ASSIGNMENT 2

EECS 4422

NOVEMBER 7 2016

Michael Williams: 211087798, mw1992

Andrew Lau: 212905253, pellost

QUESTION 1

For the first question in our assignment, we were required to write a program that would capture an image and fit, in a least square sense, a plane on the grayscale of the image. This was summarized as the following equation:

It[i, j] = α 0 + α 10i + α 01 j

To fit an image in this sense, determining the value of α was required. This could be done by minimizing the following summation:

Once we solved this, we were to use the fitted image It to threshold a continuous stream of images and display the results. Furthermore, so that the advantages and changes could be seen, we were to also have a comparable threshold of simply the average image intensity.

To implement this program, it was first necessary to compute the α values. This involved creating a set of matrices and then using these matrices to solve for α 0, α 10 and α 01. The matrices were derived by taking the partial derivatives in respect to these 3 unknown parameters. Once the matrix equation was formed the rest of the program was simple. The matrix equation was derived as such:

We started by deriving the partial derivations as described above. The results are as follows.

Given S =

And letting

Height =height of image

Width = width of image

= 2

= a(height)(width) + b(width) + c(height) - = 0

= 2

= a(width) + b(width)+ c - = 0

= 2

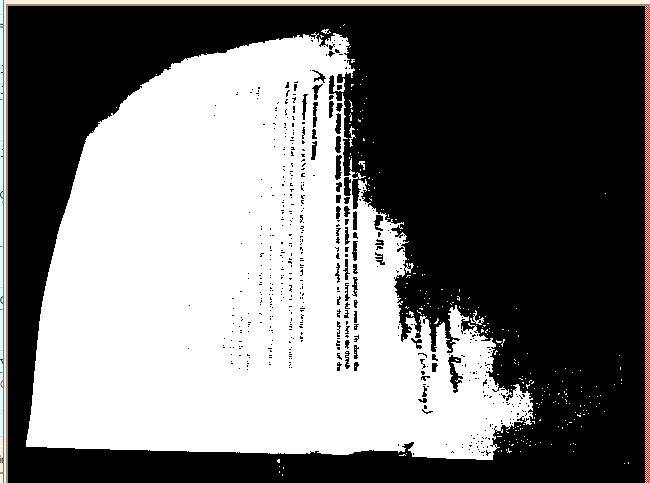
= a(height) + b + c(height)- = 0

Afterwards, we create the matrix equation:

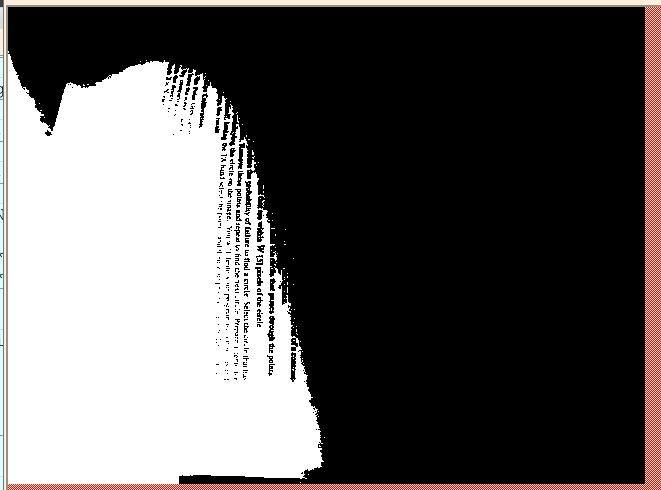
=

Now, implementation becomes rather simple. Once the camera has been set up and the live stream turned on, the program will take each image at a time and create the matrix equation above. It then solves for a, b and c values by taking the Inverse of the first matrix above and multiplying it by the third matrix above (the resultant). Then, with the alphas now solved in the equation It[i, j] = α 0 + α 10i + α 01 j , we simply apply the threshold to the image or for comparison apply the threshold corresponding to the image’s average image intensity. For applying the threshold to the original image, we simply use the comparison operation “image > threshold” across all pixels of the image and display it.

Below is an example of the effects of an averaging threshold versus an adaptive threshold as displayed by our program’s results.



The above is an image of a paper containing typing on it. The image has had a light shun on it. This image was taken with the average threshold.



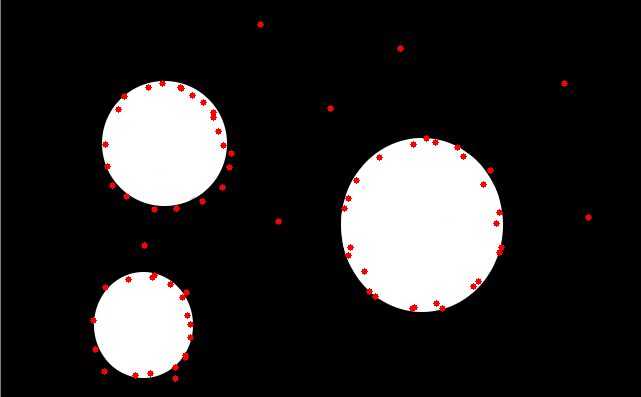
Above is the same image, only this time with an adaptive threshold applied to it. Immediately, the difference is very noticeable.

QUESTION 2

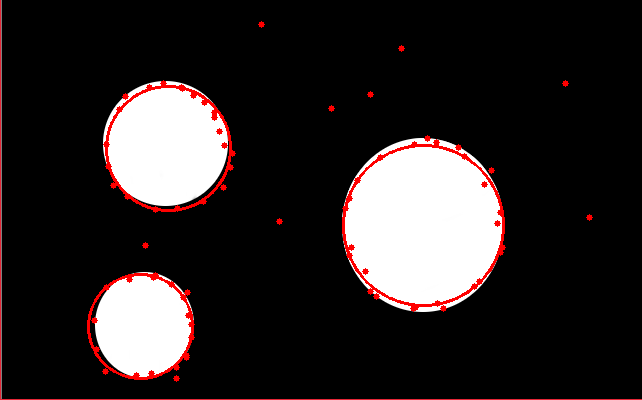
For this part of the assignment, we were to implement a version of RANSAC to detect and fit circles. It works as such: First, the program captures an image. The image must contain at least 3 circles in it. Then the user selects numerous points on the edges of the circle. This is done simply by clicking on the coordinates that will be denoted as circle edges. From this point the program creates and computes circles that pass through the selected points and chooses the best fitting circles. The result of the program is the traced out circles, approximately on the edges of the circles in the image taken.

To implement this program, first and foremost was to obviously capture an image. It is important that the image not only contain 3 circles in it, but that the images have a contrasting background. We then code the program to accept numerous selected edges, chosen by the user with the mouse. It is assumed that these edges are the edges of the circles in the images. The order does not matter. The better the contrast between the circles and the background image, and the more approximate the edges chosen are to the actual edges of the circles, the better the results one can expect to obtain. Once these edges have been selected, the program then takes 3 random edges. The program then uses these 3 edges to create a circle and computes how many of the selected edges fall within 5 pixels of that circle. This process is repeated 200 times to reduce the probability of failing to ﬁnd a circle. After the 200 cycles, we take and store the circle in which the most edges pass through. We then remove all of these points and take another 3 points, assuming they are available, before finding the next circle and repeating the entirety of the process. After the program is finished creating all the circles and assessing all the edges it then draws the circles over the image so that the user may see the circle generated and confirm that it indeed forms approximately over the edges of the circles, both in the image and selected.

Below is an example of the program running as well as its results:



The above shows the image, containing 3 circles, as well as the user chosen edges, marked by the red dots.



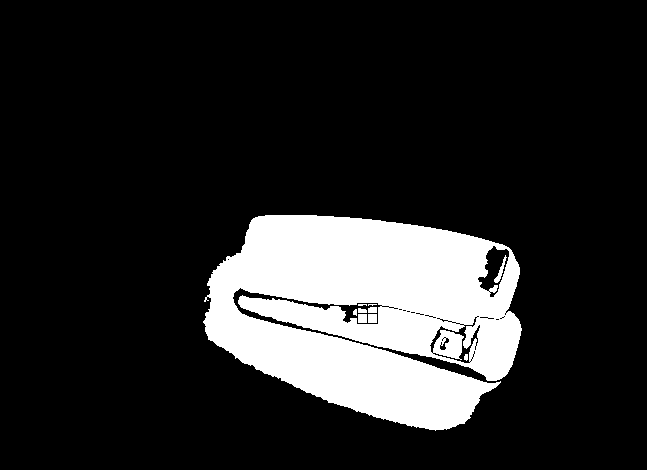
The above shows the results of the RANSAC implementation. As you can see, the best circles are formed in relation to the edges selected and the edges of the circle, assuming the edges were chosen according to the specifications described above.

QUESTION 3

This last question of the assignment involves the use of both the point grey camera and the robotic arm. In this question, the user uses the point grey camera to capture an image of the robot’s workspace. Afterwards, the robot places an object of contrasting albedo onto the workspace, and another image is captured. The goal is to detect the added object by comparing the 2 images taken. This is done several times to ensure accuracy and correctness and each time the result is displayed with the detected point superimposed onto the original image

The robot movement created for the first assignment was used to implement the arm movement in this program. First an image was taken of the field without any objects. Another image was taken after the robot moved the object to a known x and y coordinate on the world coordinate system. A Gaussian filter was applied to both images to reduce the effect of noise on them. Then the differences between the two images were computed and a mask was found. The x and y coordinates of the object on the image coordinate system can be computed by finding the average of the x and y points of the mask. Both coordinates were then stored in a matrix.

Below is an example of the program at work.



This is an image of the mask, the square marks the center of the object



Image of the image taken after each move, the square marks the center of the objec