You are tasked with writing the software for a parts inspection system. The system wants to determine if the size of the part is within a specified tolerance. The input to this system is a color image of a circular part. Your task is to locate and fit a circle with center \((c_x, c_y)\) and radius \(r\) to the part.

**Setup**

1. Download the images `disc1.jpg`, `disc2.jpg`, ... `disc5.jpg` from the course website.

2. To begin, your system should load a color image, convert the image to grayscale (see `rgb2gray`), and extract the salient edges in the image (see `edge`). The Matlab function `edge` will return a binary image with a value of 1 at a pixel location corresponding to a salient edge, and 0 otherwise (you can visualize this binary image using `imagesc`.)

3. Extract the \(x\)- and \(y\)-coordinates of all pixels with an edge value equal to 1 (don’t use a for-loop for this, instead, see `find`). It is from these \((x, y)\) coordinates that we will find and fit a circle using the expectation/maximization (EM) algorithm.

**E-step: Expectation**

4. Some of the pixel locations that you extract above will be part of the circular part and others will not. Use the expectation/maximization (EM) algorithm to simultaneously locate the boundary of the circular part and fit a circle to the part. In this case, the coordinates of each edge pixel \((x, y)\) extracted above corresponds to one of two models: (1) a circle with center \((c_x, c_y)\) and radius \(r\); or (2) an “outlier” model.

5. Begin your EM iterations by initializing \((c_x, c_y)\) as the image center \((233, 233)\), \(r = 190\) pixels, and \(\sigma = 8\).

6. We define the residual error between a pixel location \((x, y)\) and model 1 (the circle) as the shortest distance between a point and a circle:

\[
d = \left| \sqrt{(x - c_x)^2 + (y - c_y)^2} - r \right|
\]

7. Compute the probability that each edge pixel location \((x, y)\) belongs to model 1 by assuming that the distances \(d\) are governed by a Gaussian distribution (with standard deviation \(\sigma\)). Assume that the probability of belonging to model 2 is governed by a uniform distribution and fixed at 1/10. Use Bayes’ theorem to compute the probability that each edge pixel location belongs to model 1.
M-step: Maximization

8. A circle can be modeled by the equation

\[ a\vec{x}^T \vec{x} + \vec{b}^T \vec{x} + c = 0, \]

where \( a \in \mathbb{R} \ (a \neq 0) \), \( \vec{b} \in \mathbb{R}^2 \), and \( c \in \mathbb{R} \) define the circle, and \( \vec{x} \in \mathbb{R}^2 \) are the x/y-coordinates of points on a circle. As we previously saw, estimating a circle center and radius does not lend itself to a linear estimation. We can, however, approximate the fitting a series of points to a circle as a linear system of equations \( B\vec{u} = \vec{0} \), where \( \vec{u} = (a, b_1, b_2, c)^T \) and

\[
B = \begin{pmatrix}
  x_1^2 + y_1^2 & x_1 & y_1 & 1 \\
  \vdots & \vdots & \vdots & \vdots \\
  x_m^2 + y_m^2 & x_m & y_m & 1
\end{pmatrix}.
\]

With \( m > 4 \), we have an over-constrained linear system.

Use a weighted total least squares estimator to solve for \( \vec{u} \), where the weights correspond to the probability (computed in the E-step) that an edge pixel belongs to model 1.

9. We now need to convert the parameterization of a circle from \( a, \vec{b} = (b_1, b_2)^T \), and \( c \), to a circle center \((c_x, c_y)\) and radius \( r \) as follows:

\[
(c_x, c_y) = \left( -\frac{b_1}{2a}, -\frac{b_2}{2a} \right) \quad r = \sqrt{\frac{||\vec{b}||^2}{4a^2} - \frac{c}{a}}.
\]

Update

10. After performing the E- and M-step, update the value of \( \sigma \) on each iteration using:

\[
\sigma = \sqrt{\frac{\sum_{i=1}^{n} w(i)d^2(i)}{\sum_{i=1}^{n} w(i)}}.
\]

11. Repeatedly perform the E- and M-steps until convergence, defined as when the difference between subsequent estimates of the circle center coordinates and radius are each less than 0.1 pixels.
Visualization

12. As shown in our sample output, on each EM iteration you should display two images: (1) the current circle superimposed on top of the original RGB image; and (2) the current probability of each pixel location belonging to model 1, visualized as a binary image.

13. Run your code on each image disc1.jpg, disc2.jpg, ..., disc5.jpg.

Reflection

14. If your solution is similar to ours, it will fail on the image disc4.jpg. Describe why you think this is and how you might remedy this problem.