked using the original environment data whether a collision free path exists (e.g. *not* for b-e, *not* for c-e).

Sample Output:

COORDINATES :

N	x	y
a	30	30
b	15	35
c	5	35
d	35	15
e	35	5

DISTANCES :

\	a	b	c	d	e
a	0.0	15.8	25.5	15.8	25.5
b		0.0	10.0	28.3	---
c			0.0	36.1	---
d				0.0	10.0
e					0.0

Part 2:

Implement the Dijkstra Algorithm as defined in the lecture notes.

- Use the free space coordinate data from part 1 as nodes.
- The start node is the node with the lowest coordinate difference x-y (i.e, the top left node), the goal node is the node with the highest coordinate difference (i.e. the bottom right node).
- Distances are symmetrical, so $\text{dist}(a, b) = \text{dist}(b, a)$.
- Drive the robot from its starting position to the graph start node in a straight line. (This assumes there is no obstacle between these two points. You may have to adjust the robot's starting position.)
- Use a robot simulator (e.g. EyeSim) to execute the driving.

Note:

- For driving from pose A to pose B, use the sequence:
 - turn on the spot (for getting the correct heading), followed by
 - drive straight (to get to the desired position)
- Write the output of the Dijkstra algorithm (node sequence of the shortest path) to the robot's screen.
- For debugging purposes (part 1), if you require it, print the quadtree subdivision for each recursive call as a pgm file (however, this may create be a very large number of files). Use a global counter to generate new filenames/
- For debugging purposes (part 2), write the driving command sequence to a file (or the robot's screen).

Robotics&Automation

ENGT4314

Lab Assignment - Grand Challenge

Lab Assignment 5 *Group Assignment (4 students, 4 weeks)* Due: week 12

For this project, you will implement a navigation task on the Pioneer AT robots using multiple sensors, similar to the DARPA Grand Challenge competition. Sensors available are:

- GPS sensor with USB-serial interface
- Digital compass with USB-serial interface
- Digital color camera (Logitech Sphere, can be rotated in two axes)

Full documentation on the Pioneer AT robots and the sensor systems will be made available to you.

As this is a group project, you need to discuss among your group member who will take over which part of the project. You will need to set internal deadlines and introduce internal reporting. Marking of this lab will be based on the group's lab solution as well as on lab reports of each individual group member.

Task: Implement a robot navigation system that solves the following requirements:

- The robot reads a given set of **GPS way points** at initialization time.
 - The robot will follow the GPS way points in the given order.
up to a certain proximity (e.g. 10m), then use image processing for fine path planning
 - After clearing an obstacle, the robot continues driving to the next way point.
 - Each way point is marked with a box of distinctive color (yellow or red).
 - Use **image processing** to identify each box by color.
For every detected object, record the x,y-position of the object center together with the robot's position and orientation in an (fixed size) array data structure.
 - Considering the previous and next way point, pass the box on the **outside** track.
Always **avoid collision** with a box.
 - During each run, the robot will maintain a data structure that plots its driving path (at least one entry per second).
 - Upon completion of the task, the robot will write this data structure to a text file, which can be then converted to a graphics plot (e.g. using Excel). This **plot** has to be included in the **lab report**.
-

Bonus points:

- Avoid stationary and moving obstacles, such as buildings, walls, trees, and people with the help of image processing (2 points).
- Use differential GPS for improving navigation accuracy. An additional GPS module is available, but you also need an additional laptop with WIFI to implement this (3 points).

Lab report: A lab report has to be submitted by each individual student, including the following information:

- Split of assignment tasks between group members
- Time spent on own assignment task
- Documentation, describing design choices and solution path
- Well-documented source code
- Full set of measurements and graphs
- CD-ROM (or other media) with drive video and recorded path data (only one per group)