FINDING CIRCLES IN IMAGES

ABSTRACT

PURPOSE: Compare least-squares circle fit to organic vs. non-organic data both qualitatively and quantitatively

METHODS: Given an edge dataset, least-squares approximations and rmse are calculated using circlefit function. Circlefit returns a center point, radius and the rmse which is all used for plotting the approximation against the original image, except rmse, using the *circleplot.m* function provided.

RESULTS: Non-organic objects produce minimal error and a practically perfect fit whereas organic objects produce larger error with greater variance

CONCLUSIONS: The implemented algorithm adequately produces approximations to each object. There is a loss of data in facial recognition for organic objects.

INTRODUCTION

The objective was to assess the quality of fit to the edge data of circles in edge images. Quantitatively, this is accomplished through a least-squares circle solution with calculations for RMSE(quality of fit), a RMSE of 0 would indicate a 'perfect' fit. Qualitatively, we can assess the quality of fit by the use of the function *circleplot.m* provided by the instructor which maps each approximation to its data image. Each object in the image(s) being mapped to has an individual data matrix (edge data) provided in the data directory of the folder provided for the assignment. Each object must be processed through a least-squares circle solution which returns the center point, radius and error of the current fit, we then provide *circleplot.m* with the center point, radius and a color value where it maps the approximation to the image. We must fill in the circlefit skeleton function to where it returns the correct values needed to call *circleplot.m*.

METHODS

I began with filling out the skeleton of the circlefit function, it's the heart of the operation. Circlefit calculates all the values required to quantitatively and qualitatively assess the quality of fit. Circlefit has one parameter, it must be passed a tall-thin matrix of edge data, not short-wide. Each column is represented by a vector x_j, m represents rows (number of data points) and n represents columns. We can now start the least-square circle solution provided by the instructor. First we must create a matrix, A and vector, b such that Ax=b is overdetermined and can be solved using QR decomposition. The matrix A can be simplified as $\left[-2x_j \;\; 1\right]$, an m-by-3 matrix, with the last column being all ones. The matrix b can be simplified as $\left[-x_j x_j{}^T\right]$. Vector x will consist of two points (the center), g, and a scalar, σ to deduce radius when solved. To create matrix A, an initial matrix of size m-by-n+1 was instantiated with all ones, vector b was instantiated similarly with an m-by-1 vector filled with all ones. A double for loop is used to fill each matrix/vector with its desired values as stated above. The result of the for

loop is A = $\begin{bmatrix} -2x_j & 1 \end{bmatrix}$ and b = $\begin{bmatrix} -x_j x_j{}^T \end{bmatrix}$, we can now go to solve the overdetermined linear equation using QR decomposition. The built-in qr function in matlab was used with an economy setting, returning a matrix, Q of coefficients and an upper-triangular matrix, R of scalar weights. Solving for x can now be deduced to $Rx = Q^T b$, we can create a new data vector, $y = Q^T b$ and use back-substitution to solve for x. A for loop was used to implement back-substitution with the result being a new vector, x, what we have been solving for. Now that we have x, we have the center point and a scalar, σ, to which we can solve for radius. The radius of the approximation can be solved as $\sqrt{g \cdot g - \sigma}$, the square root of the dot product of the center points minus scalar, σ. These are all values in x, the first two rows represent the center point, the last row represents σ. We now have enough information to plot the approximations using *circleplot.m*, however rmse still needs to be calculated, our quantitative assessment of fit. To do so, first we will transpose our original edge data matrix, so that it is short-wide and create a new variable, rmseSum to sum the errors over each data point. We calculate the residual error for each each data point (column) using a for loop. Each iteration calculates $(\|x_j - g\| - p)^2$, the residual error of the current data point and adds it to rmseSum. Upon loop completion, we can calculate the root mean squared error as simple as $\sqrt{\|rmseSum\|/m}$. Now we can plot all our approximations in the main function. First we must load each object from its directory and create a new figure, next we can use built-in functions *imread* and *imshow* to display the image, and lastly we must calculate *circlefit* for each object and use *circleplot.m* to plot on the desired image. This is done successively for both images, disks and pill.

RESULTS

Table 1: All values calculated in MATLAB code

Object Number	**Center point (x,y)**	**Radius**	**RMSE**
1	(100.0,100.0)	52.0490	0.1216
2	(75.0,250.0)	37.1176	0.0788
3	(249.9990,400.0077)	62.0469	0.1119
4	(300.0,120.0)	42.0877	0.0785
5	(450.0,240.0)	52.0490	0.1216
6	(329.9861,370.0037)	56.9895	0.1200
7	(119.8694,129.6525)	49.0829	3.8099
8	(319.5886, 83.0099)	38.6597	3.3738

9	(273.3834, 208.1167)	76.1723	11.0829
10	(434.3979, 167.9061)	35.7537	7.6020

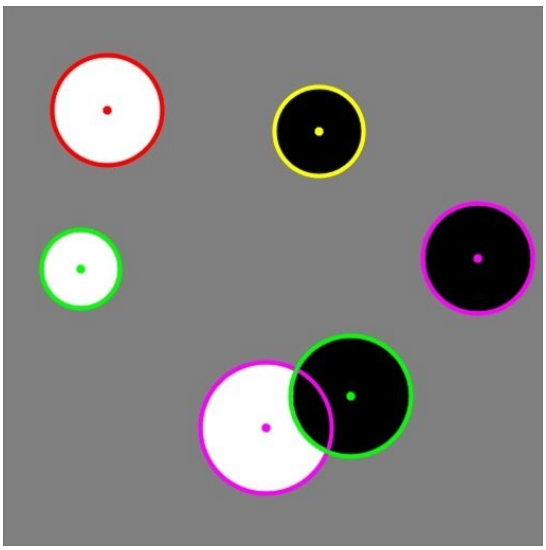

Figure 1: circlefit approximation using circleplot on disksimage

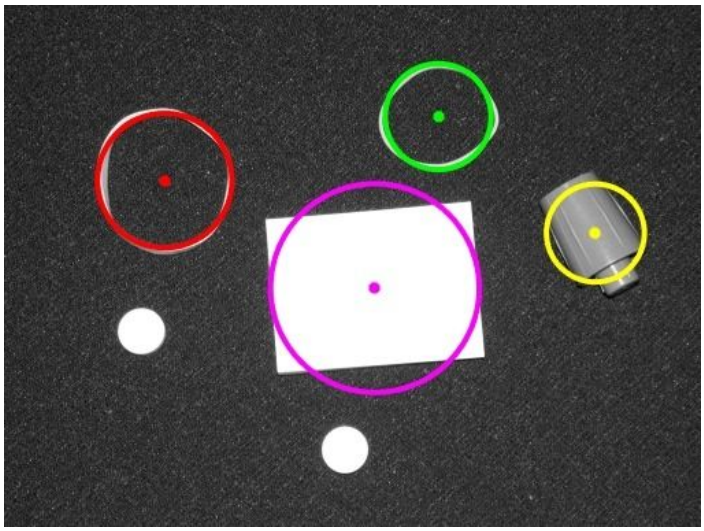

Figure 2: circlefit approximation using circleplot on pillsetgray

DISCUSSION

From the results, particularly of *figure 1* and *figure 2*, we can deduce that the circlefit function correctly calculates circle approximations given a tall-thin edge dataset (object). In *figure 1* we can see a 'perfect fit' qualitatively, the disks in the image are not organic such as those in *figure 2*, it takes a circle to match a circle. Furthermore we can see how organic vs. non-organic objects effects RMSE. From *table 1* we can see that the first 6 objects (non-organic) have a very minute RMSE, none of which are greater than 0.1216, implying a practically perfect fit for each of those objects. This is due to the fact that they are computer generated near perfect circles which explains why there is error at all. Meanwhile the RMSE's of the organic objects are much greater, ranging from 3.3708 to 11.0829. From *figure 2* we can see that none of the objects are near perfect circles, however the rubber bands (red and green) are the closest to it and produce errors of 3.8099 and 3.3738. This helps us grasp the difficulty of facial recognition in video surveillance, everyone has an organic face, so a perfect fit will never occur when detecting with circles. We can see the difficulty and perhaps shortcomings of detecting with circles by looking at the yellow fit in *figure 2* which has an RMSE of 7.6020. The marker cap has the most human face like shape, taller than it is wide, and we see that it is not enveloped by the approximation, if it were to be a humans face there would be a loss of data in the region of interest. In this case, the neck and the top of the head would be cut off which could contain vital information for facial recognition. We can see this same occurrence with the magenta fit who has

the greatest RMSE of 11.0829. Overall, the matlab code is able to approximate a fit to each object with minimal RMSE for disksimage and pillsetgray within reason of organic vs. non-organic objects.

REFERENCES

[1] CISC 271, Winter 2019, Assignment #3 [Internet]. queensu. Available from: https://onq.queensu.ca/content/enforced/260955-CISC271/Homework/A3.pdf

[2] QR Decomposition [Internet]. queensu. Available from: https://onq.queensu.ca/content/enforced/260955-CISC271/Notes/Class24.pdf

[3] QR and Regression [Internet]. queensu. Available from: https://onq.queensu.ca/content/enforced/260955-CISC271/Notes/Class25.pdf