2) `Start(#spatialdiscretization,#timesteps,#t0 = initial time,#tf = final time,#curvenumber)`

`It will start the execution`

Curvenumber=1,2,3 for three curves above.

Above code was run by:

Curve1:

`Start (`512,30000,0,4,1`)`

SUBMISSION has only 1 File : Main.m

2) Texture Detection

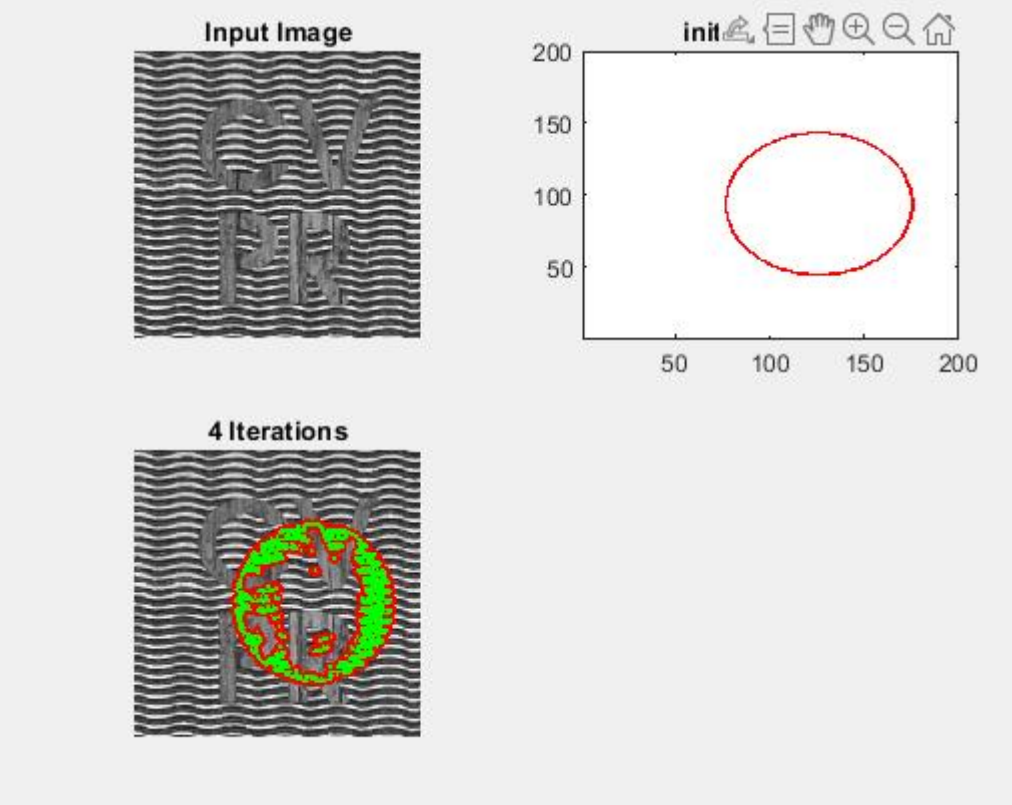
i) the Euclidean Forbenius norm

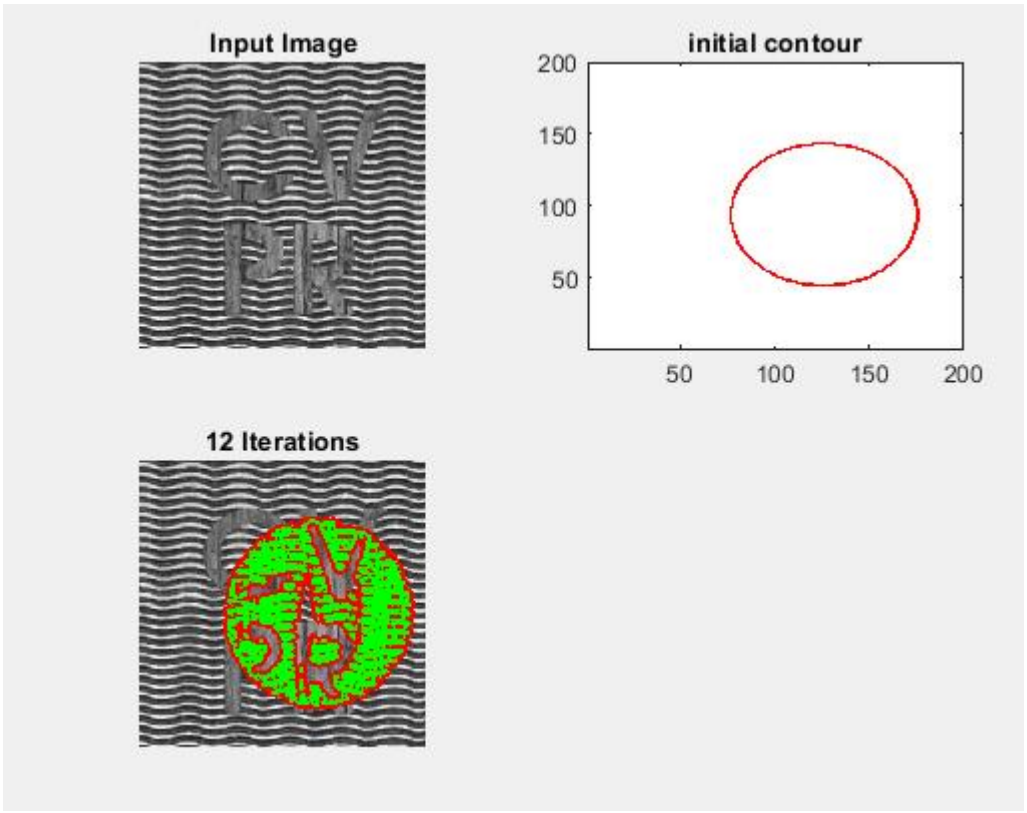
Kernel used was gaussian.

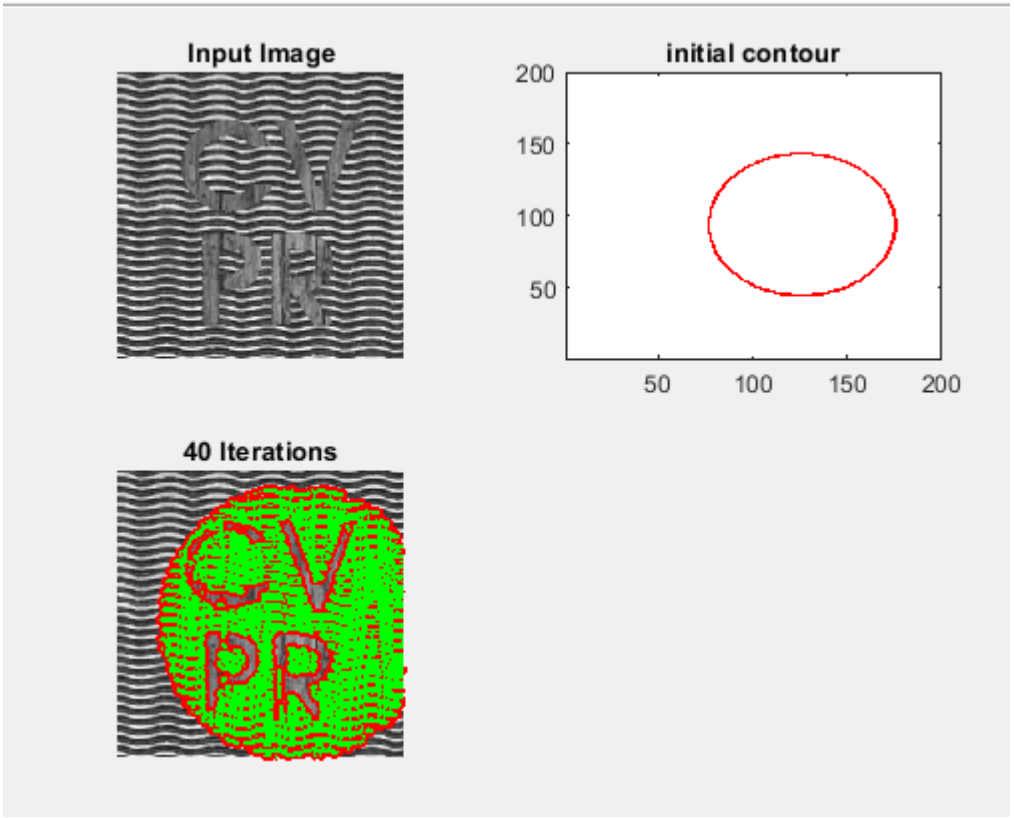
$$\mathbf{T} = K_\rho * (\nabla \mathcal{I}\ \nabla \mathcal{I}^T) = \begin{pmatrix} K_\rho * \mathcal{I}_x^2 & K_\rho * \mathcal{I}_x \mathcal{I}_y \\ K_\rho * \mathcal{I}_x \mathcal{I}_y & K_\rho * \mathcal{I}_y^2 \end{pmatrix}$$

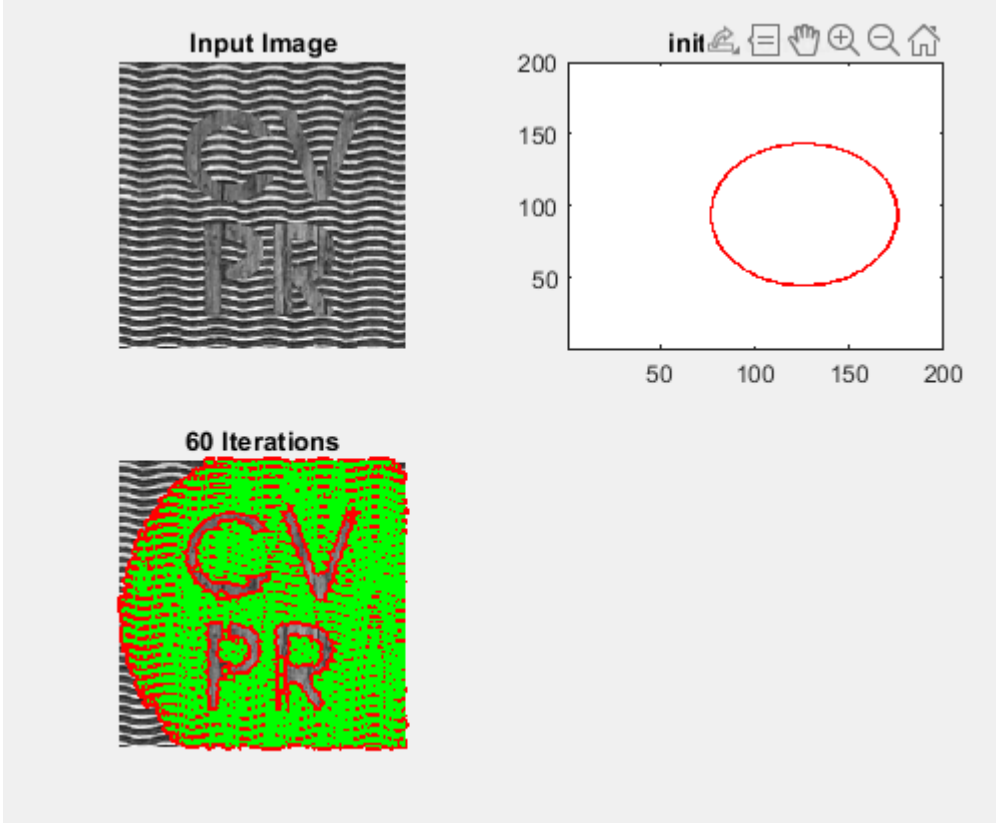
Tensor Flow for each pixel was built using the above epression:

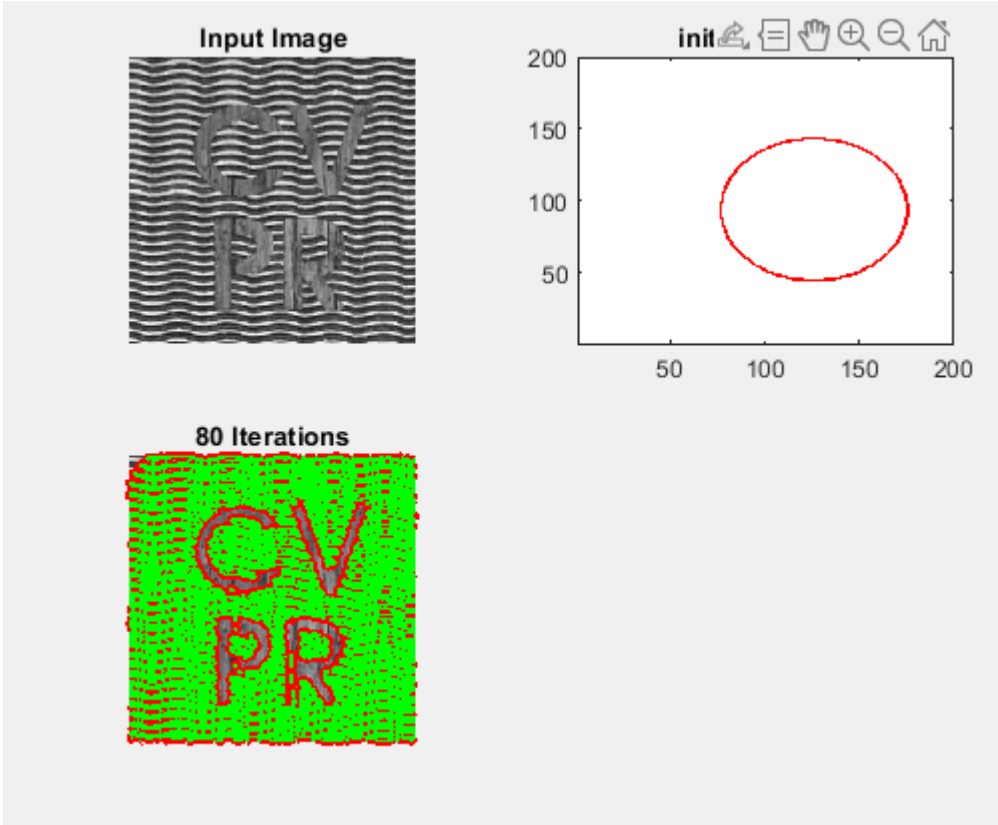
Below are the results for at different iterations:

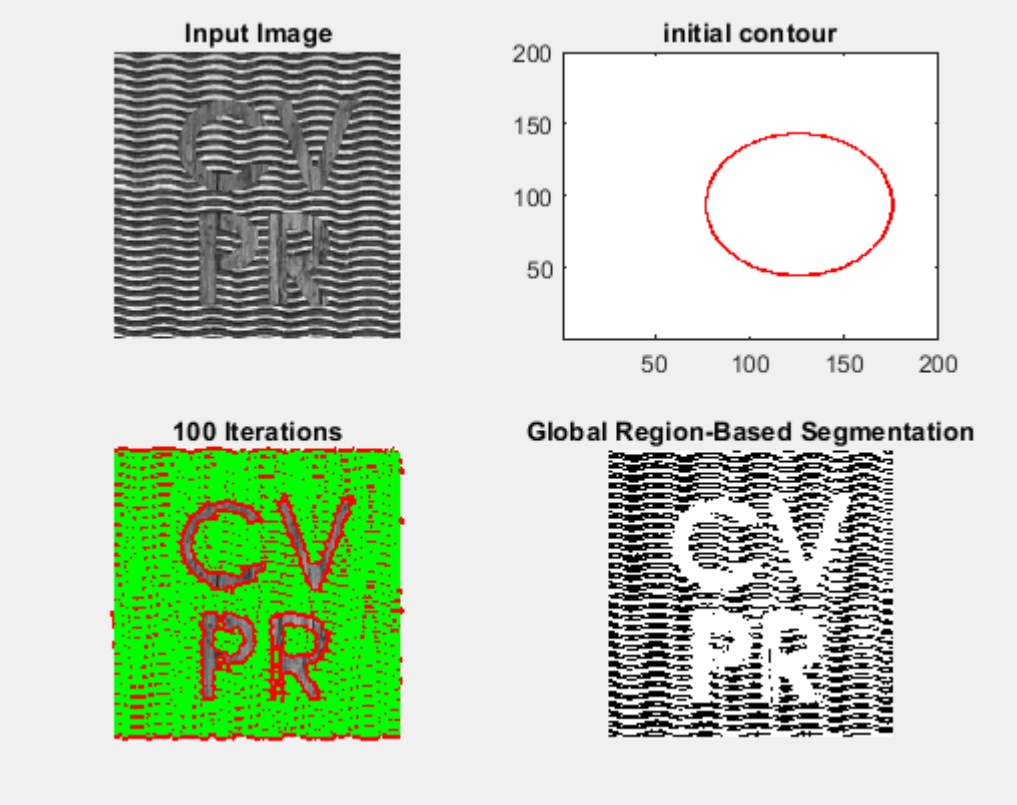



Input Image
initial contour
200
150
100
50
50
100
150
200
12 Iterations


Input Image
initial contour
200
150
100
50
50
100
150
200
40 Iterations


Input Image
init
200
150
100
50
50
100
150
200
60 Iterations


Input Image
init
200
150
100
50
50
100
150
200
80 Iterations



Animation:

https://uflorida-my.sharepoint.com/:v:/g/personal/jyotik_parikshya_ufl_edu/EWwKYK6E-HtAtgm55ZwFHLABR6FCFYmdYhyMEKW4Bk-x-Q?e=vOJ0rc

The changes made in the file chenvese.m are from line `145-225. A comment %CHANGED Has been added.`

Basic Extra Library Used:

```
Imgradientxy ->for directional gradient
Imgaussfilt -> for gaussian convolution

Eig -> for eigen values
Logm -> log of matrix
Norm -> for calculating Norm
```

Finding Gradient and storing 2*2 tensor flow matrix at each pixel in a 4D matrix Master.

```
[ix,iy]=imgradientxy(P);
Master=zeros(200,200,2,2);

 for i=1:200
    for j=1:200
          grad=[ix(i,j)*ix(i,j) ix(i,j)*iy(i,j); ix(i,j)*iy(i,j) iy(i,j)*iy(i,j)];
          Master(i,j,:,:)=imgaussfilt(grad,2);
    end
 end
```

Find Mean:

```
temp=Master(h,t,:,:)*hv(h,t);
sum1=sum1+[temp(:,:,1,1) temp(:,:,1,2);temp(:,:,2,1) temp(:,:,2,2)];
c1=sum1/(length(inidx)+eps);
```

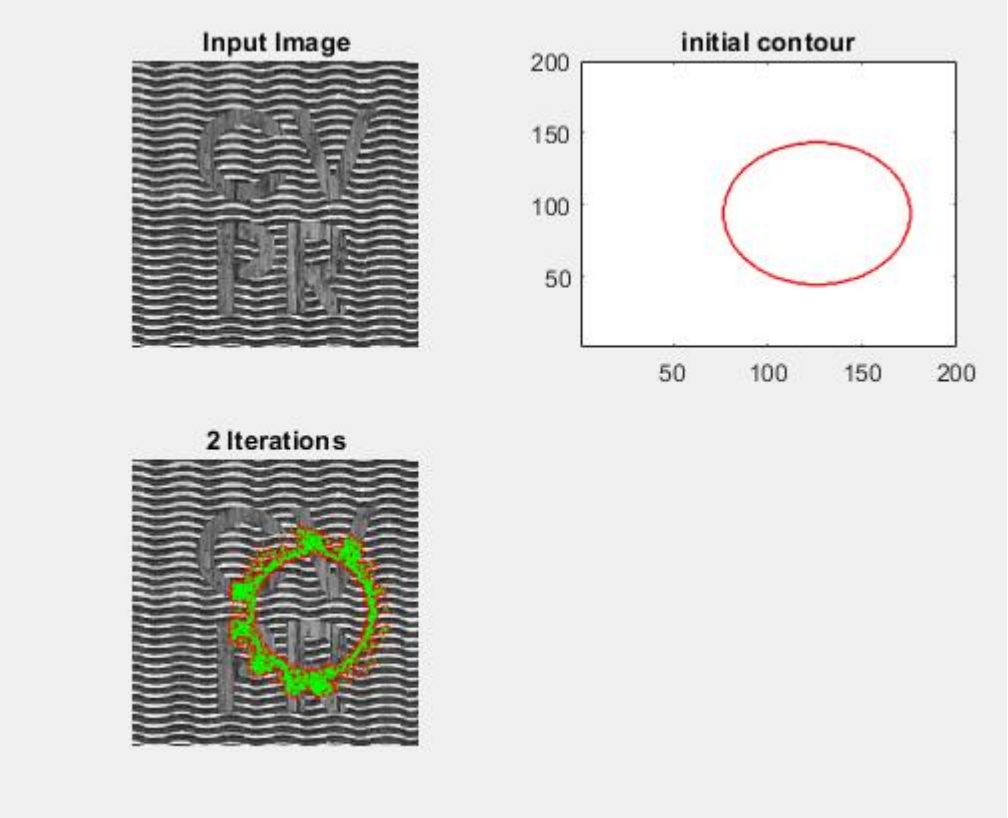
Find distances using Euclidean disance:

```
temp=Master(h,t,:,:);
k1=[temp(:,:,1,1) temp(:,:,1,2);temp(:,:,2,1) temp(:,:,2,2)]-c1;
dis1=sum(sum(k1.^2));    %Sum of all squares
dis1=dis1^(0.5);
```

ii) Log

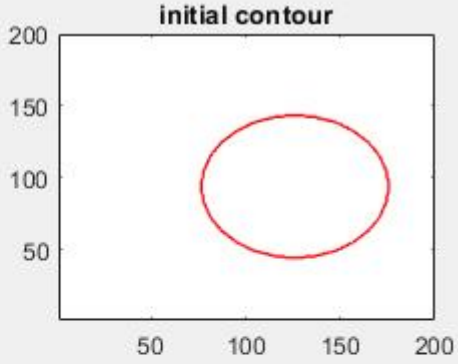
Kernel used was gaussian.

Below are the results at different iterations:

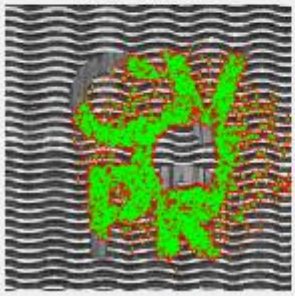


Input Image

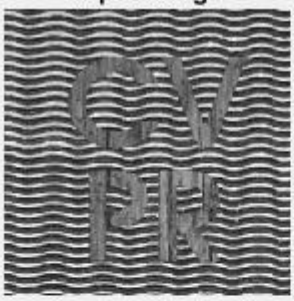
initial contour



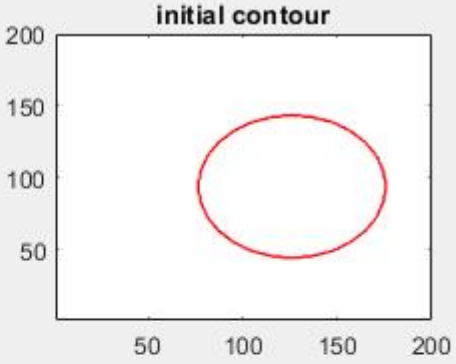
7 Iterations



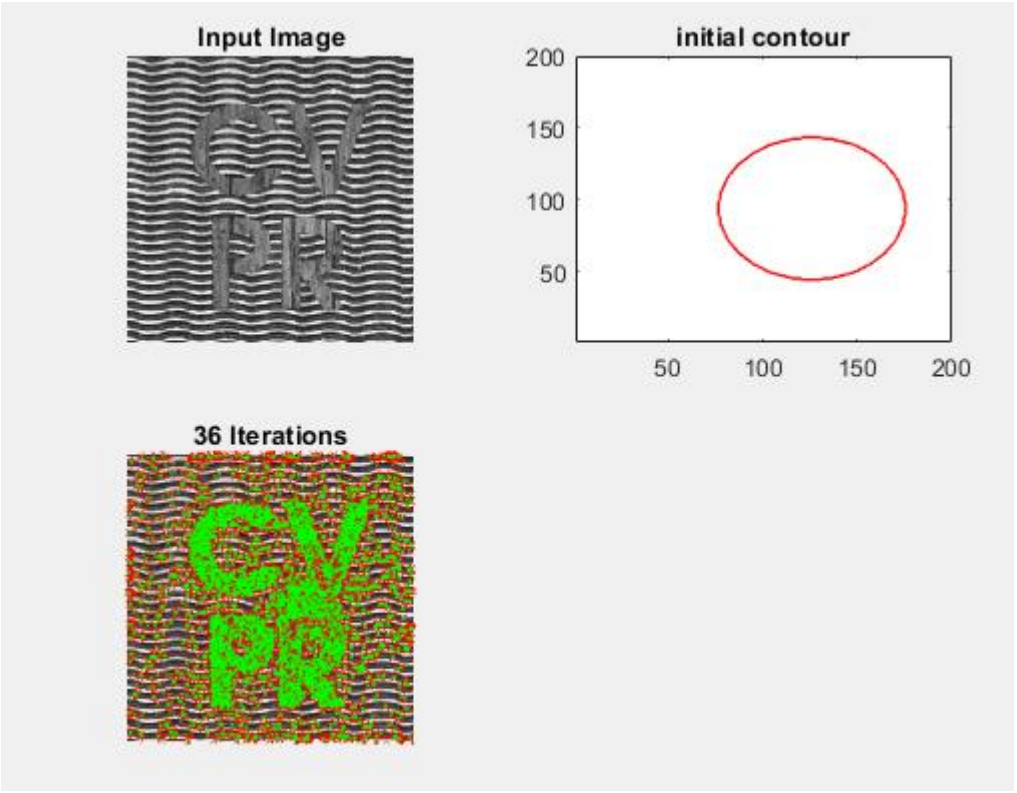
Input Image

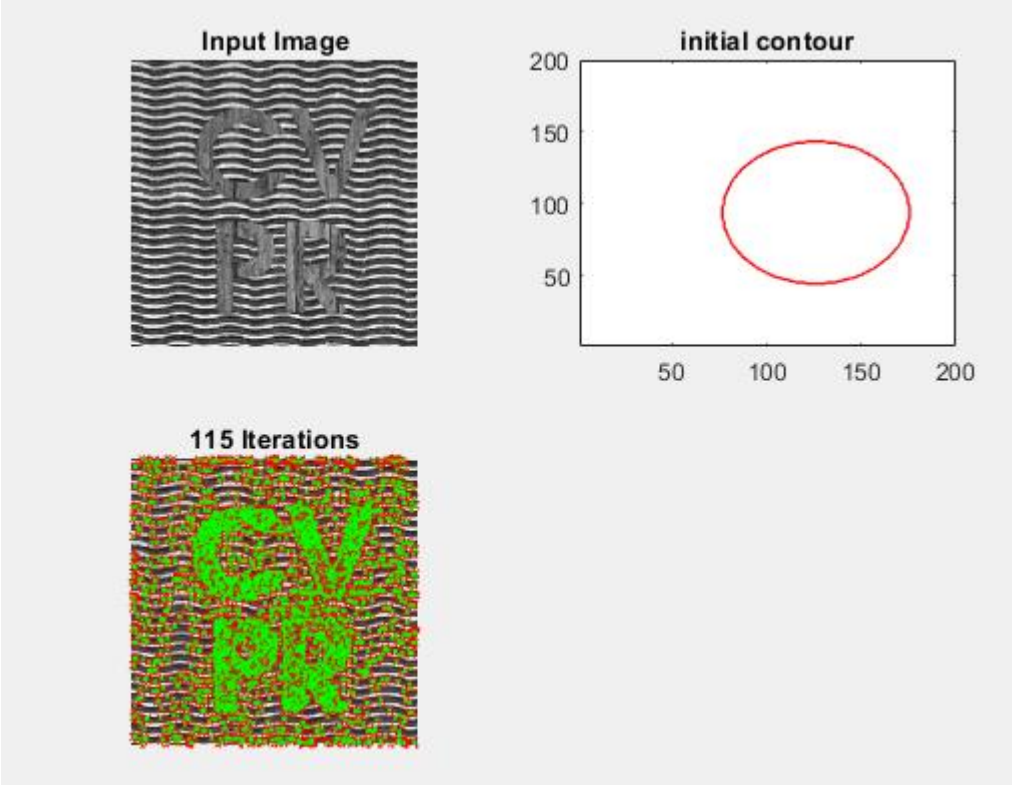


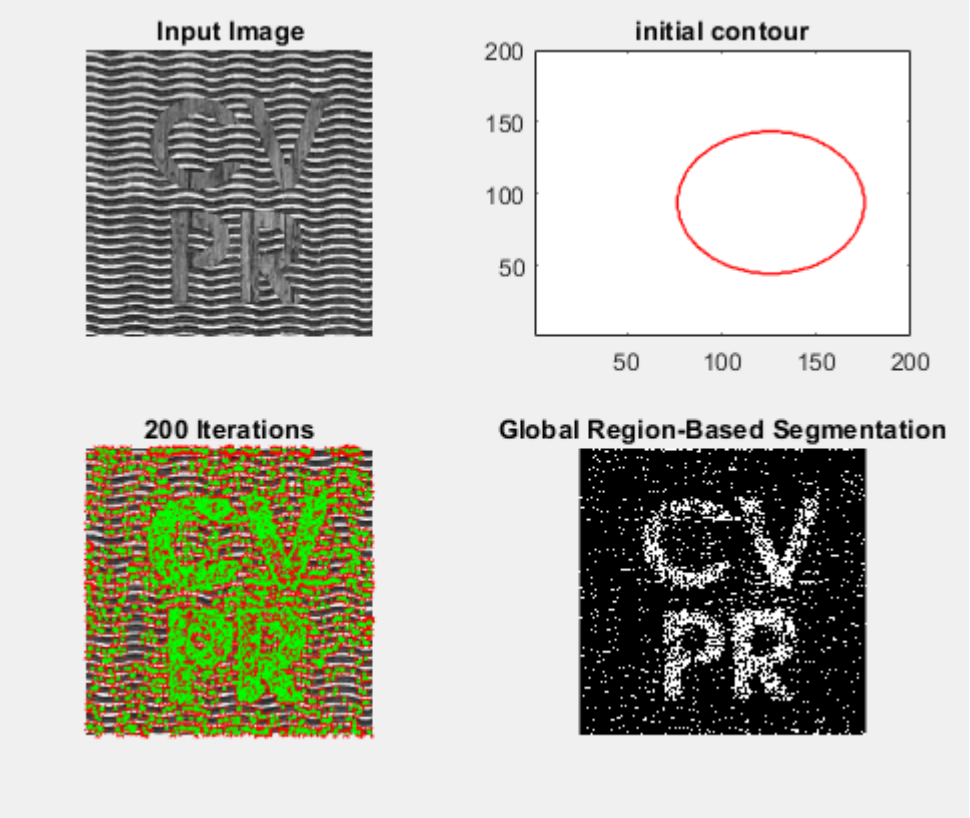
initial contour



12 Iterations


Input Image
initial contour
200
150
100
50
50
100
150
200
36 Iterations


Input Image
initial contour
200
150
100
50
50
100
150
200
115 Iterations



Find Log distance:

```
temp=Master(h,t,:,:);
k1=[temp(:,:,1,1) temp(:,:,1,2);temp(:,:,2,1) temp(:,:,2,2)];
e1=eig(k1,"matrix");
M = min(e1);
e1=e1-M+eps;
e1(e1<eps)=eps;
e2=eig(c1,"matrix");
e2(e2<eps)=eps;
M = min(e2);
e2=e2-M+eps;
[Li1,ex1]=logm(e1);
[Li2,ex2]=logm(e2);
Li1(Li1 == -Inf)=eps;
Li2(Li2 == -Inf)=eps;
Li1(isnan(Li1))=eps;
Li2(isnan(Li2))=eps;
dis1=norm((Li1-Li2),'fro')^2;
```

Observations:

1) Normal chan method doesn't work very well due to texture.
2) By taking gradient we get better results.
3) Euclidean works little better than Euclidean-log.
4) Euclidean-Log is slower due to any calculations at every iteration.
5) Changing value of mu or dt might give even better results.

How to Run:

In file demo_chenvese.m

Run:

1) For Euclidean:

```
seg1 = chenvese(I,'medium',100,0.02,'chan');
```

2) For Log-Euclidean:

```
seg2 = logchenvese(I,'medium',200,0.2,'chan');
```

Note:

1) This is a modification on base file provided in the question.
2) Run the command **warning('off','all')** before running `logchenvese to avoid warnings due too eigen decomposition.`

SUBMISSION has 9 matlab files:

1) **Chenvese.m and logchenvese.m are the modified files.**
2) **Demo_chenvese.m is used to initialize the program.**
3) **Other 6 files are base files and are unchanged.**

Total File Submission: 11

1) Main.m for Q1
2) Other 9 Matlab files for Q2 (Zip file was not allowed)
3) Report.pdf