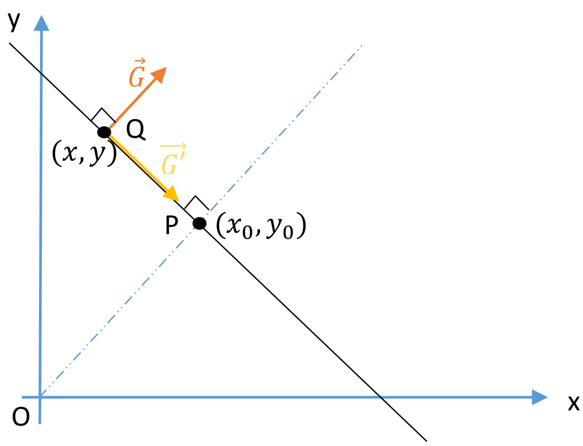
ME6406 – Machine Vision – HW#2

Problem 1: Hough Transform

a. We use the following figure to derive the relationship of the question :



$\vec{G}$ is the gradient of the straight line at the point (x,y) and $\overrightarrow{G'}$ is perpendicular to the gradient such that :

$\vec{G} = [g_x, g_y]$ and $\vec{G}' = [g_y, -g_x]$

The straight line is collinear with $\overrightarrow{G'}$, so we can express the slope of the line as :

$y = ax + b$ where $a = \frac{\Delta y}{\Delta x} = -\frac{g_x}{g_y} \rightarrow$ The slope is constant along a straight line, so at point P, we get:

$a = -\frac{x_0}{y_0}$ (1)

Then, we also have that the vector $\overrightarrow{OP} = [x_0, y_0]$ and the vector $\overrightarrow{PQ} = [x - x_0, y - y_0]$ are perpendicular. So, we also get:

$$\overrightarrow{OP}.\overrightarrow{PQ} = 0 \Rightarrow x_0(x - x_0) + y_0(y - y_0) = 0 \ (2)$$

By using equations (1) and (2), we have:

$$\begin{cases} y_0 = x_0 \dfrac{g_y}{g_x} \\ x_0(x - x_0) + x_0 \dfrac{g_y}{g_x}\left(y - x_0 \dfrac{g_y}{g_x}\right) = 0 \end{cases} \Rightarrow \begin{cases} y_0 = x_0 \dfrac{g_y}{g_x} \\ x - x_0 + \dfrac{g_y}{g_x} y - x_0 \left(\dfrac{g_y}{g_x}\right)^2 = 0 \end{cases}$$

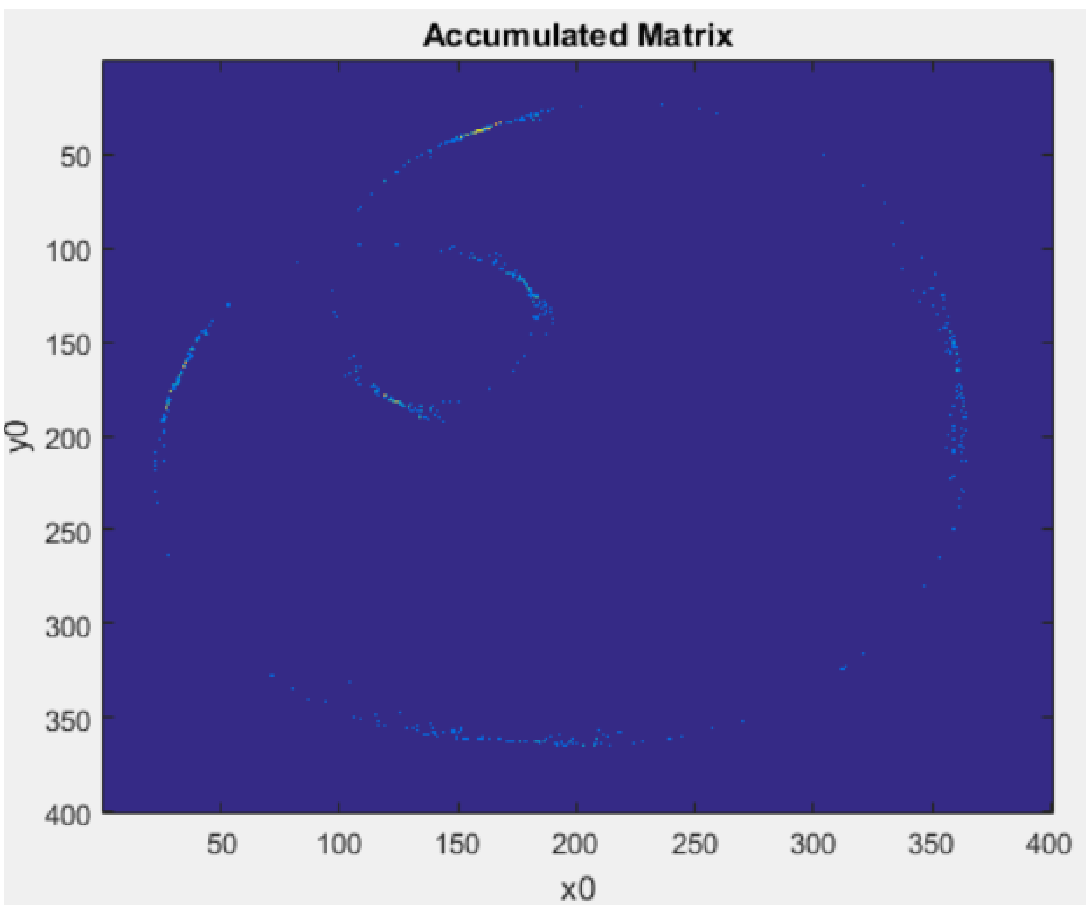
$$\Rightarrow \begin{cases} y_0 = x_0 \dfrac{g_y}{g_x} \\ x_0\left(1 + \left(\dfrac{g_y}{g_x}\right)^2\right) = x + \dfrac{g_y}{g_x} y \end{cases} \Rightarrow \begin{cases} y_0 = x_0 \dfrac{g_y}{g_x} \\ x_0 = \dfrac{x g_x + y g_y}{g_x} . \dfrac{g_x^2}{g_x^2 + g_y^2} \end{cases}$$

$$\Rightarrow \begin{cases} y_0 = g_y \dfrac{x g_x + y g_y}{g_x^2 + g_y^2} \\ x_0 = g_x \dfrac{x g_x + y g_y}{g_x^2 + g_y^2} \end{cases}$$

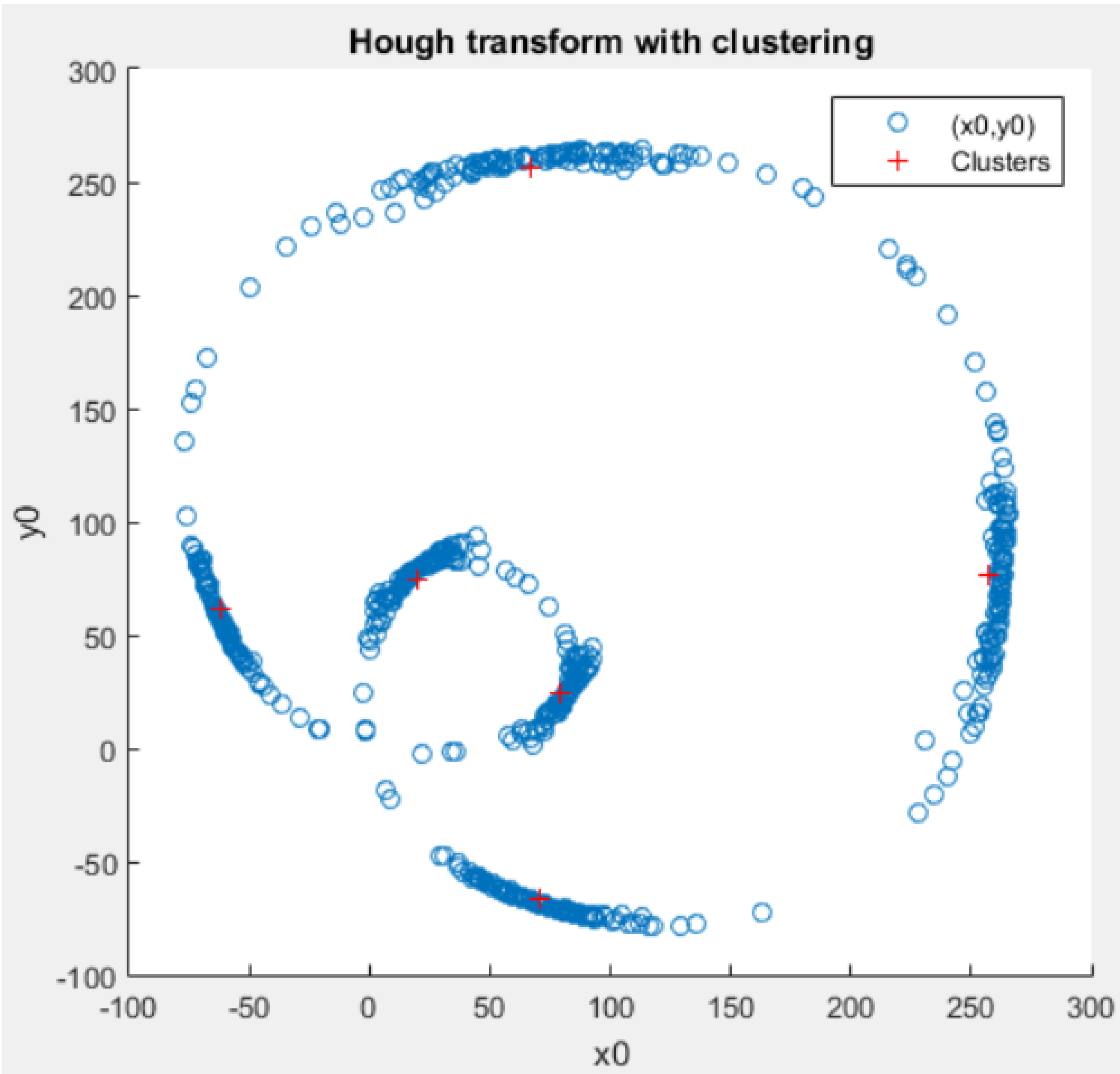
If we take $v = \frac{x g_x + y g_y}{g_x^2 + g_y^2}$, then we end up with $\binom{x_0}{y_0} = v \binom{g_x}{g_y}$

b.

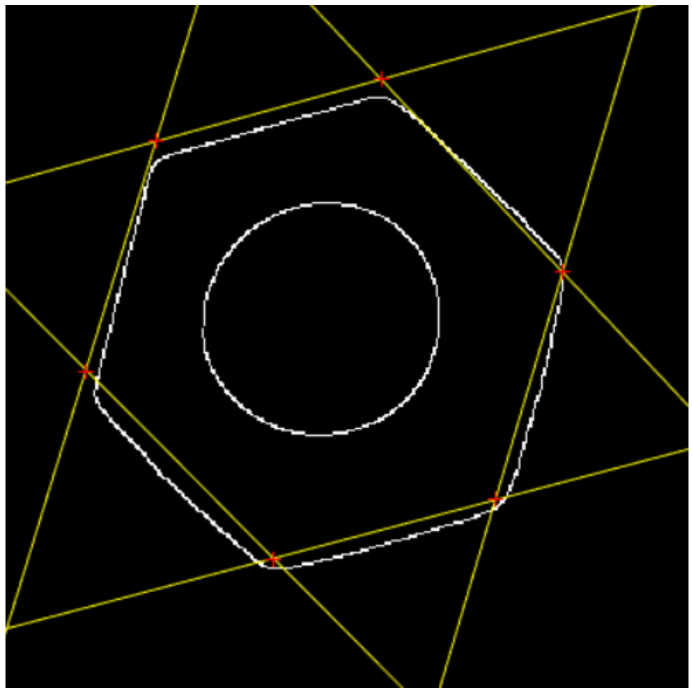
1) We use the previous equations to perform Hough Transform. We accumulated each value of (x0,y0) with the number of their repetition in the accumulated matrix. The method is detailed directly on the script.



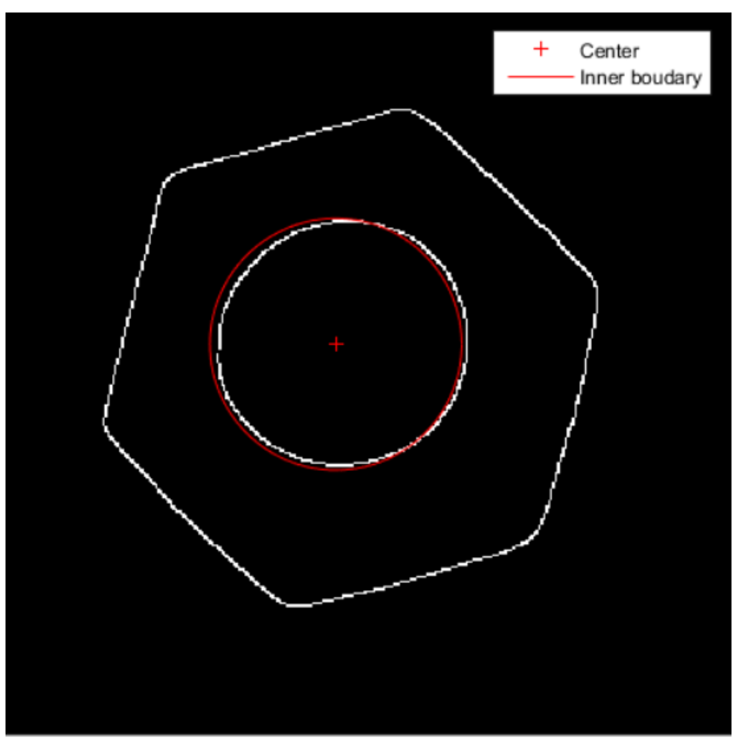
2) We also accumulate the values of (x0,y0) in an array that give us directly the different values of (x0,y0) and we just have to find the areas where there is lot of similar (x0,y0). We use k-means clustering method to partition these data. You can see on the plots the clusters of (x0,y0).



3) Now that we have the six centroid values of (x0,y0), we determine the equations corresponding to each straight lines by using the relationships fund in the first question, such that we find a relationship between y and x with $x_0(x - x_0) + y_0(y - y_0) = 0$
   To get the edges, we compute the intersections of the straight lines and we take the ones near the outer boundaries.

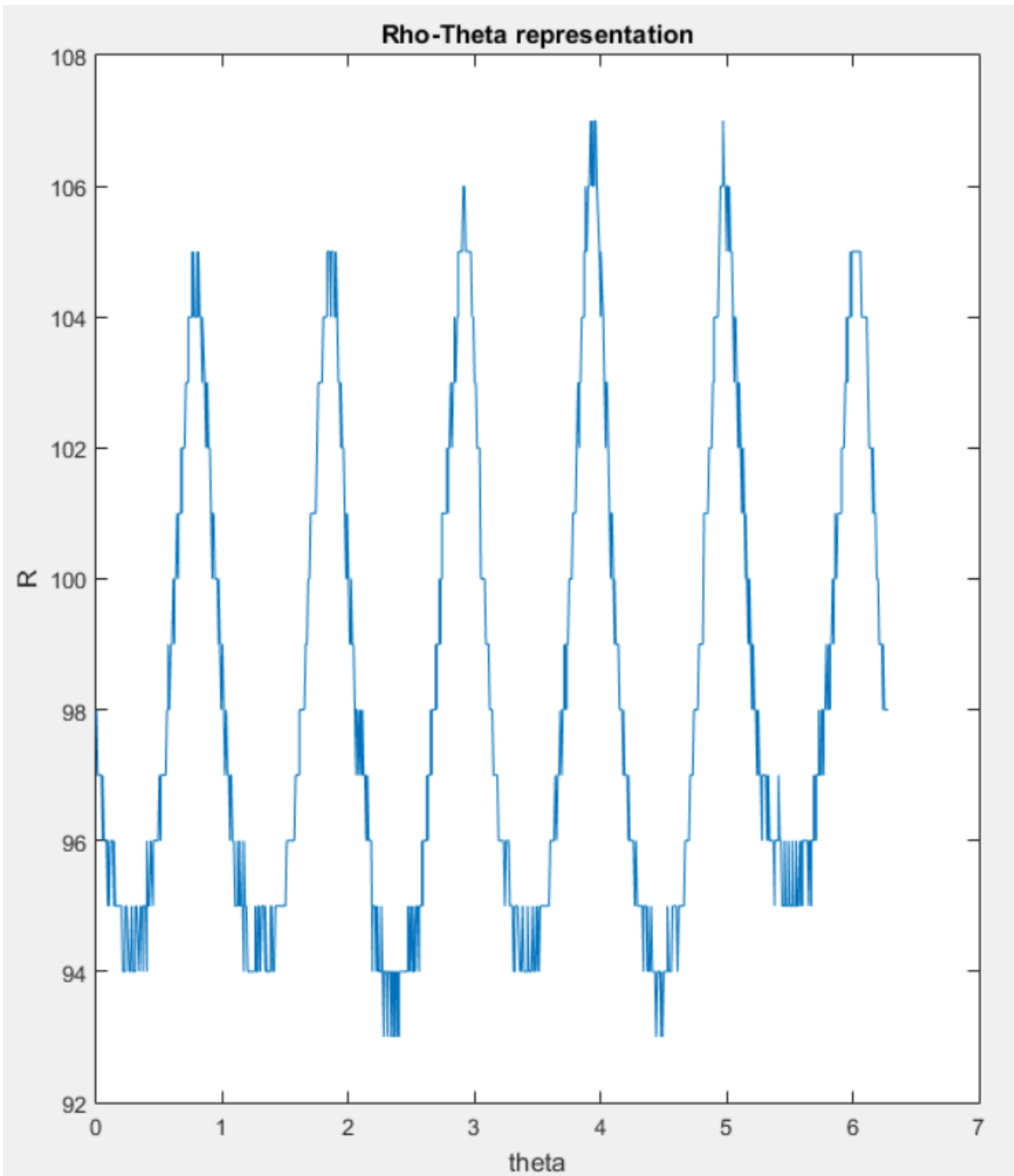


c. We use the Hough Transform method in rho-theta to get the inner boundary. We compute each values of (xc, yc, R) and we accumulate their values. Then, we find the maximum point of the accumulated matrix to get the centroid and radius of the circle.

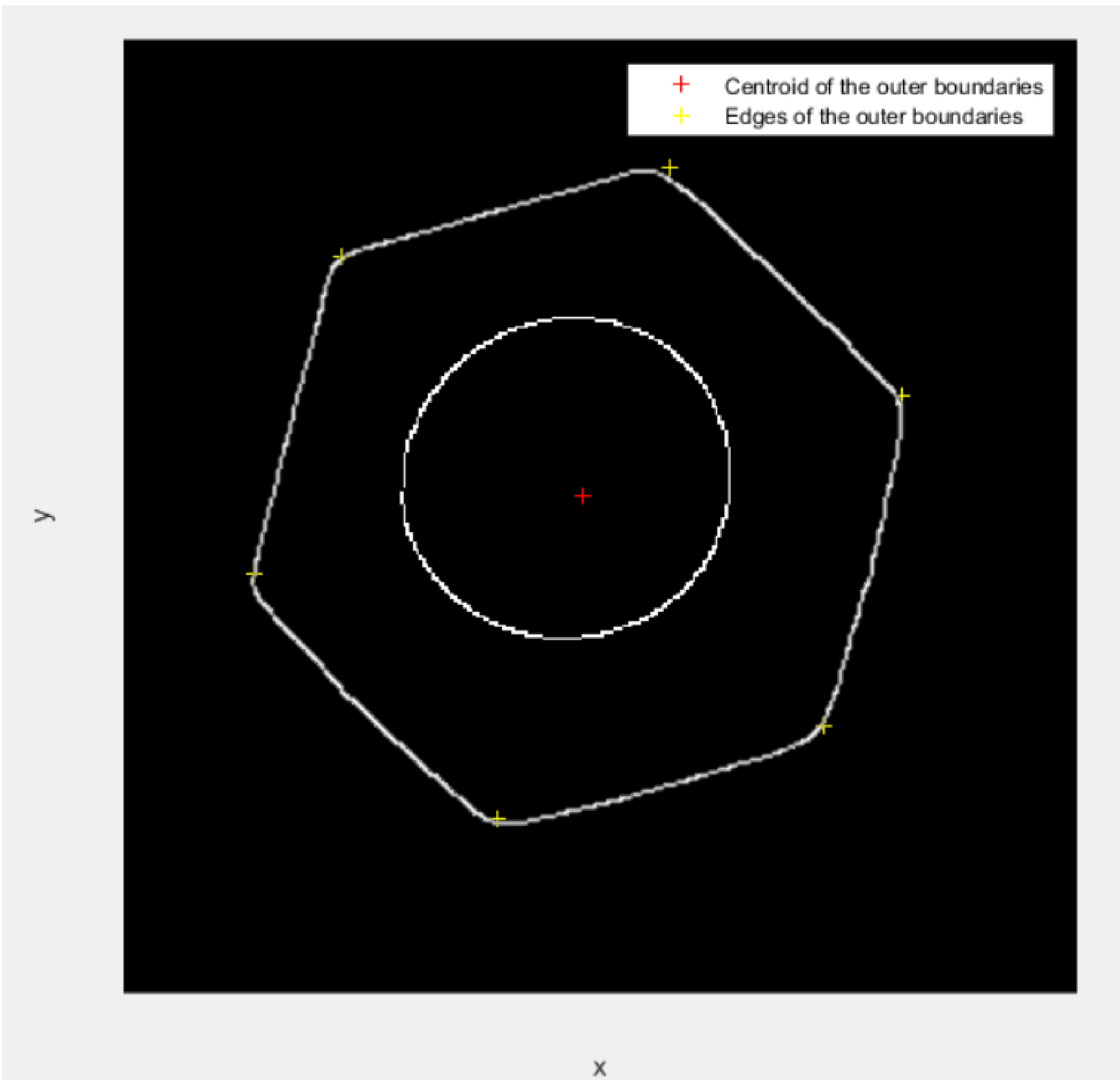


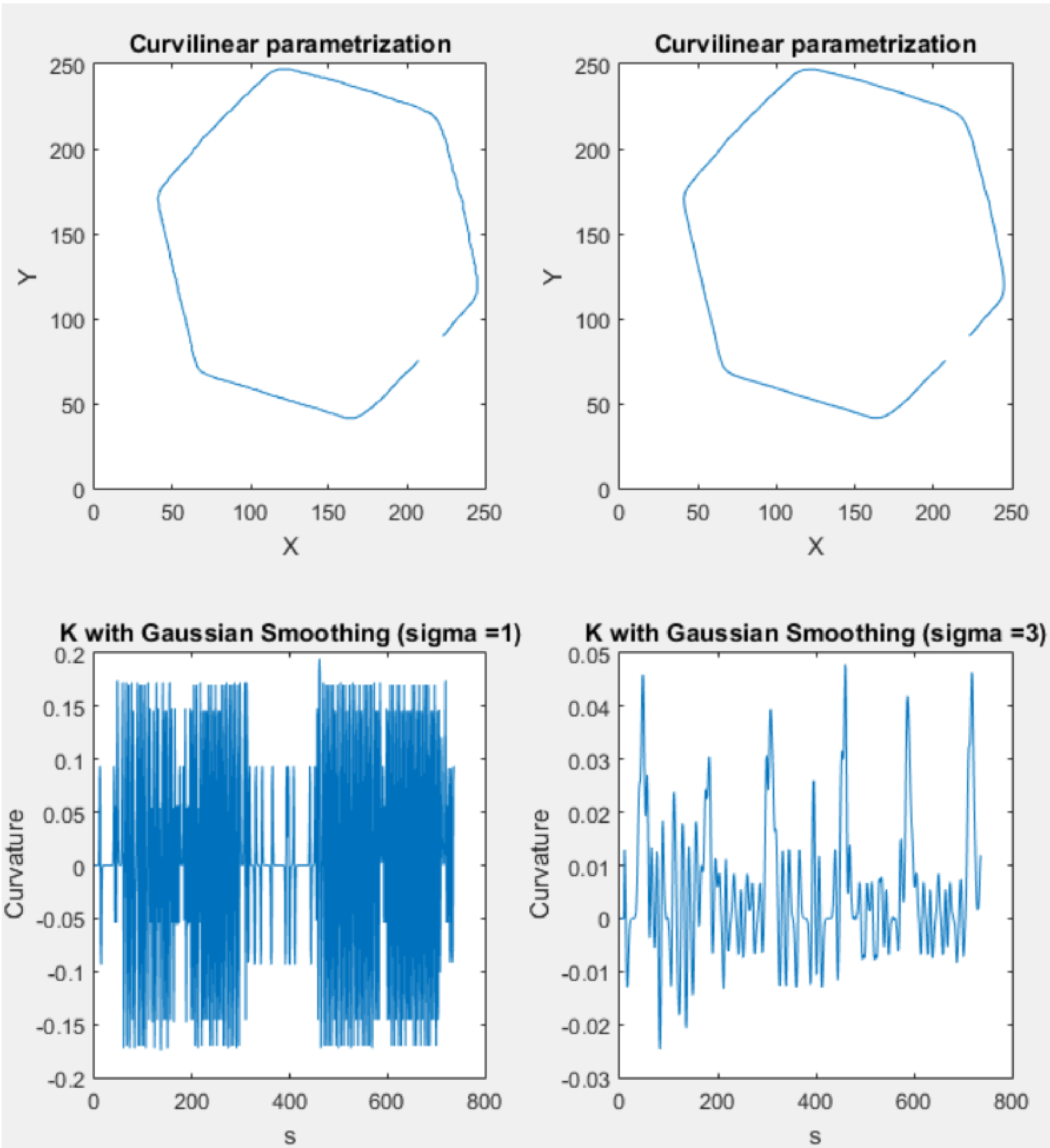
Problem 2: Feature Points Detection

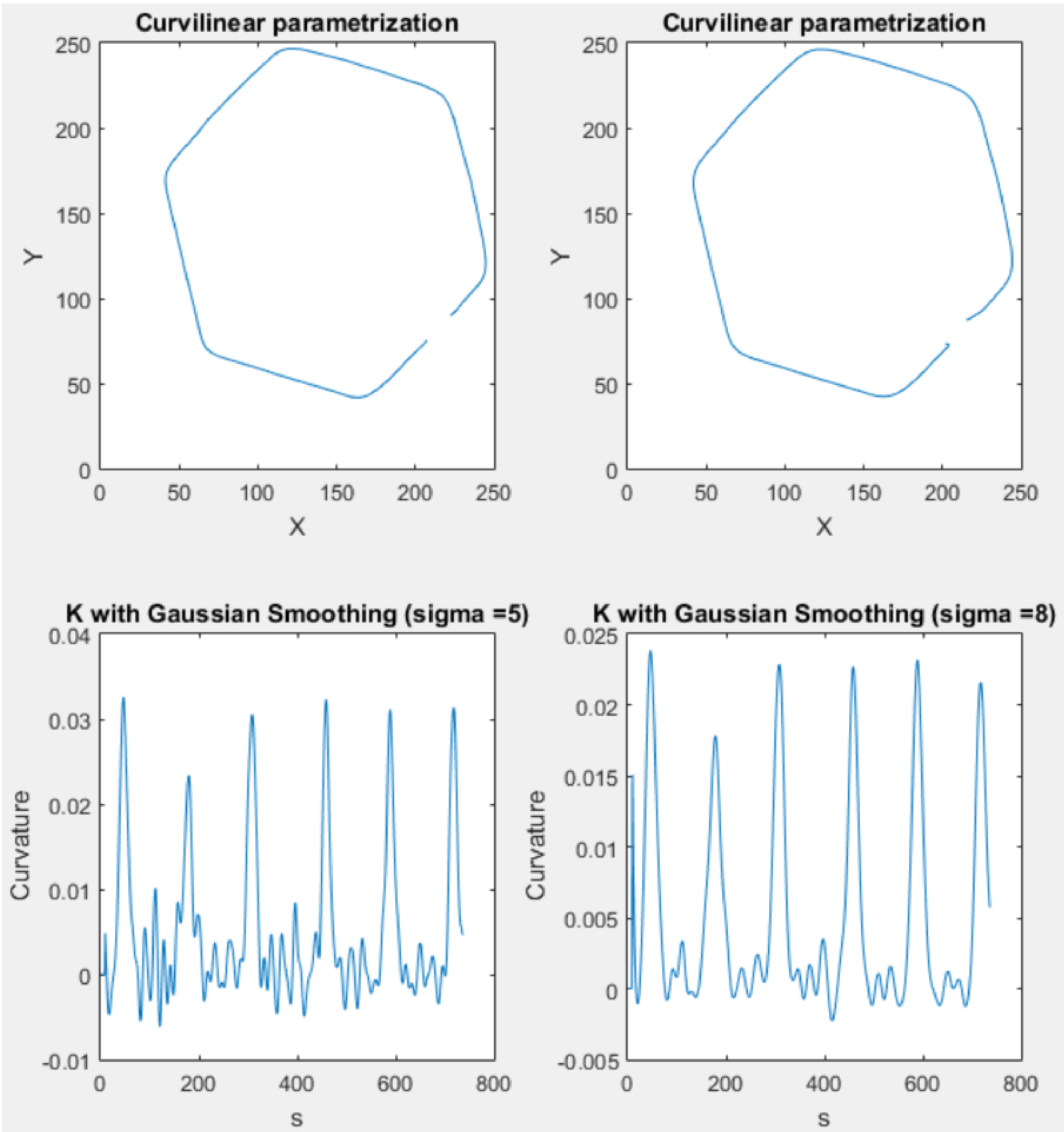
a. We find the centroid of the inner boundary. Then, for each theta, we find the radius equal to the distance between the centroid and the outer boundaries. The maximums of the radius



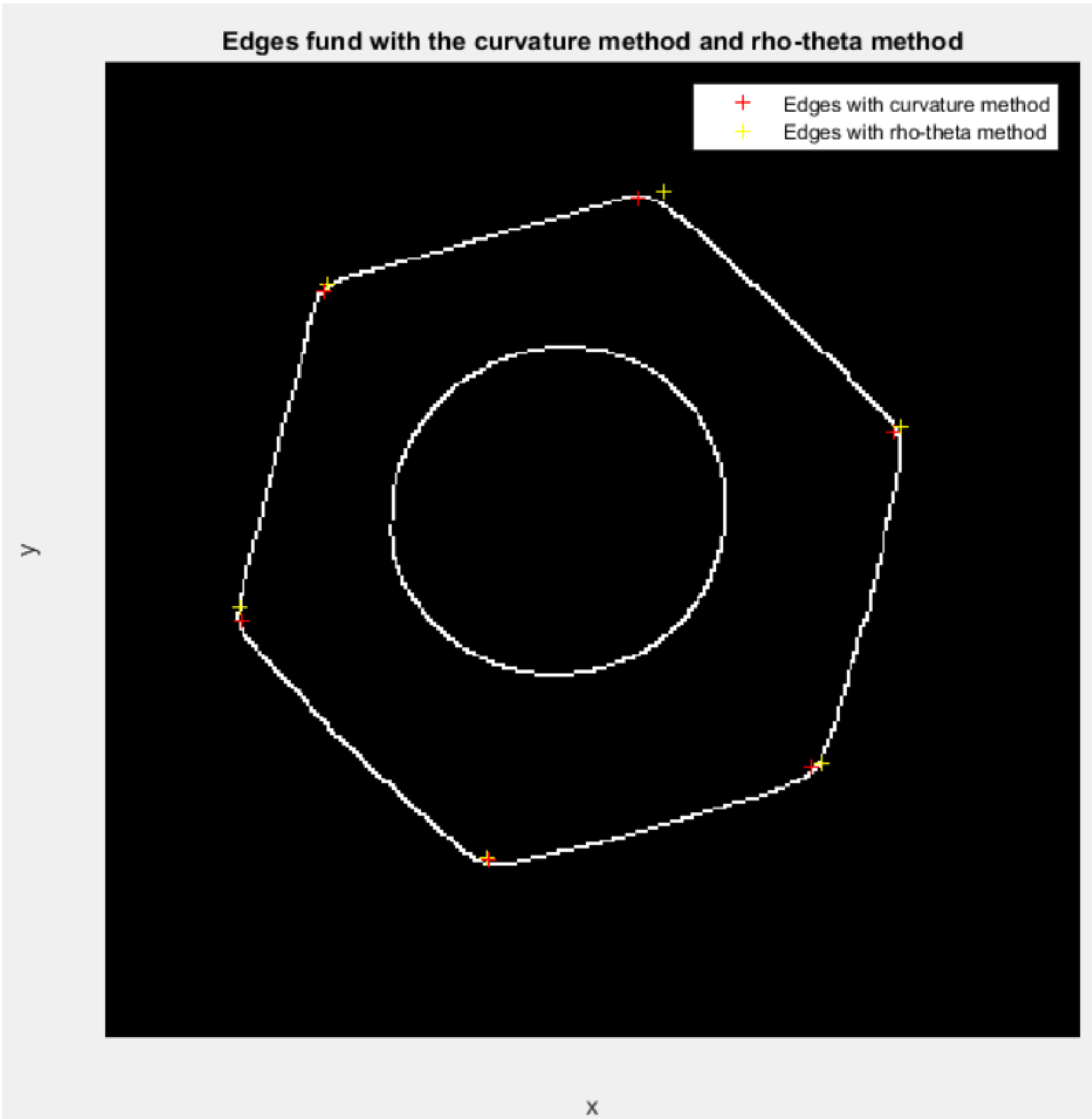
b. The second method to detect the edge is to use a curvilinear parametrization of the outer boundaries, such that the local maximums of its curvature are at the edges. Moreover, we have to smooth the initiale curve by using convolution with Gaussian functions. Indeed, the curvature of the initiale curve is to discontinue to get proper values of the maximums.




Curvilinear parametrization
Curvilinear parametrization
K with Gaussian Smoothing (sigma =1)
K with Gaussian Smoothing (sigma =3)
X
Y
s
Curvature

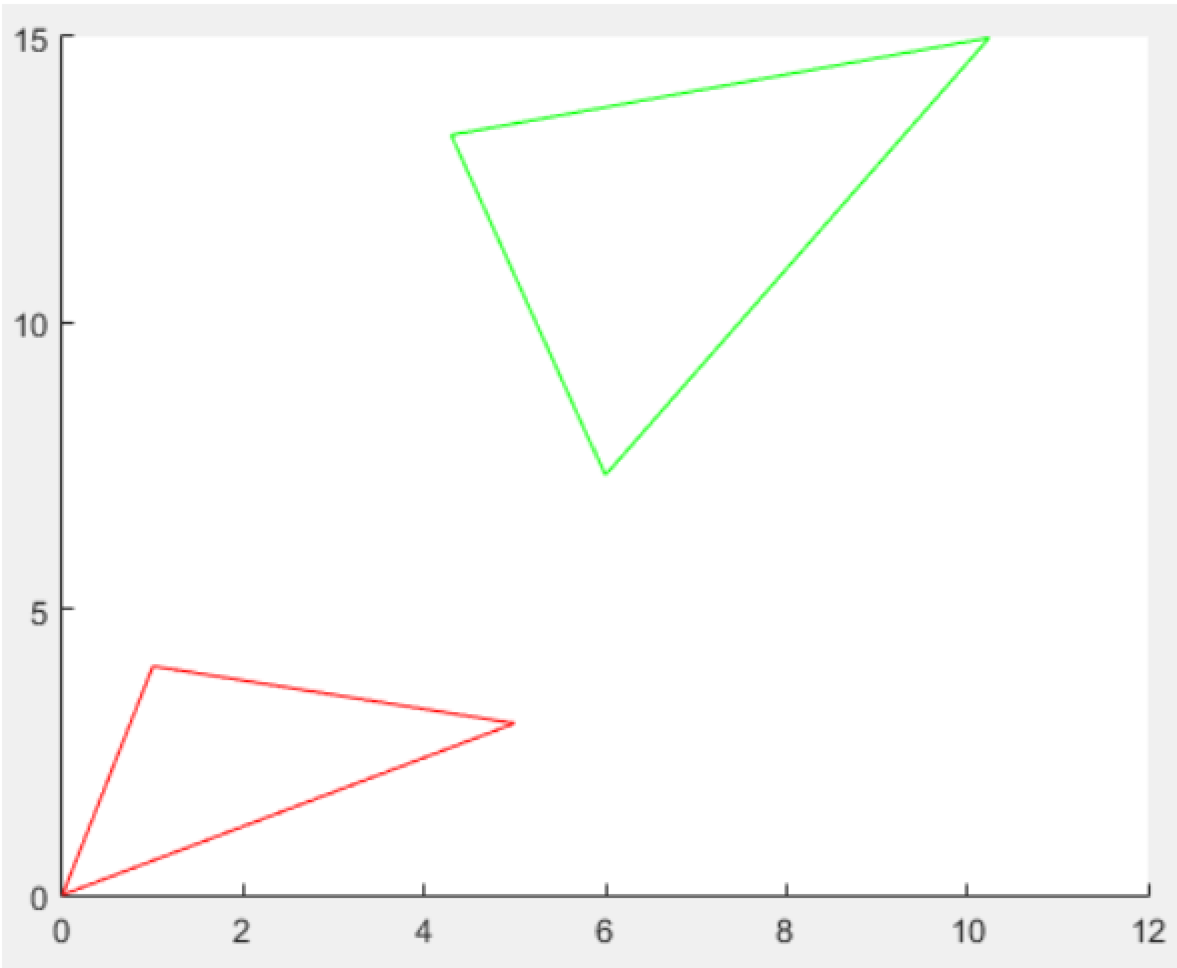

Curvilinear parametrization
Y
X
Curvilinear parametrization
Y
X
K with Gaussian Smoothing (sigma =5)
Curvature
s
K with Gaussian Smoothing (sigma =8)
Curvature
s

*

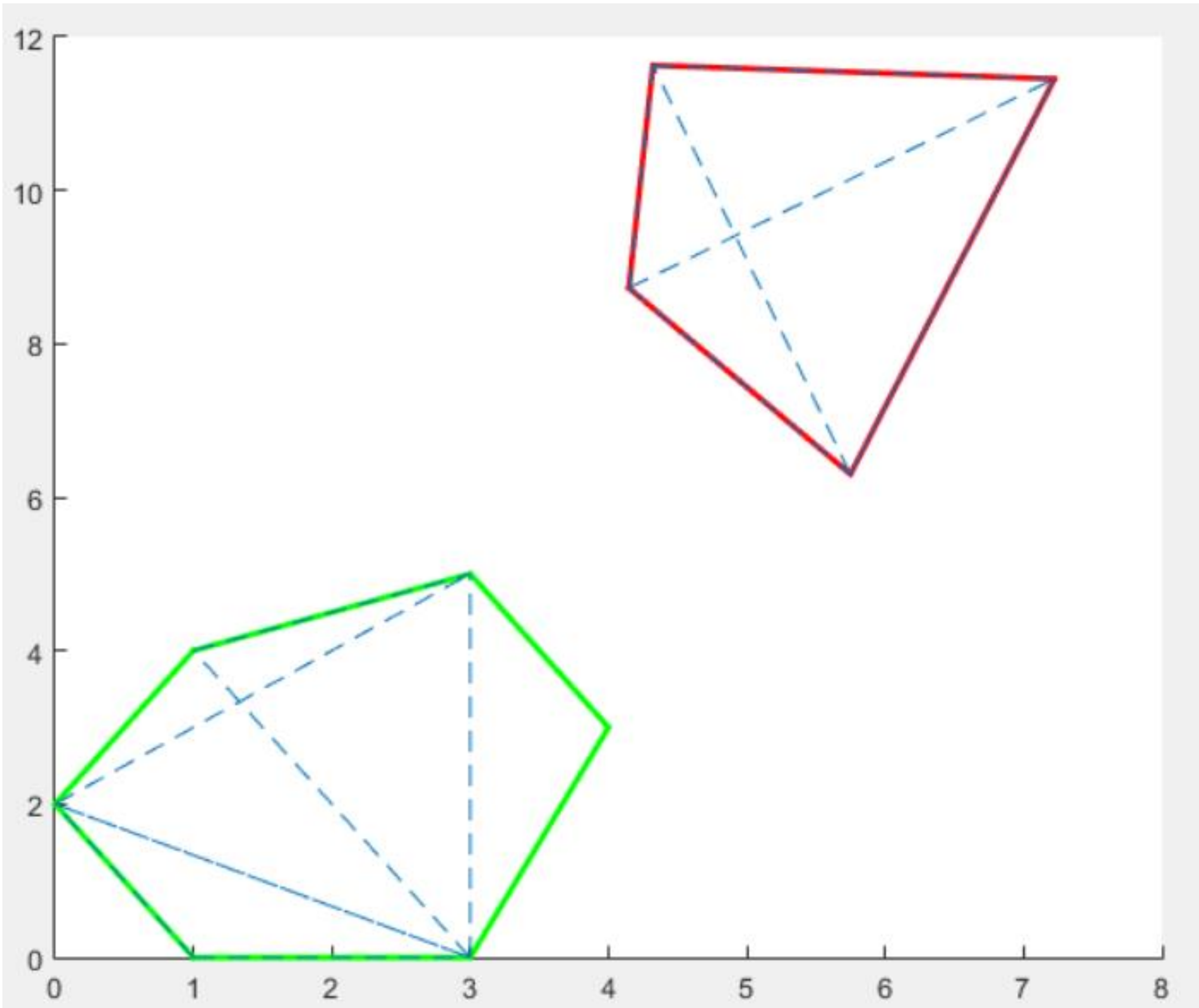

Edges fund with the curvature method and rho-theta method
Edges with curvature method
Edges with rho-theta method
y
x

Problem 3: Template Matching

a) We use the function Template_to_image and Pseudo_inverse to answer to the question 1 and 2. The details of the method used is on the script.



b) We calculate the distance between all combination of triangle from the template to the image and we find the ones which match the more accuratly the image (in dashed line). We find k=1.15, theta = -0.27, xd=-0.096 and yd=4.19.



12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8