CSE 4422.03
Lab Assignment II.

Due Date: Nov. 7, 9:30am
NOTE: the proposal is due on the same day.

1. Adaptive Thresholding

Write a program that takes an image and fits, in a least square sense, a plane on the grayscale of the image. In other words

$$I^t[i, j] = \alpha_0 + \alpha_{10} i + \alpha_{01} j$$

where the parameters α are estimated by minimizing

$$\sum_{i,j} (\alpha_0 + \alpha_{10} i + \alpha_{01} j - I[i, j])^2.$$

Use this fitted image I^t to threshold a continuous stream of images and display the results. To show the advantage of the method your program should be able to switch to a simpler thresholding where the threshold is just the average image intensity. For the demo choose your images so that the advantage of the method is shown.

2. Circle Detection and Fitting

Implement a version of RANSAC that detects and fits circles. it does it in the following way:

Take a picture of an image that contains at least 3 circles (like an image of a few balls in front of a contrasting background) and hand select one or two dozen points on the edges of the circles.

(1) Your program randomly selects triplets of points and finds the circle that passes through the points.

(2) Your program counts the number of points that are within W [5] pixels of the circle.

Repeat the procedure 200 times to reduce the probability of failure to find a circle. Select the circle that has the most points within W pixels. Remove these points and repeat to find the next circle. Prepare a demo for it like the previous lab overlaying the circle on the image. You will demo your program by setting up and taking a picture yourself, letting the TA hand select the points, and then your program detects the first three circles and displays the result.

3. Hand-Eye Calibration.

Mount the Point Grey camera on a tripod near the robot and take a picture of the workspace of the robot. Then have the robot place a small object of contrasting albedo on the workspace and detect it with the camera by comparing the image with the one without it. Do it several times, each time displaying the result with the detected point superimposed on the original image. Your program should produce a two matrices, both $2 \times N$ each column of which is the $2 - D$ point on the image and the bench plane.