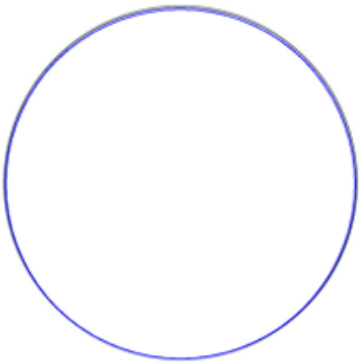
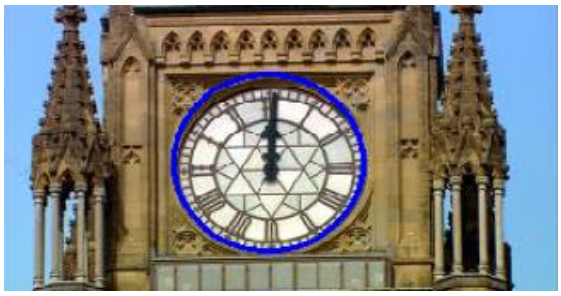




ELEC 474

Lab 3 – RANSAC

RANSAC Circle Detection

Kevin Huang and Michael Greenspan

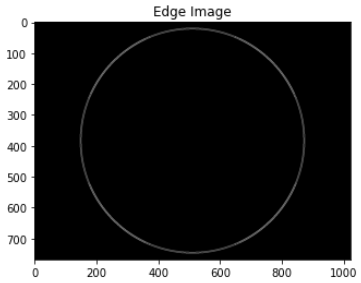
Contents

1. RANSAC Circle Detection

For this lab you will be implementing a Random Sample Consensus (RANSAC) Circle Detection Algorithm that will display the single most significant circle within an image.

1.1 Edge Image

You will need a binary edge image on which to apply RANSAC. For this, you can use the edge extraction routine that you developed in Prelab 3, or OpenCV's Canny edge detector, whichever you prefer. For the test input image "circle.jpg", the resulting binary edge image should look something like this:



1.2 Three Point Circle Fitting

The detected circles should be in the form of: $(x - x_c)^2 + (y - y_c)^2 = r^2$, where (x_c, y_c) is the center of the circle and r is the radius. For this use either the "Perpendicular Bisector" method, or the "Algebraic Method".

Perpendicular Bisector Method:

1. Generate two lines from the three input points (e.g. line 1 = point 1 to point 2, line 2 = point 1 to point 3).
2. Calculate the **middle points** for these lines.
3. Calculate the **slope** m of your lines, and find the **perpendicular slopes** $m_{per} = -\frac{1}{m}$. Be careful here as you are dividing to find the perpendicular slope, make sure you are accounting for the possibility of dividing by a very small number, or zero. This will happen if your chosen line is horizontal.
4. From the **middle points** and your **perpendicular slopes**, find the equations of the two perpendicular bisectors.

5. The center of the circle is the point of intersection of the two perpendicular bisectors.
6. The radius of the circle is the distance of the center to any of the three input points.

Algebraic Method:

1. See lecture slides 56-57.

Notes:

- Be sure to check the three points for **collinearity**. If they are collinear, then they lie on the same line and do not form a circle.
- You should have each subsection coded as separate functions. Test each function for functionality (e.g. are the three points being generated and returned correctly)
- Remember to account for image bounds.
- Use the **cv2.circle()** function to draw your circles for debugging.

1.3 RANSAC

Implement RANSAC using the following algorithm structure:

1. Initialize variables iteration number $I=0$, *max* count $C = -1$, and best circle $(best\_x_c, best\_y_c, best\_r) = (0,0,0)$

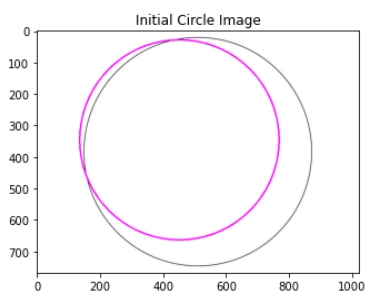
2. In a loop:
    a. Randomly select 3 edge points. From these 3 points, estimate the circle parameters (x_c,y_c,r) using the circle fitting method that you implemented in Section 1.2 .

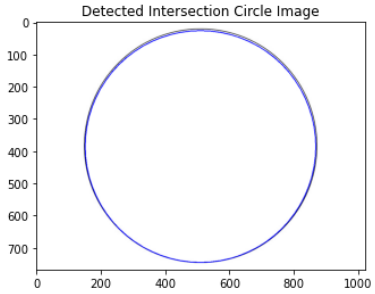
    b. Count the number K of inlier edge pixels that are close to (i.e. lie within a small distance e_1) the circumference of the estimated circle.

    c. If $K > C$, then update your variables so that the current estimated circle is now the best circle:

        i. $(best\_x_c, best\_y_c, best\_r) = (x_c,y_c,r)$
        ii. $C = K$

    d. Increment your iteration number $I = I + 1$, and repeat Steps 2a to 2d until your stopping criterion has been met.





Notes:

- In Step 2b, the simple approach is to loop through all edge pixels, and estimate their distance to the circle circumference. This will also be slow. A more efficient way is to generate the locations of the pixels that lie on the estimated circle circumference (e.g. using **cv2.circle()**, or **cv2.fitEllipse()**), and compare these to the edge image values.
- The stopping criterion of Step 2d can be a maximum number of iterations $I = N$. Alternately, another stopping criterion could be a minimum threshold on the value of C, or a ratio of C to the number of edge pixels.
- Don't be afraid to have a large iterations (e.g. $I > 10000$).

1.4 Post Processing

The circle returned from RANSAC can be improved upon further, to more accurately describe the data. The ***inliers*** are the edge points that lie within some small distance e_2 of the estimated circle circumference.

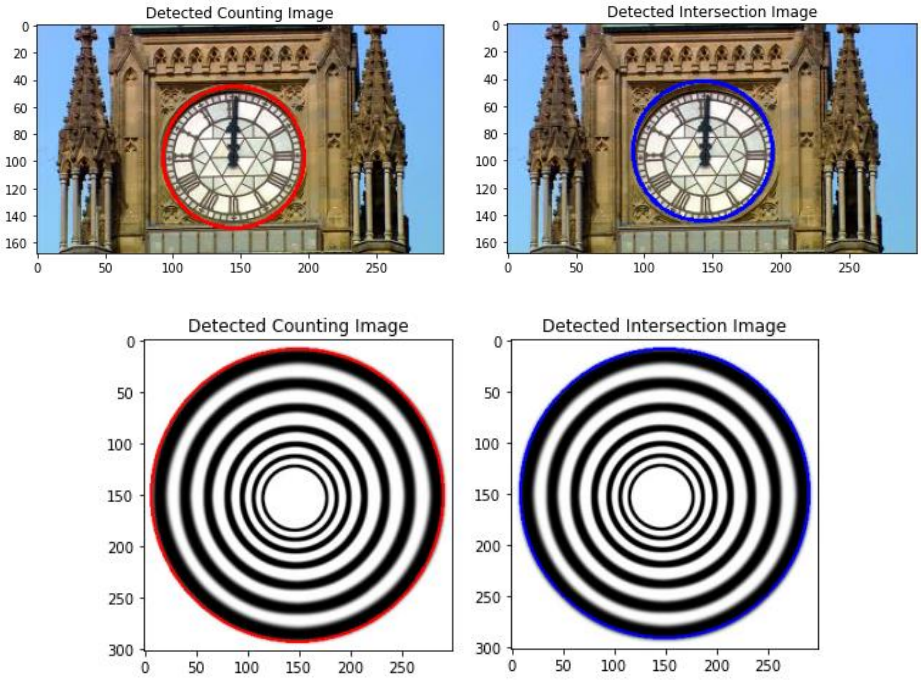
First, recalculate the circle center, by finding the centroid of the set of inliers.

Next, using this new center, recalculate the circle radius, by finding the mean distance from the new center to all inliers.

Notes:

- e_2 should be no larger than e_1 from RANSAC Step 2b.

2. Submission



The submission for this lab should include a .zip of:

- .ipynb file that:
  - Your code RANSAC Circle Detection.
    - Tested your code on “circle.jpg”, “concentric_circles.jpg”, and “parliament_clock.jpg”.
  - Each tested image with their best circle overlaid onto the test image
  - For each result, indicate the following:
    - The number of iterations required
    - The estimated circle center and radius values before and after preprocessing

Your code will be run in Jupyter Lab to test for functionality.

For full marks, follow the lab submission instructions specified under ‘Content->Reources’ on the course OnQ site.

The marking rubric is as follows:

Item	mark
1. Three point circle fitting correct	0.5
2. RANSAC correct	2
3. Post processing correct	1
4. Submission format correct	0.5
Total:	4