Stefan B. Williams
March, 2016

MTRX5700: Experimental Robotics

Assignment 2

Note: This assignment contributes 10% towards your final mark. This assignment is due on Wednesday, April 20th during **Week 7** before 5pm. Submit your report via the eLearning TurnItIn submission on the course website. Late assignments will not be marked unless accompanied by a valid special consideration form. Plagiarism will be dealt with in accordance with the University of Sydney plagiarism policy. The objective of this assignment is to explore **perception** using with a focus on laser and vision sensing. You may use the Matlab image processing toolbox, python or the OpenCV libraries developed at Intel. The choice of tools will depend on your preference and level of proficiency in Matlab, python and/or C/C++. This assignment should take an average student **12-15 hours** to complete with a passing grade.
Total Marks: 100

You may work in pairs on this assignment. The front page of your report should include:

- Your name(s) and SID
- Your tutorial session

Laser Data Processing and Visualisation

Download and display indoor and outdoor laser data

a. The code and datasets can be downloaded from the assignment website.

b. ACFR indoor dataset: run "laserShowAcfr.m" in Matlab to see the ACFR indoor data set. Figure 1 shows a scan example.

c. CMU Navlab urban dataset: run "laserShowNavlab.m" in Matlab to see the Navlab urban data set. Figure 2 shows a scan example.

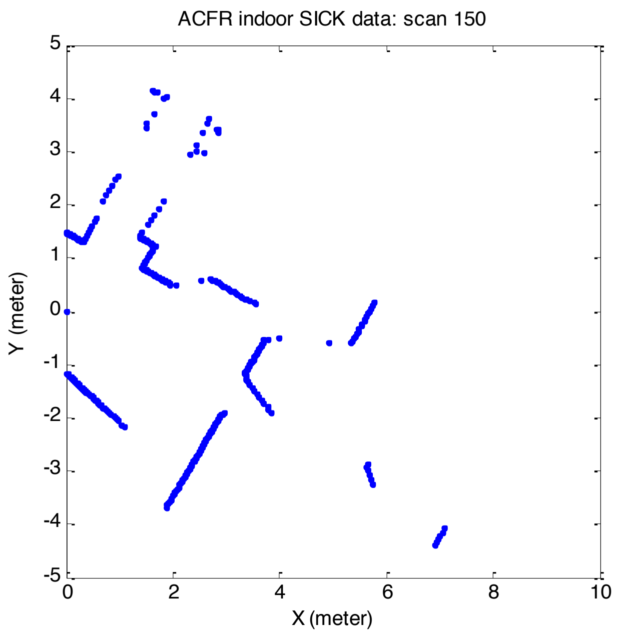


Figure 1: A scan example of the ACFR indoor dataset

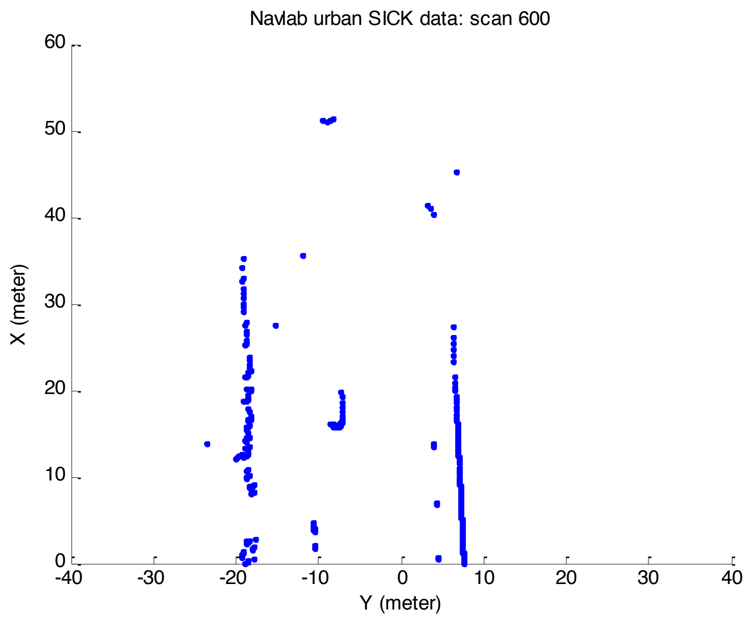


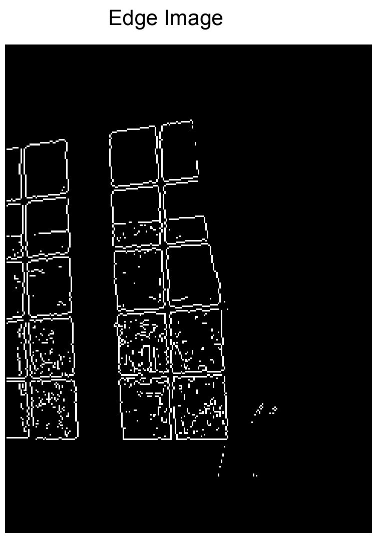
Figure 2: A scan example of the Navlab urban dataset

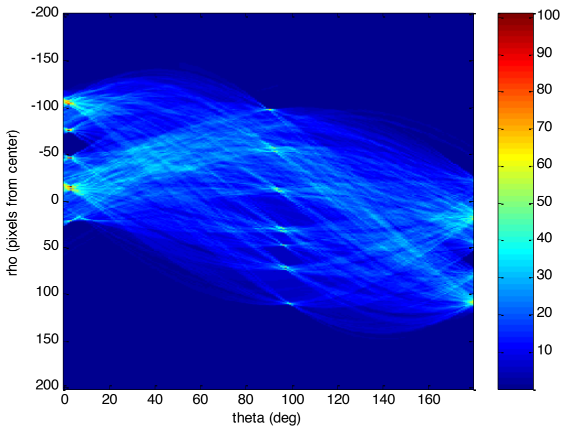
Note: The coordinate systems used in "laserShowAcfr.m" and "laserShowNavlab.m" are different.

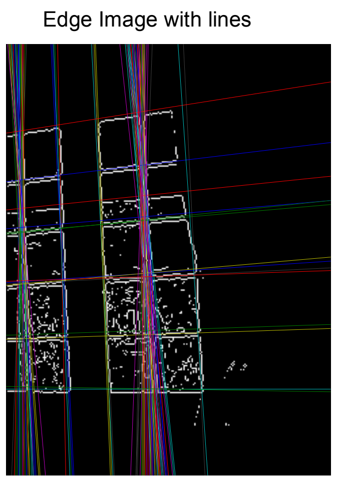
1. Line estimation and segmentation [20 Marks]
   a. Implement a function to compute the perpendicular distance of a point from a line segment.
   b. Implement a function to compute the least square minimization for fitting a line to a list of scan points.
   c. Implement the line splitting/segmentation algorithm described in the lecture.

d. Add your function to one of the above scripts and display the resulting lines as part of the laserShow script. Include a few sample plots in your write up showing how your system performs.
e. Finally, corners can be characterised by the intersection of two lines that are nearly perpendicular. Select two scans that contain corners (you may select them manually by stepping through the scans and identifying scans with a few corners) and find adjacent lines that are approximately perpendicular. (HINT: the dot product of perpendicular vectors describing two lines is approximately zero)

2. The Hough Transform [15 Marks]: We have provided you with functions for implementing the Hough transform algorithm. Run “Houghvisualimage.m” to get familiar with the implementation details. The following figures show an example of finding lines in visual images using the Hough transform.
   a. Select an image of your choosing (hopefully one with lines in it!), apply an edge extractor (Canny or Sobel for example) to identify lines and produce a gradient image similar to the ones shown below. Use this image with the Hough transform algorithm and include the resulting image in your write up.

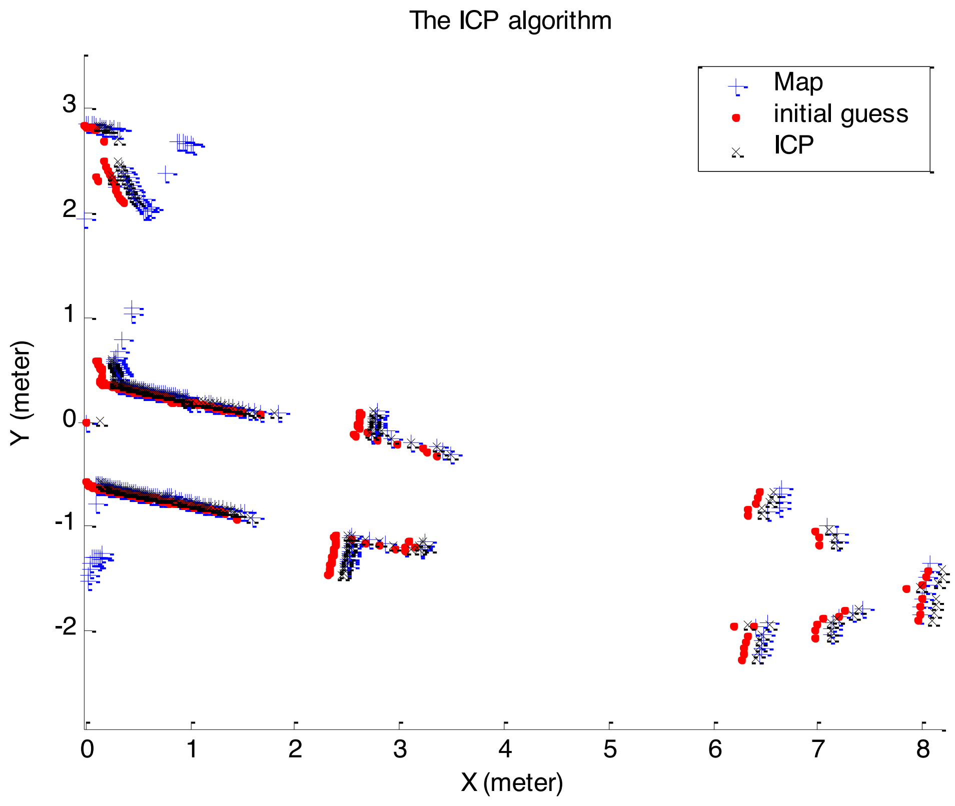






b. Explore the use of the Hough algorithm to find lines in range images (laser scans) and test your code with the ACFR indoor data and the Navlab urban data. You shouldn't need to convert the laser scans into an image for processing but input the raw points into an appropriate Hough parameter space.

3. The Iterated Closed Point (ICP) algorithm [15 Marks]: We have provided you with functions for implementing the ICP algorithm. Run "showICP.m" to get familiar with the implementation details. The following figure shows an example.
    a. Given the pair of laser scans and initial estimates of their relative pose, use the ICP functions to refine the estimate, report the resulting relative poses between the scans and discuss the problems of the ICP algorithm you found.
    b. Now integrate the ICP algorithm into the laserShowACFR example and concatenate the pose increments to estimate how the vehicle has moved during the run. Does the resulting estimate seem reasonable? How can you validate the quality of the resulting path estimate? Discuss any problems that might have arisen with the algorithm.



Vision Data Processing

4. The following images show various views of a calibration board used for identifying the intrinsic parameters of a camera. Download the Matlab calibration toolbox from the assignment website and use it to identify the intrinsic parameters of the camera used for capturing these images. Apply the resulting calibration to the images to verify the calibration using the tool. [10 Marks]

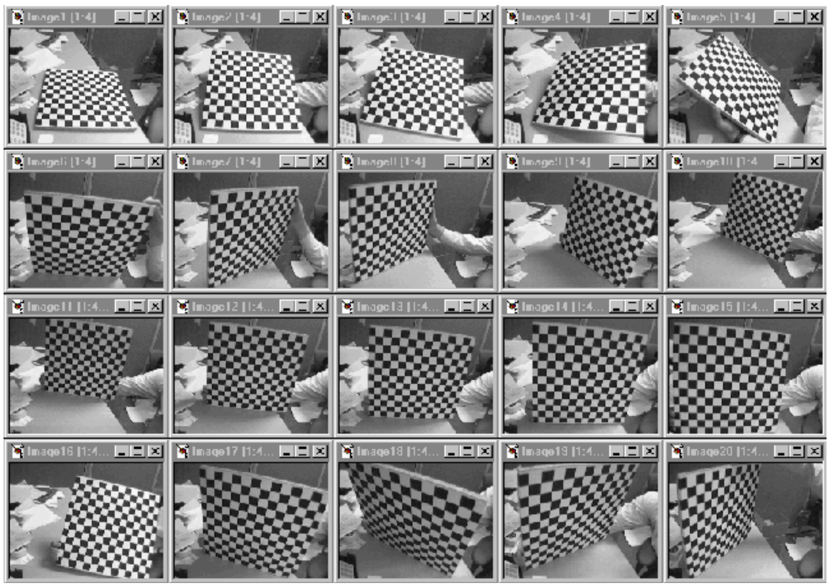


Figure: Camera calibration example

5. We have provided you with a number of sample images to be processed. Use the Matlab image processing toolbox or OpenCV to extract the following features [15 Marks]
    a. Sobel and Canny edge detectors. (Hint: help edge)
    b. Harris Corners
    c. Sift Corners (available at http://www.cs.ubc.ca/~lowe/keypoints/)

Figure: sample images from a two lane lined street

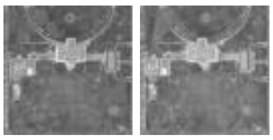
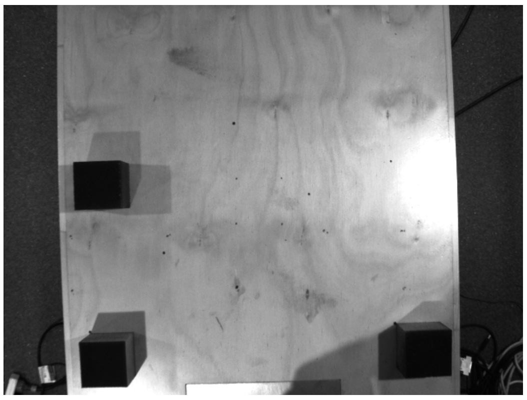


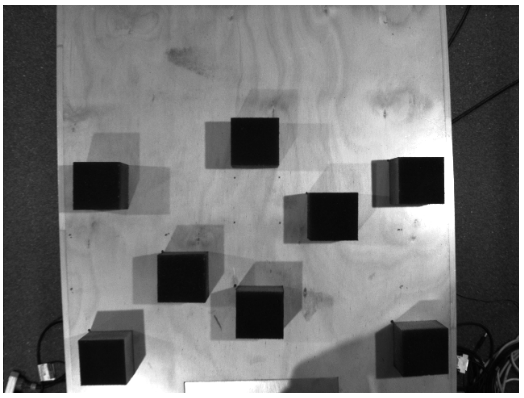
Figure: Stereo view of the White house

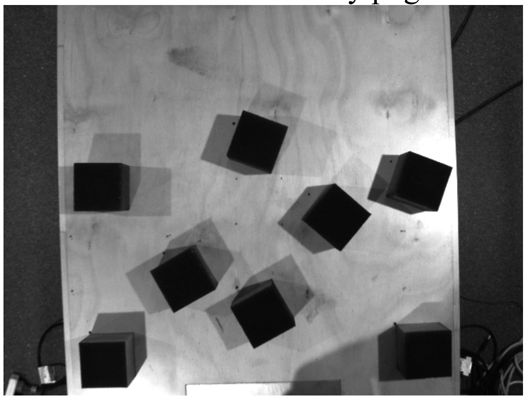
    d. Using the corners identified from similar pairs of images, implement a feature matching algorithm that allows the correspondence between features to be established.

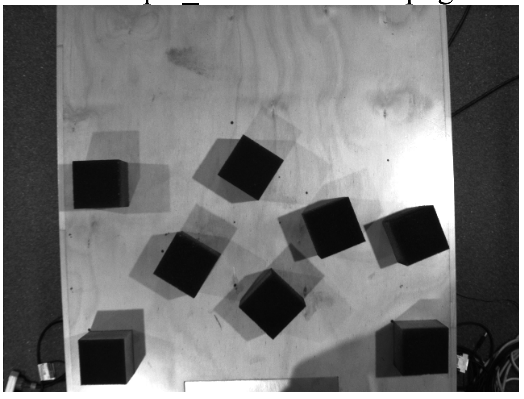
6. In assignment 1, we set up a task in which the robot arm was required to pick up a number of blocks from known positions in its workspace. Now we would like you to work on providing feedback to the robot using a vision system in order to allow it to retrieve blocks from arbitrary locations in the workspace. We will place 5 blocks at random in the workspace and you should use feedback from the camera to calculate where the blocks are and to retrieve them, piling them up in the same location as in lab 1 (i.e. at position (0,490)). There will be fiducial blocks in the camera workspace at known locations that you can use to provide a calibration between the image and robot workspaces. Demonstrate your working solution to your tutor [25 Marks].

   The goal of this exercise is to estimate the positions and orientations of five blocks using visual input, using the resulting estimates to drive the robot arm to construct a tower. To help get you started, you are being provided with the following example images, shown below.


ThreeFiducialsOnly.png


Example_KnownBlocks1.png


Example_KnownBlocks2.png


Example_UnknownBlocks1.png

Sample block images

   In all images three fiducial blocks will be present at the same, known locations. These are seen in ThreeFiducialsOnly.png. The fiducial positions, in mm and in arm coordinates, are:

   -270, 220
   -270, 523
   280, 220

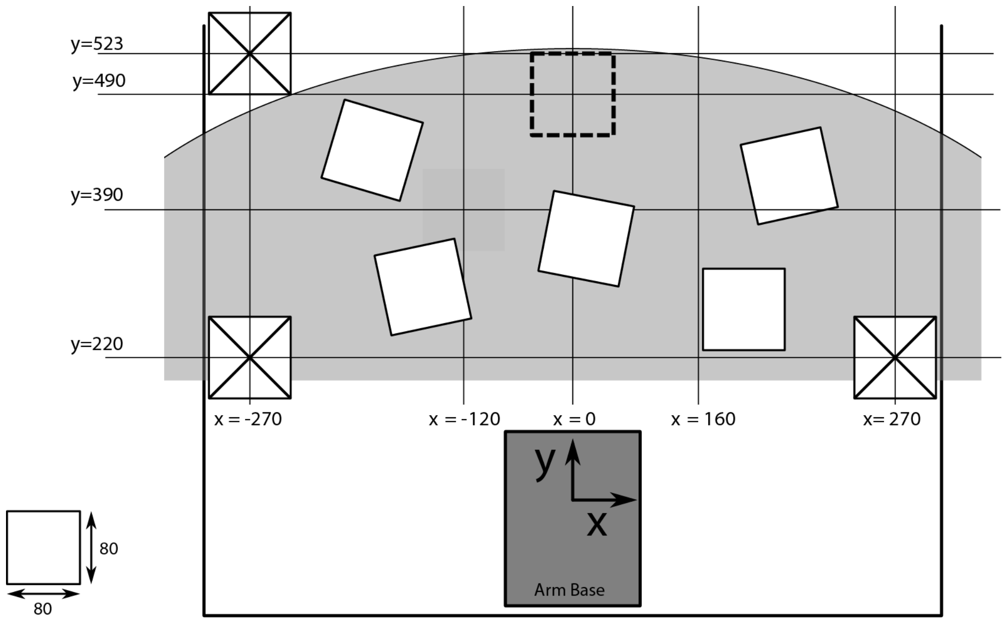
For the demo, five additional blocks will be placed on the board, leaving space at 0,490 for a tower to be built. Example_KnownBlocks1.png and Example_KnownBlocks2.png are being provided to assist you in developing a solution. Both images have blocks at the following known locations, again in mm and in arm coordinates:

0,300
-130,360
0,590
280,515
130,460

In Example_KnownBlocks1.png, all blocks are at 0 degrees, while in Example_KnownBlocks.png, the two angled blocks closest to the arm are at 30 degrees, the two furthest from the arm are at 70 degrees, and the remaining block is at 50 degrees.

Example_UnknownBlocks1.png is provided for the report writeup: please include in your report estimates of the positions and orientations of all blocks (including fiducials) in all three example images. Ideally, you will include both a list of coordinates and angles, and a graphical depiction of your solution.

We have provided you with example code in MATLAB illustrating how you can drive the robot arm directly from the control computer. This will allow you to close the loop between the visual block detection and the commands you send to the robot. The commands are in the form of text strings sent over a network connection to the arm computer. The code also includes a new function to set the tool angle. Keep in mind that its parameter is forced to the range 0 to 180 degrees. You can use the MATLAB example to help structure your solution. If you wish to use C/C++, Python or another language, the MATLAB code should give you an idea of how to interface to the robot.



Goal : Using blocks within the workspace (white), construct a tower of 5 blocks at the dashed location
The three fiducial blocks (marked with an X) are at known locations and can be used to calibrate the camera images.

Sample block configuration in robot workspace