Math 228B
Homework 6
Ahmad Zareei

Introduction

In this problem set, we will talk about interface propagation using initial value problem. Basically, we want to find a boundary of a curve Γ at different times, assuming that it is propagating with speed F normal to the boundary. We can either solve a boundary value problem or an intital value problem.

In the case of boundary value problem, we should solve the following equation:

$$|\nabla T|F = 1, \qquad T = 0 \text{ on } \Gamma \tag{1}$$

where T shows the arrival time function. To better understand this equation, suppose we have a 1-D region and its boundary is moving with speed F. We know that the differential time passed is equal to speed times the time elapsed, i.e.

$$dx = FdT \quad \Rightarrow \quad 1 = F\frac{dT}{dx} \tag{2}$$

So T will show arrival time for each point. This way could be extended to higher dimension. Now if we implement this method, we can find a curve at different time by just plotting the contours at different times. The problem with using this method for finding boundary, is that to solve this boundary value problem, F should have single value, either positive or negative, otherwise we cannot evolve the boundary, since the might be points that have multiple values (The boundary could pass through some points several times).

The other procedure, that we can take is to solve initial value problem of

$$\phi_t + F|\nabla\phi| = 0 \qquad \Gamma(t) = \{(x, y)|\phi(x, y, t) = 0\} \tag{3}$$

where zero-contour of ϕ is our surface. In this method, we do not require that our value for F to be single valued. In this problem set we will solve this equation and use different velocity functions F to solve different problems.

Problem Definition

In this problem set, we will first use marker particle method to solve the interface propogation problem. In this method, we first parametrize our boundart curve with $0 \leq s < S$. Then we will evolve points on this curve with time and follow these points. These points represent the bounary. We will see that this method is no good since it cannot entropy condition.

In this HW, we will propagate a cosine curved interface, and we will see that after the sharp corner developed through the simulation, the derivative gets ambiguous and this will destroy our solution.

Then We will solve the same interface propagation using level set method. In level set method, we solve the initial value partial differential equation of

$$\phi_t + F|\nabla\phi| = 0, \qquad \Gamma = \{(x,y)|\phi(x,y,0) = 0\} \tag{4}$$

where $\gamma(t) = \{(x,y)|\phi(x,y,t) = 0\}$ is our interface. We will first solve the boundary propagation using this method and then try different values for F to check what happens to the boundary.

In the next part, we will first use Gaussian method for a noise cancelling problem. In Gaussian noise cancelling, we solve

$$I_t = I_{xx} + I_{yy} \tag{5}$$

where $I(x,y)$ is the intensity of (x,y) point of our image. Then we will use boundary propagation method to shrink the noise into a point and then disappear them. We will use a speed function that is dependent on curvature. Because little noises have larger curvature, they will shrink faster and disappear.

Then we will try to segment a part of a picture. The way that it works is that, first we define a boundary propagating problem for ϕ. We draw a little circle in a part of image that we want to segment out. The initial boundary propagating problem that we are solving is

$$\phi_t + (\frac{1}{1+|\nabla I|})|\nabla\phi| = 0 \tag{6}$$

where I is the intensity of the picture. So, the curve that is described by $\phi(x,y,t) = 0$ is gonna start to grow, untill we have a sharp change in intensity. At this point, $|\nabla I|$ gets a very large value, so the speed becomes near zero. So this way we could capture the boundary of that part of our image.

Implementation

Marker Particle Method: To implement the marker particle method, we first discretize the boundary curve using some parametre calles s, such that as s goes from 0 to S, we go along the boundary curve one time. The direction is such that the inner surface is on our left. Then using this parametrization, we find new values for x_i and y_i. Suppose our speed function is $F(\kappa) = 1 + \epsilon\kappa$. We update each point using the following scheme

$$(x_i^{n+1}, y_i^{n+1}) = (x_i^n, y_i^n) + \Delta t F(\kappa_i^n)\frac{(y_{i+1}^n - y_{i-1}^n, -(x_{i+1}^n - x_{i-1}^n))}{((y_{i+1}^n - y_{i-1}^n)^2 + (x_{i+1}^n - x_{i-1}^n)^2)^{1/2}} \tag{7}$$

where

$$\kappa_i^n = 4\frac{(y_{i+1}^n - 2y_i^n + y_{i-1}^n)(x_{i+1}^n - x_{i-1}^n) + (x_{i+1}^n - 2x_i^n + x_{i-1}^n)(y_{i+1}^n - y_{i-1}^n)}{((y_{i+1}^n - y_{i-1}^n)^2 + (x_{i+1}^n - x_{i-1}^n)^2)^{3/2}} \tag{8}$$

In the above equation, we suppose that Δs is same between different poinst on the curve.

Level Set Method: The equation that we are solving in here is

$$\phi_t + F|\nabla\phi| = 0 \tag{9}$$

where zero level contours is our boundary. We first initialize the value of ϕ such that, value of ϕ at each point is the distance of that point from the boundary. And also inside of the domain we have negative value and outside we have positive value for this distance. In other words, we ϕ is the signed distance function.

Speed function $F = A + \epsilon\kappa$, is composed of two parts. One part is (A) which is some known function and the other part $(\epsilon\kappa)$ has curvature in it. We use centeral difference for the curvature part and upwind method for the function part of the F. Our scheme is as

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} - \Delta t\ [\max(A_{i,j},0)\nabla^{+} + \min(A_{i,j},0)\nabla^{-} - \epsilon\kappa_{ij}\nabla^{0}] \tag{10}$$

where

$$\nabla^{+} = [\max(D_{i,j}^{-x},0)^2 + \min(D_{i,j}^{-x},0)^2 \tag{11}$$

$$\max(D_{i,j}^{-y},0)^2 + \min(D_{i,j}^{-y},0)^2]^{1/2} \tag{12}$$

$$\nabla^{0} = [(D_{ij}^{0-x})^2 + (D_{ij}^{0-y})^2]^{1/2} \tag{13}$$

Here we have used the standard shorthand notation, where D_{ij}^{+x} means forward difference value in x direction (e.g. $D_{ij}^{+x} = \frac{\phi_{i+1,j}^{n} - \phi_{i,j}^{n}}{\Delta x}$). For the curvature part, we also have

$$\kappa = \nabla.\frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_y^2\phi_{xx} - \phi_x\phi_y\phi_{xy} + \phi_x^2\phi_{yy}}{(\phi_x^2 + \phi_y^2)^{3/2}} \tag{14}$$

where all the terms are computed with centeral difference.

Problem

1. Part One:
   Take the cosine curve $(s, cos(s))$, s between 0 and 1. Let the speed function normal to the curve be $F = 1 - \epsilon\kappa$, where κ is the curvature. Wall off the sides and bottom are,

   (a) for s between 0 and 1, $x(s) = s\,, y(s) = cos(2\pi s)$

   (b) for s between 1 and 2, $x(s) = 1\,, y(s) = 1 - 3(s-1)$

   (c) for s between 2 and 3, $x(s) = 3 - s\,, y(s) = -2$

   (d) for s between 3 and 4, $x(s) = 0\,, y(s) = -2 + 3(s-3)$

   Using a marker particle scheme, let $\epsilon = .1, .01, .001, .0001$, and 0.0. Solve for the evolution of this front, and see if you can keep it alive. Using a level set method, repeat the calculation. See what happens.

2. Part Two:
   Start with an initial simple closed curve, something worse than a circle or ellipse. Find the signed distance function on a mesh to this thing. Write a level set method for flow $F = A - \epsilon * \kappa$.

   (a) For $A = 0$, show that your "arbitrary" closed curve shrinks smoothly to a circle and then dissappears.

   (b) For $\epsilon = 0$, and $A = 1$, run your code and see what happens.

   (c) For $\epsilon = 0$, and $A = -1$, run your code and see what happens. (Comment: be very careful about signs, upwinding, etc. so that your curve goes in the opposite direction.)

3. Part Three:
   Find a gray scale image on the web, say 256x256. Run three experiments: Change $1\%, 10\%, 20\%$ of all the pixels to a different random value between 0 and 255. For each of these noisy images:

   (a) Use Gaussian smoothing on the noisy image to see what happens.

   (b) Run curvature flow on the noisy image to see what happens.

4. Part Four:
   Build an image, with gray scale values, say 256x256, in which there is a region you want to segment out. (For example, put a dark disk inside a white background.) Using speed function $F = 1/(1 + |\nabla I|)$, start with an initial front inside the disk, and find its boundary. Run four experiments: Change $0\%, 1\%, 10\%, 20\%$ of all the pixels to a different random value between 0 and 255.

   Resolve the mesh, see what happens to oscillations, sharpness, etc. Figure out a way to assess your error.

Solution

Part One

(a) - In this part we used both particle marker method as described in Implementation part. In the case of Particle Marker method, the results are shown in figures 1 up to 4. As it can be seen in these figures, in the cos part of the boundary, the entropy condition is not satisfied. Since the derivative is not well defined after the sharp (shock) is formed on the boundary.

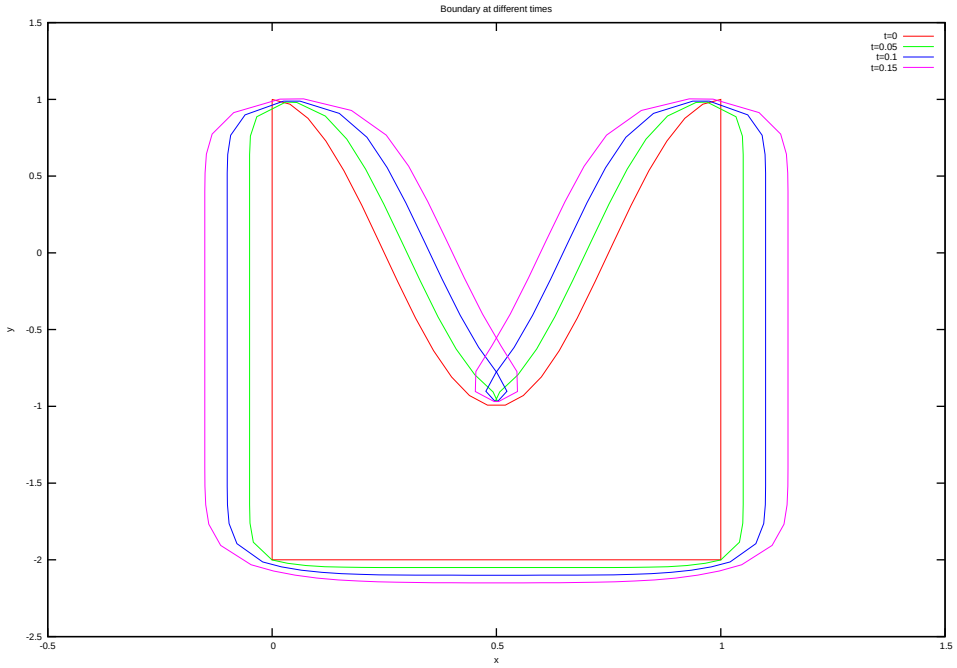

Figure 1: Particle Marker method for $\epsilon = 0.1$ at different times

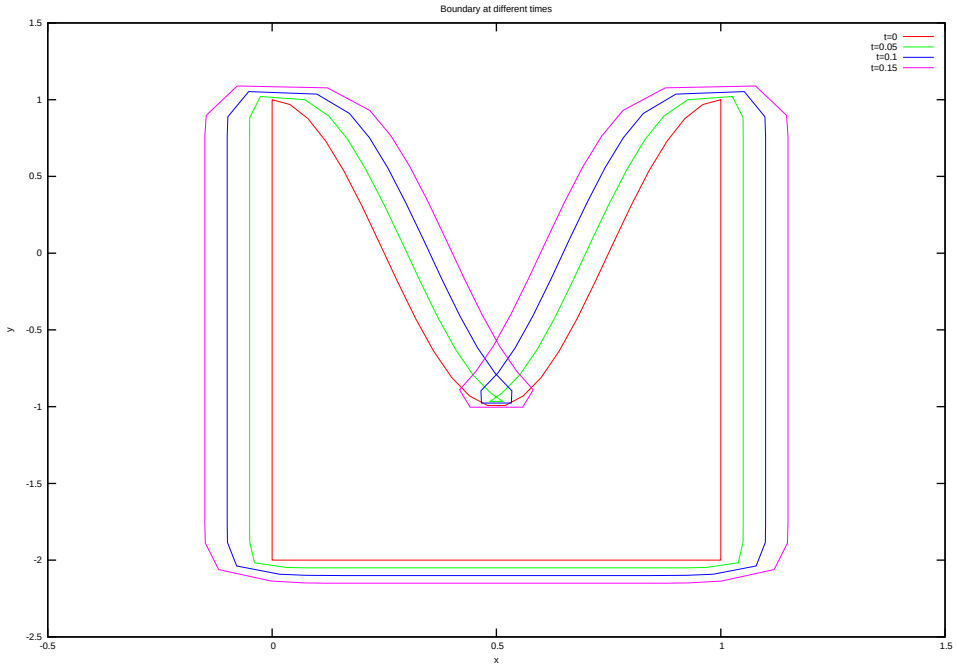

Figure 2: Particle Marker method for $\epsilon = 0.01$ at different times

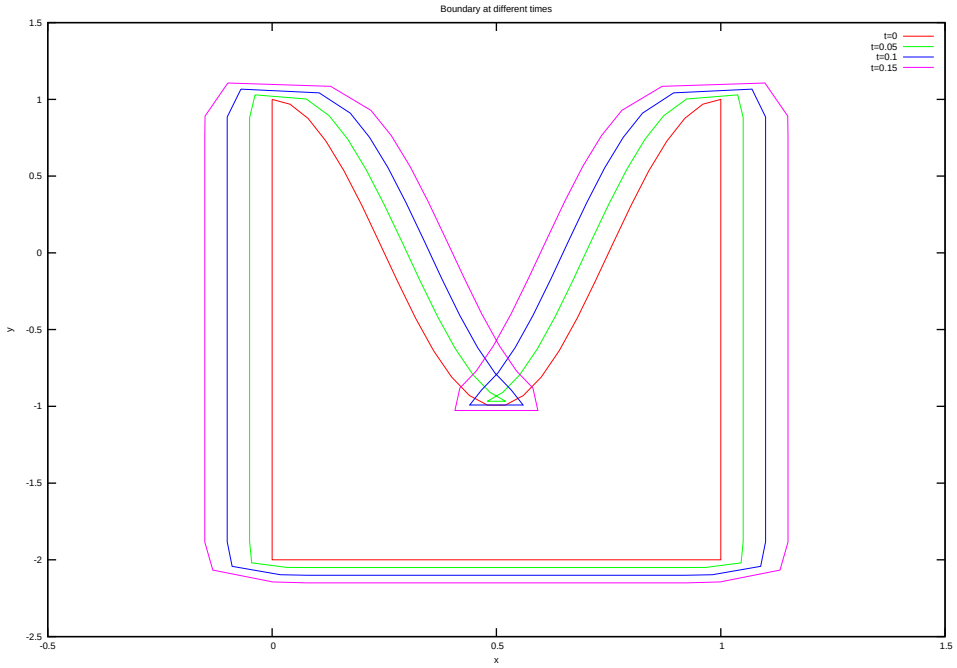

Figure 3: Particle Marker method for $\epsilon = 0.001$ at different times

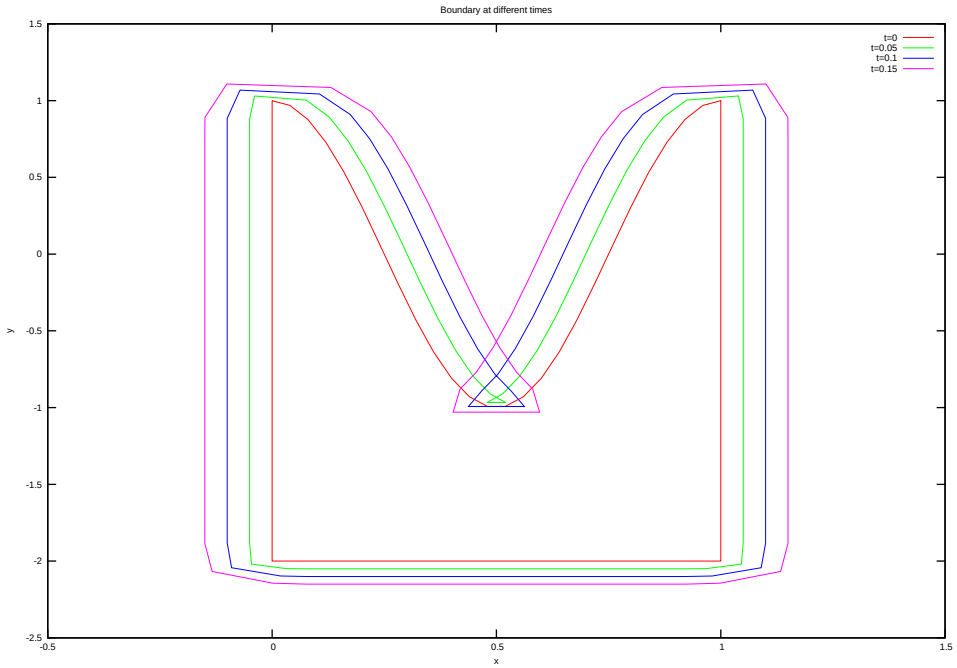

Figure 4: Particle Marker method for $\epsilon = 0.0001$ at different times

(b) - In the second part, we implemented Level Set method. In this scheme we used winding test number to find if we are in the boundary or not. The distance is also found by finding the minimmum distance from all points on the curve. The resulst are shown in

Figures 5 upto 7. As can be seen in these figure, the boundary rematins fine at the sharp corner of cos.

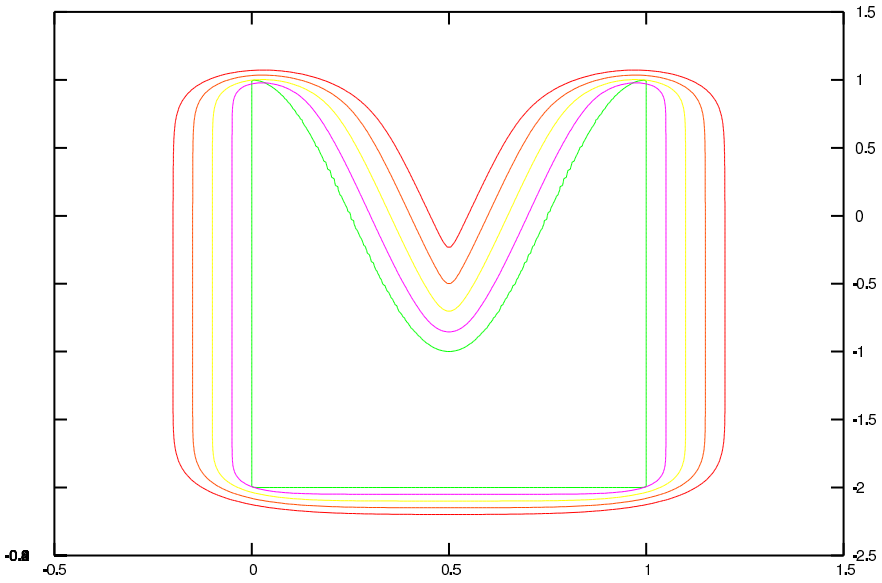

Figure 5: Level Set Method for $\epsilon = 0.1$ at different times

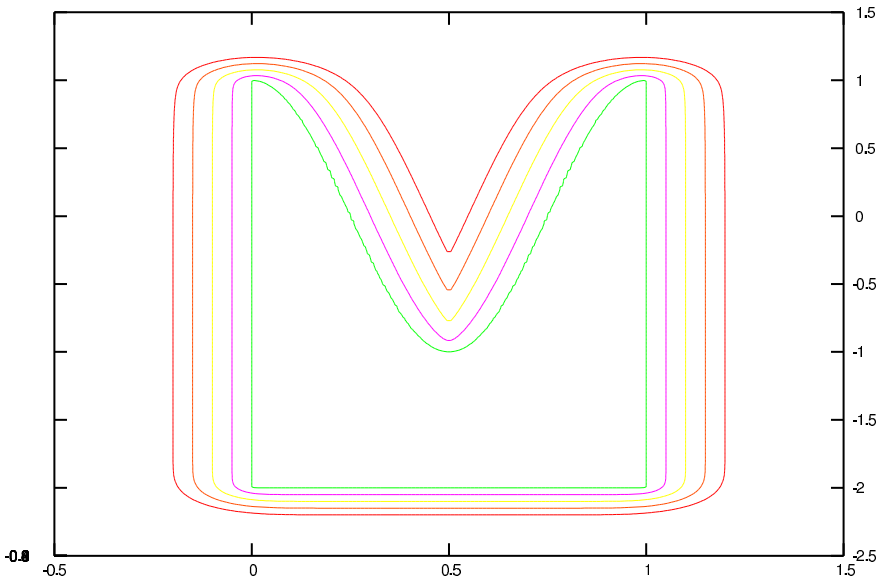

Figure 6: Level Set Method for $\epsilon = 0.01$ at different times

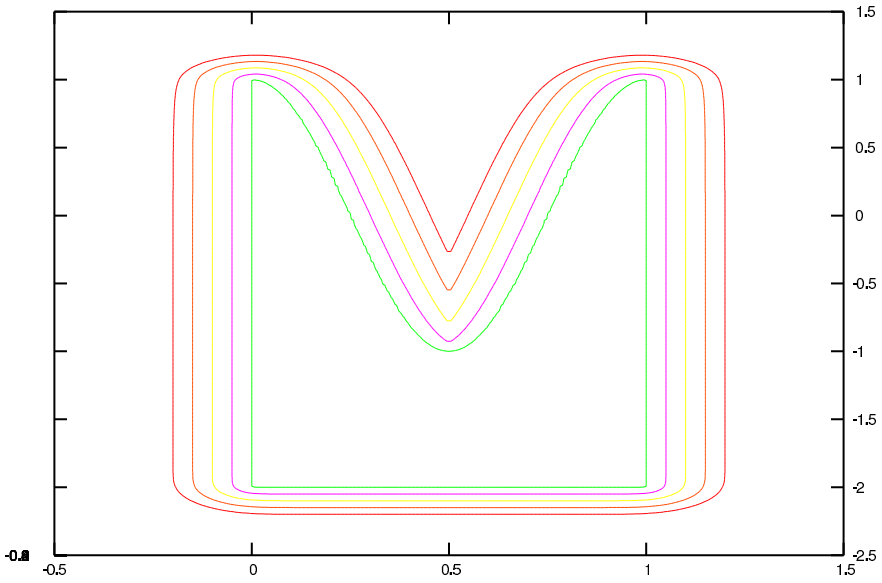

Figure 7: Level Set Method for $\epsilon = 0.001$ at different times

Part Two

In this part, we again use the functions that we wrote for the previous part. We found a heart like curve (worse than elipse or circle). We run this integrator for several time witg speed function depending on curvature to basically confirm that the shape will shrink into circle and then disappear. We also tested two cases of $A = 1$ and $A = -1$. The results are in Figures 8 upto 10. We can see that when we are shrinking the image base on the curvature, the curve shrinks to circle first and then disappear, but when we shrink the image base on just some velocity the image will keep its shape and shrink.

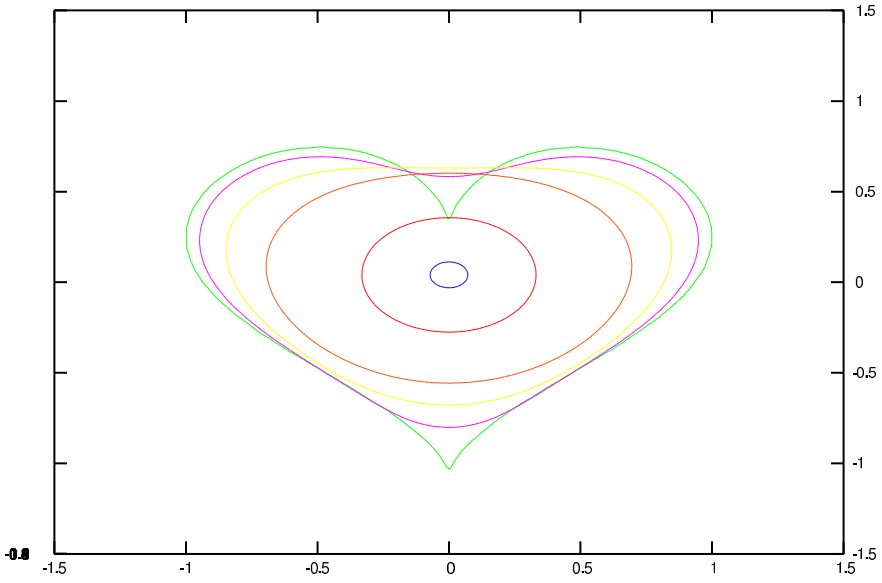

Figure 8: Arbitrary Curve shrinking to a cicle and disappearing

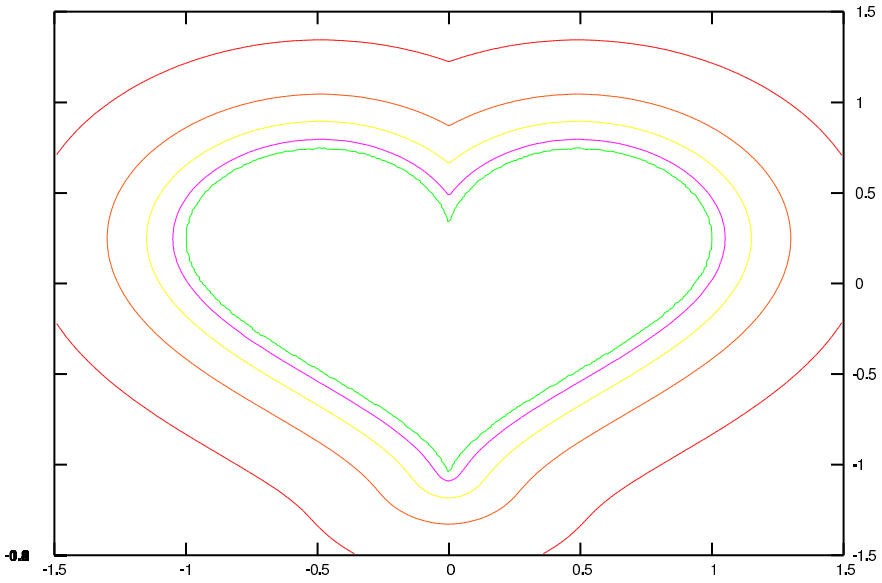

Figure 9: Arbitrary Curve expanding with$A = 1$

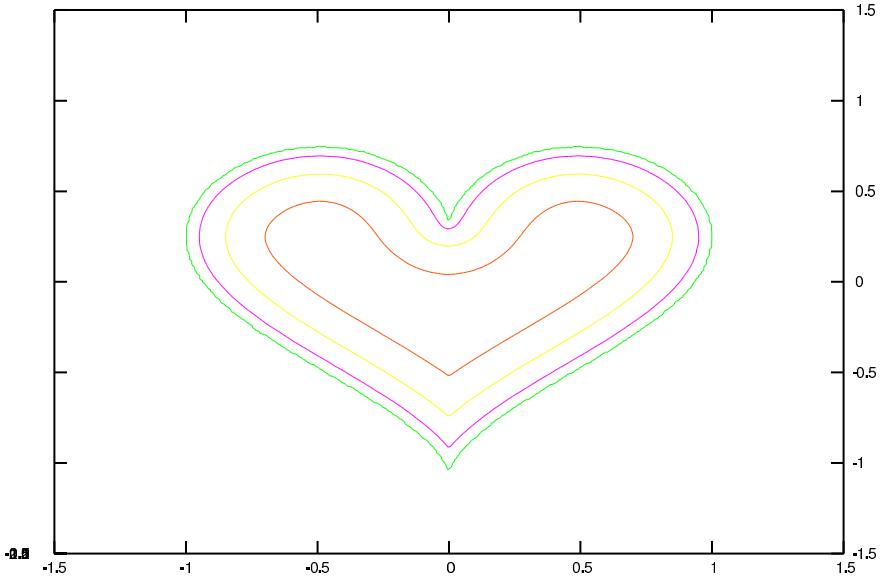

Figure 10: Arbitrary Curve shrinking $A = -1$

Part Three

In this part, we used intensity as ϕ function in our code for previous parts. We wrote the Guassian heat equation for noise cancelling based on central difference. We also implement the level set method to cancel the noises that we put into the picture. Basically, as we use level set or gaussian scheme, the im age loses its sharpness. So we define a function called *renormalization* to renormalize the image. What we do is after we cancelled the noise image, we will set all the cells whose value are greater than 127 to 255, otherwise 0. Results for

Guassian Noise cancelling for different cases in Figures 11 upto 22. For the case of level sets method, results are shown in Figures 24 upto 39.

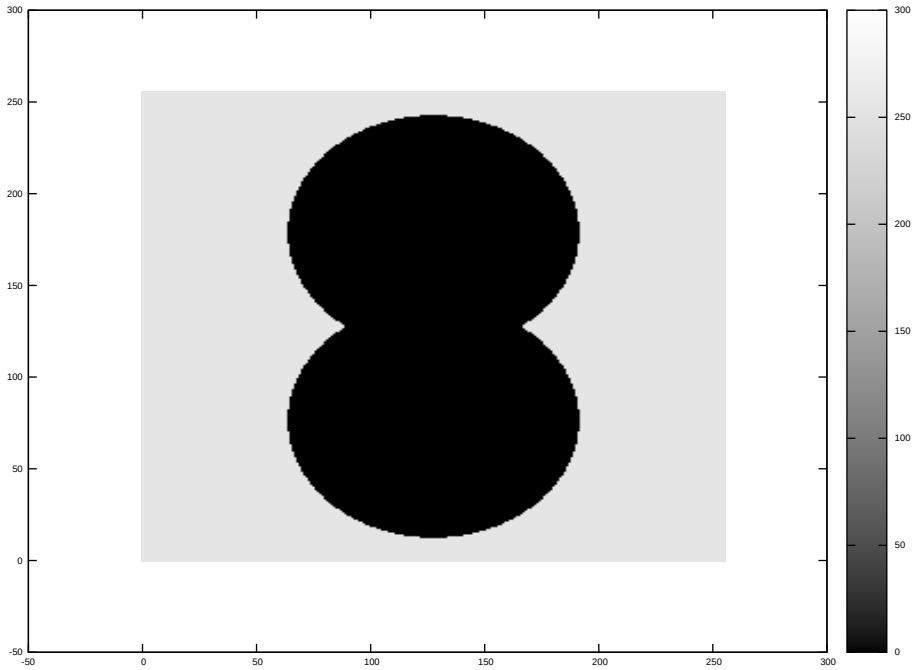

Figure 11: Initial shape

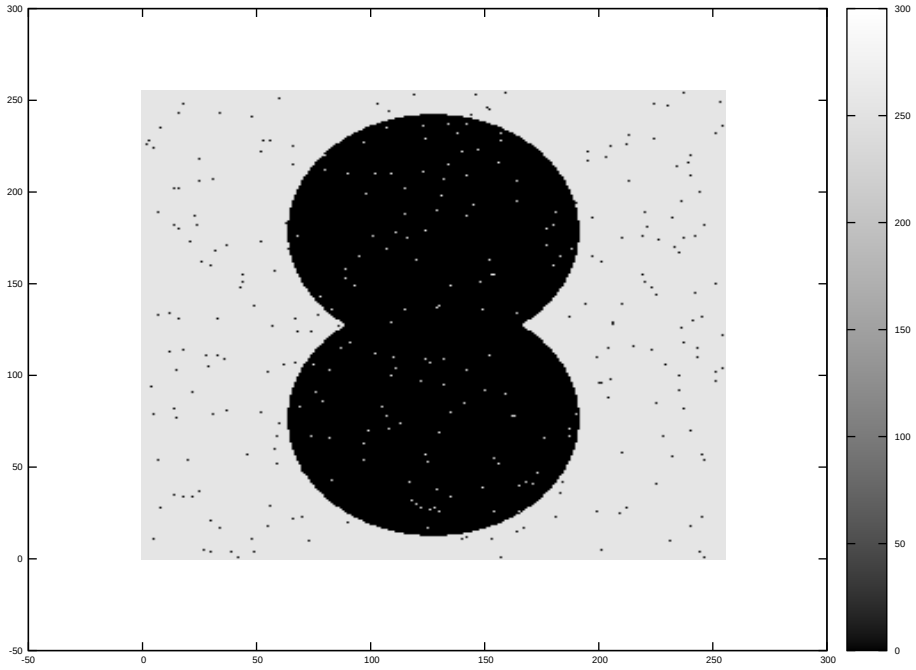

Figure 12: Initial shape after 1% putting noise

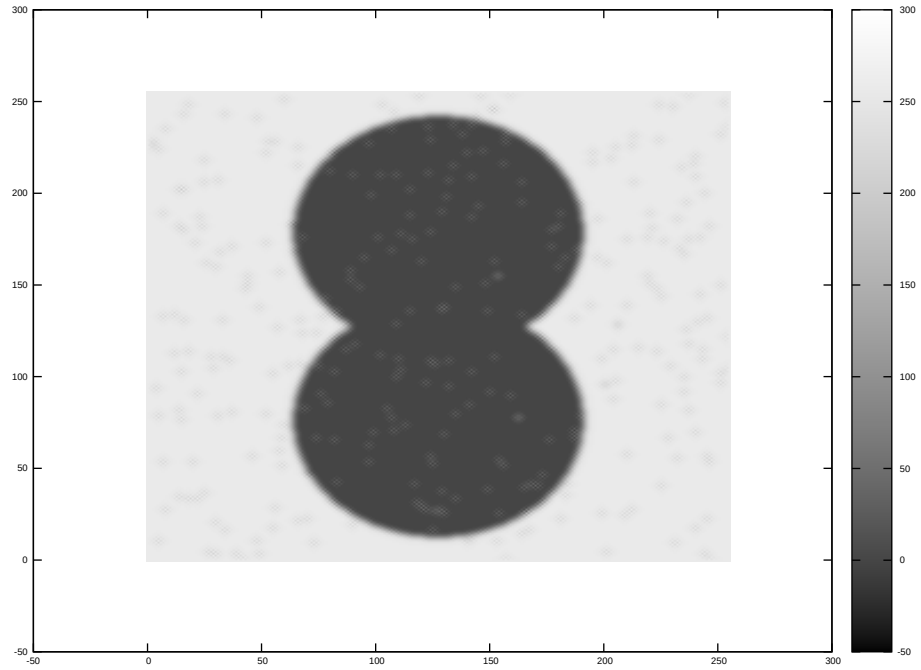

Figure 13: final shape after Guassian Noise cancelling for 1% case

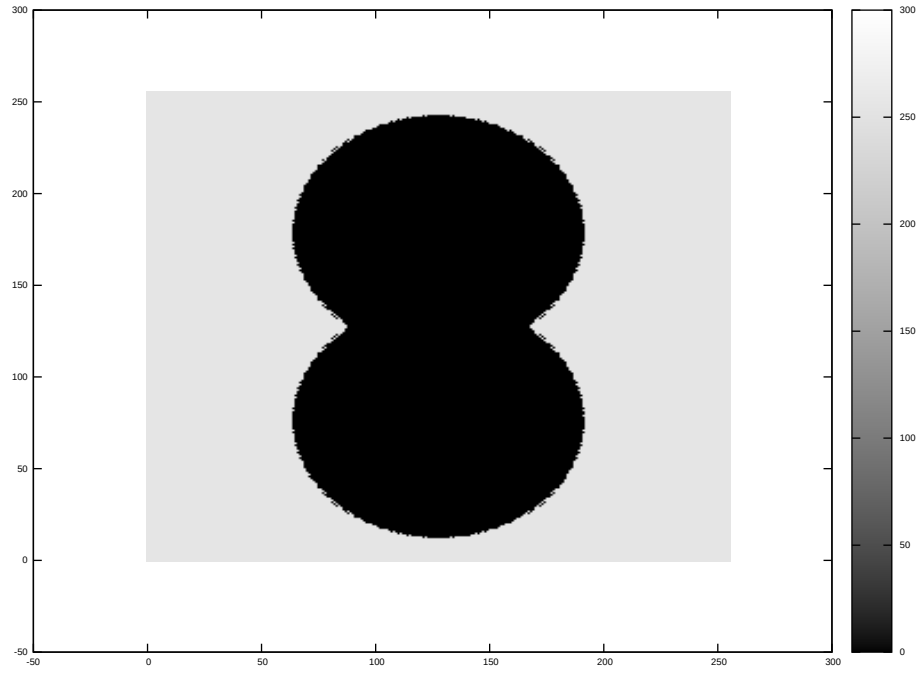

Figure 14: final shape after Guassian Noise cancelling and renormalization for 1% case

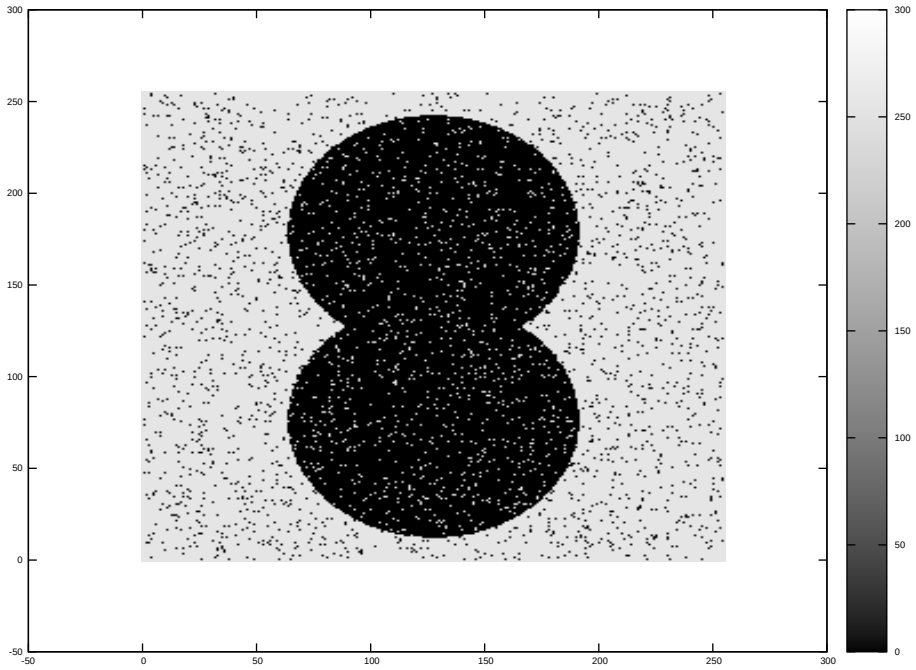

Figure 15: Initial shape after 10% putting noise

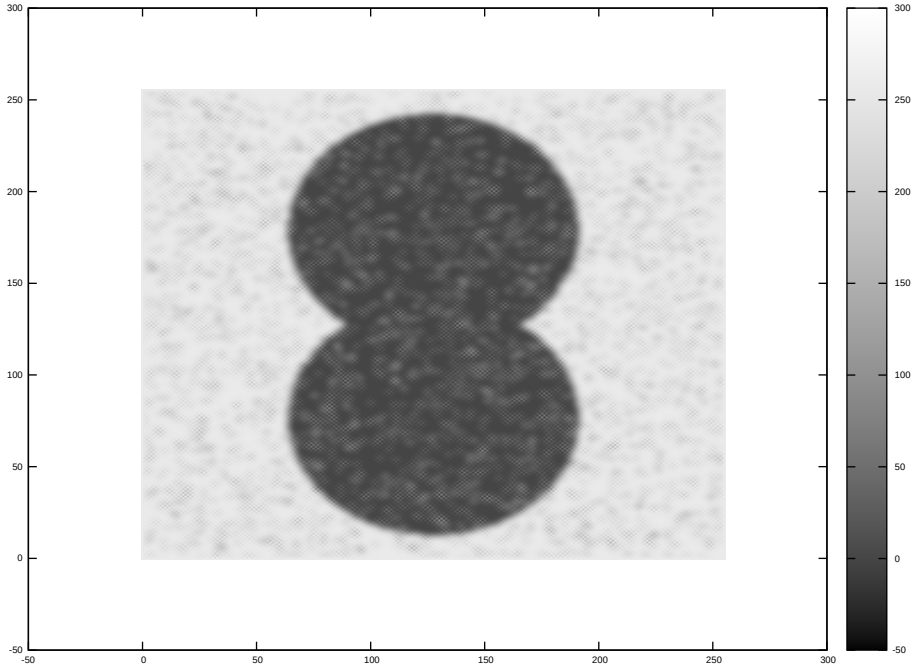

Figure 16: final shape after Guassian Noise cancelling for 10% case

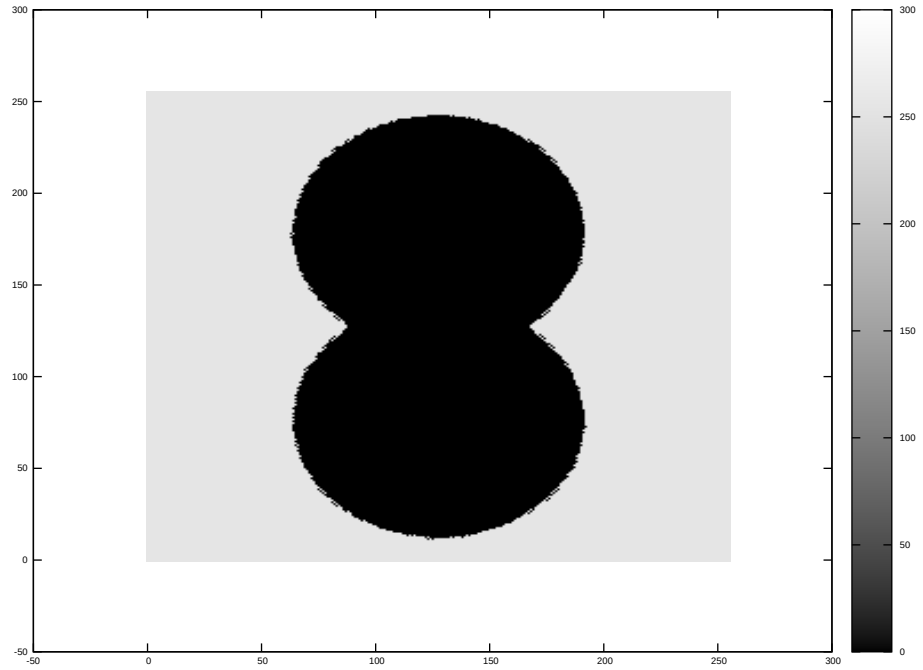

Figure 17: final shape after Guassian Noise cancelling and renormalization for 10% case

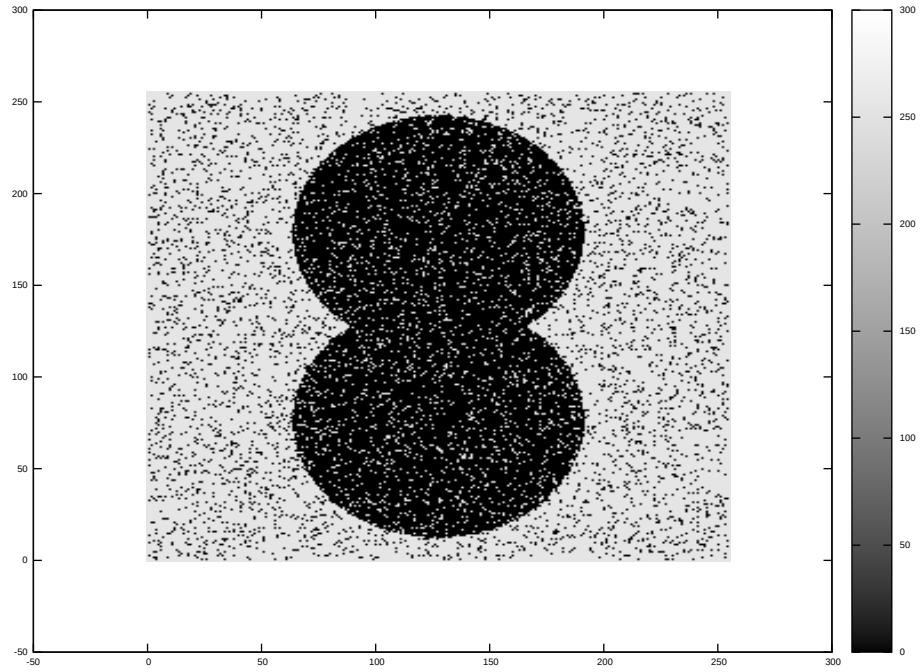

Figure 18: Initial shape after 20% putting noise

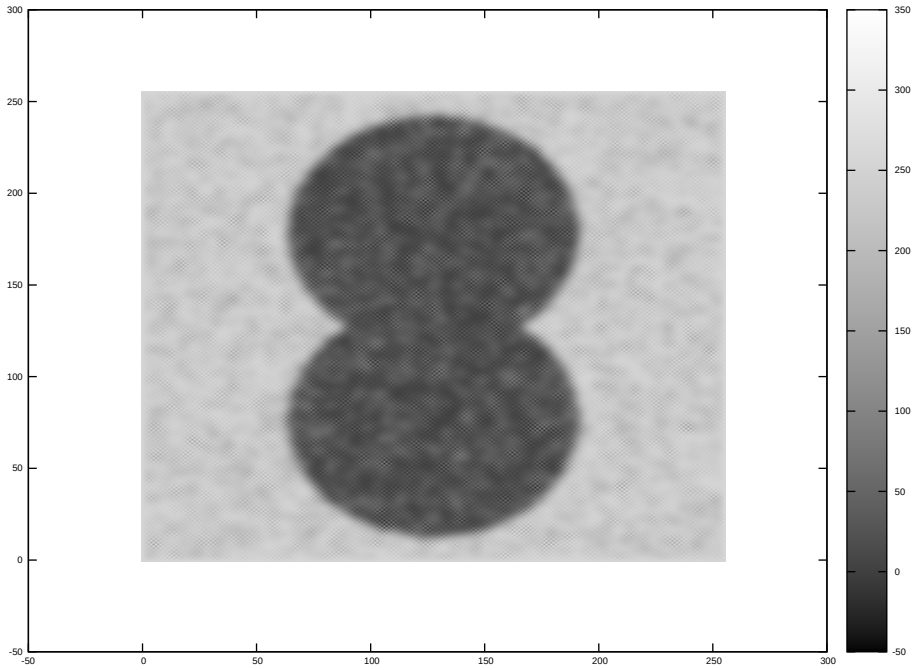

Figure 19: final shape after Guassian Noise cancelling for 20% case

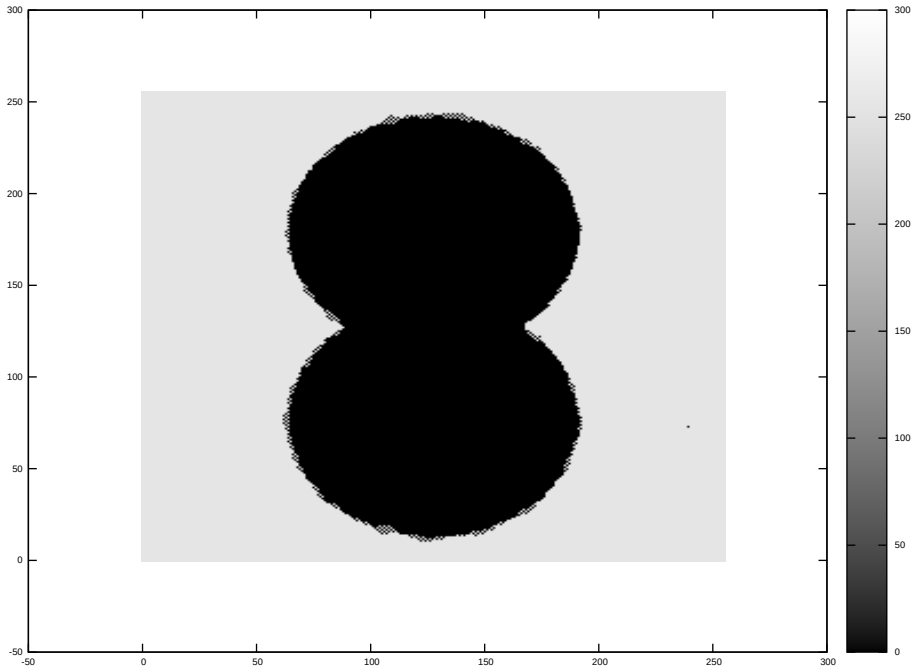

Figure 20: final shape after Guassian Noise cancelling and renormalization for 20% case

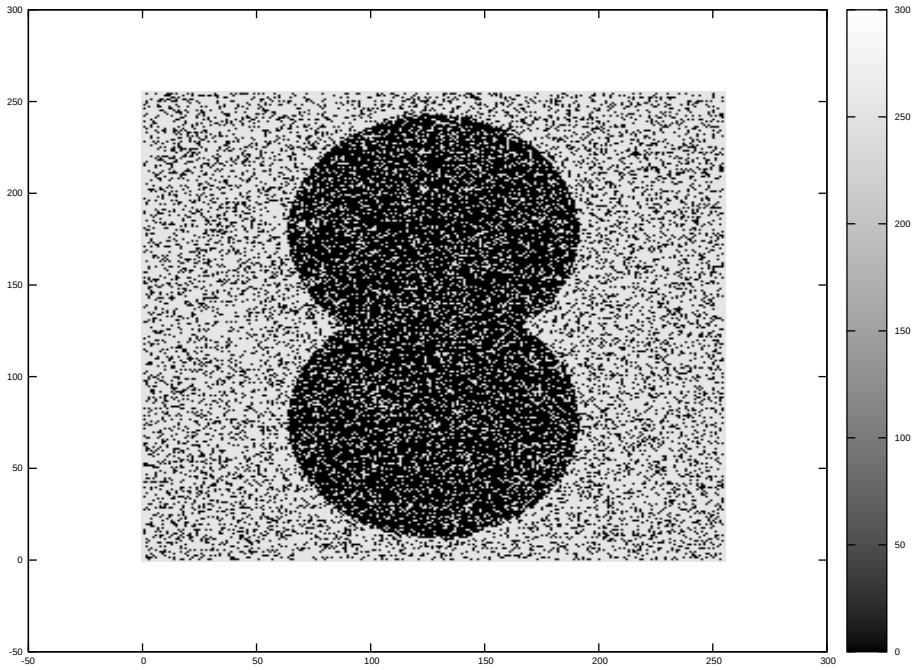

Figure 21: Initial shape after 40% putting noise

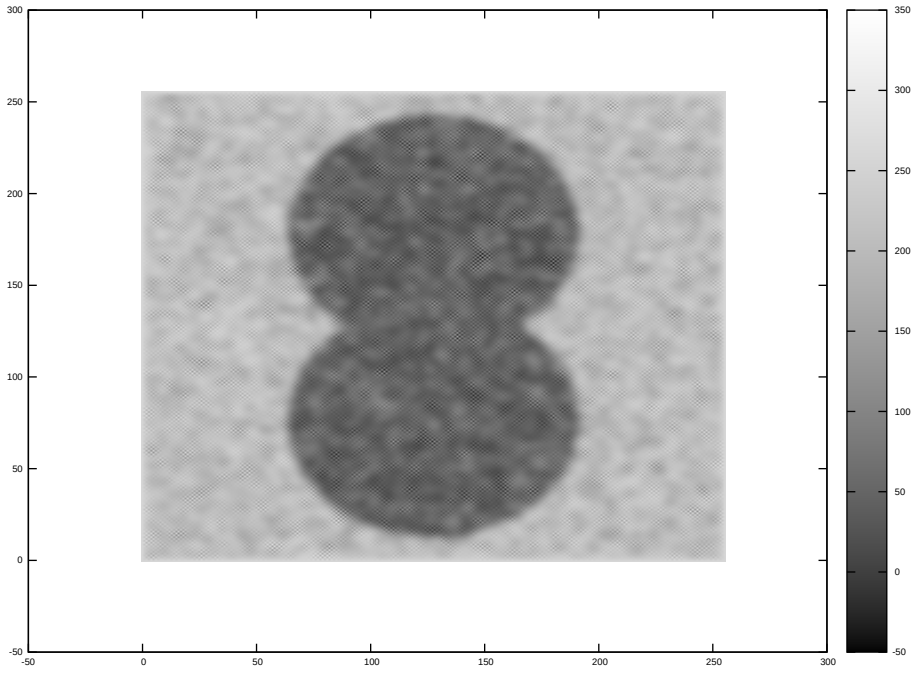

Figure 22: final shape after Guassian Noise cancelling for 40% case

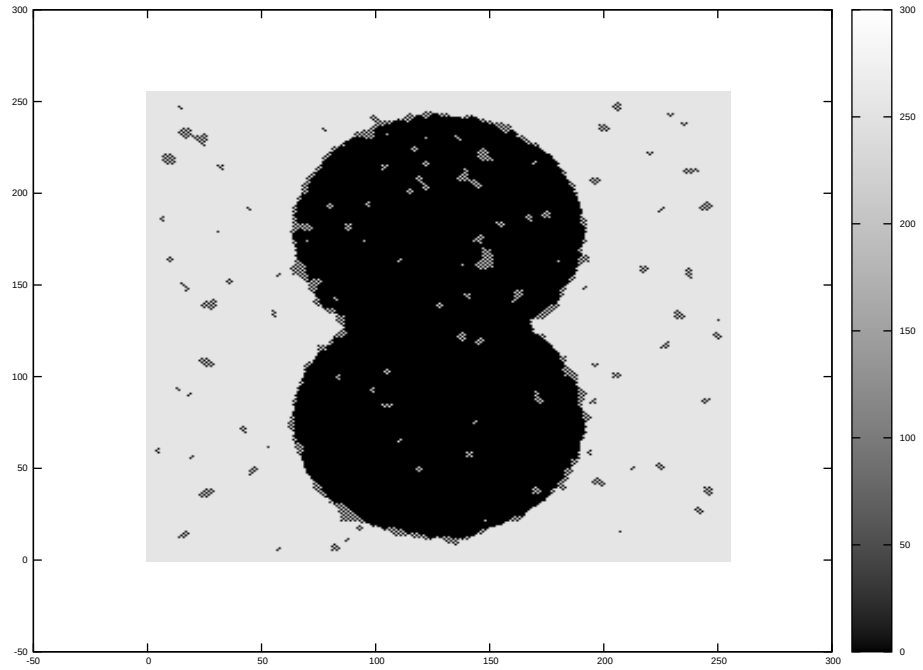

Figure 23: final shape after Guassian Noise cancelling and renormalization for 40% case

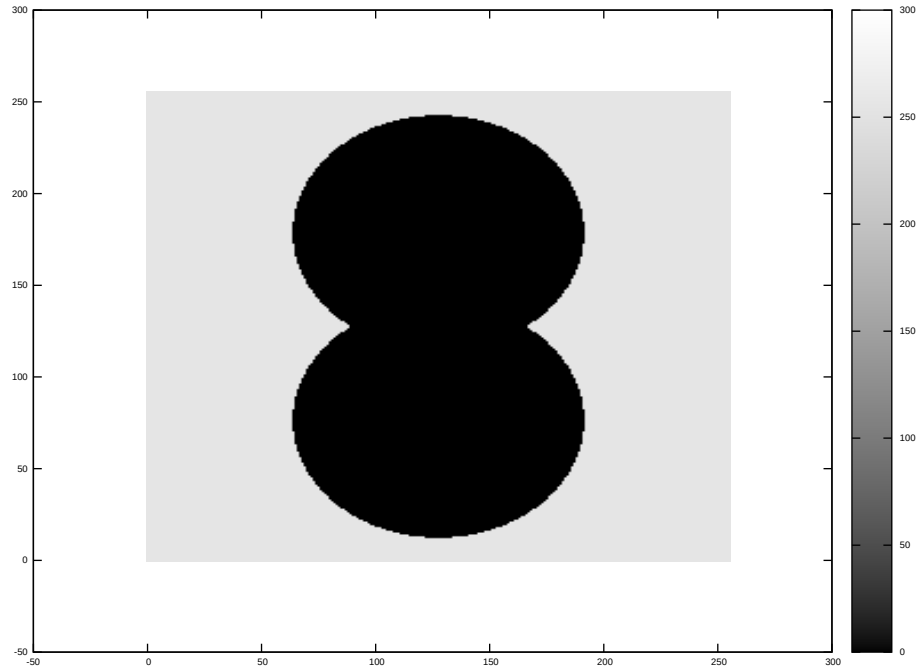

Figure 24: Initial shape

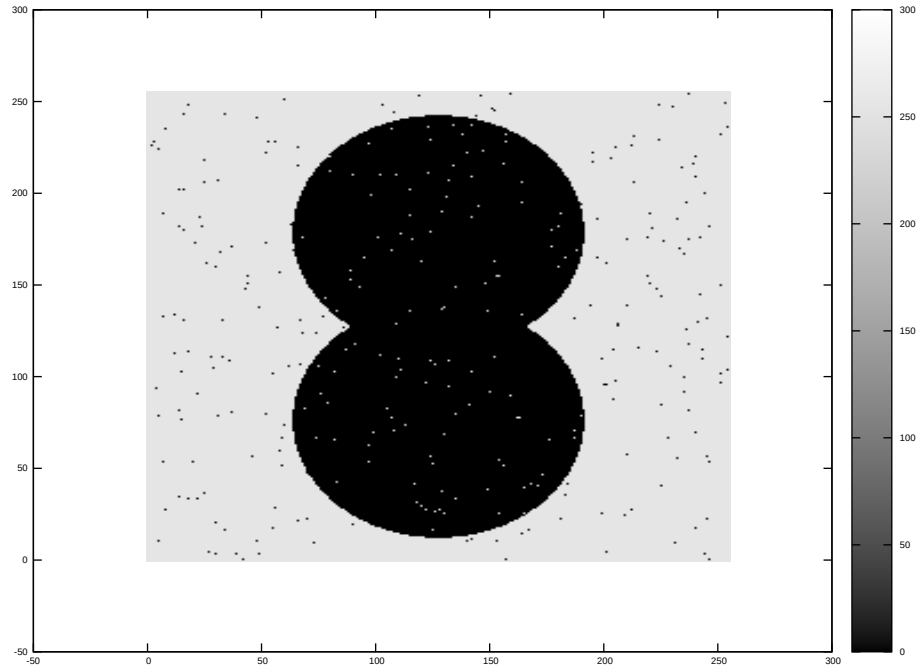

Figure 25: Initial shape after 1% putting noise

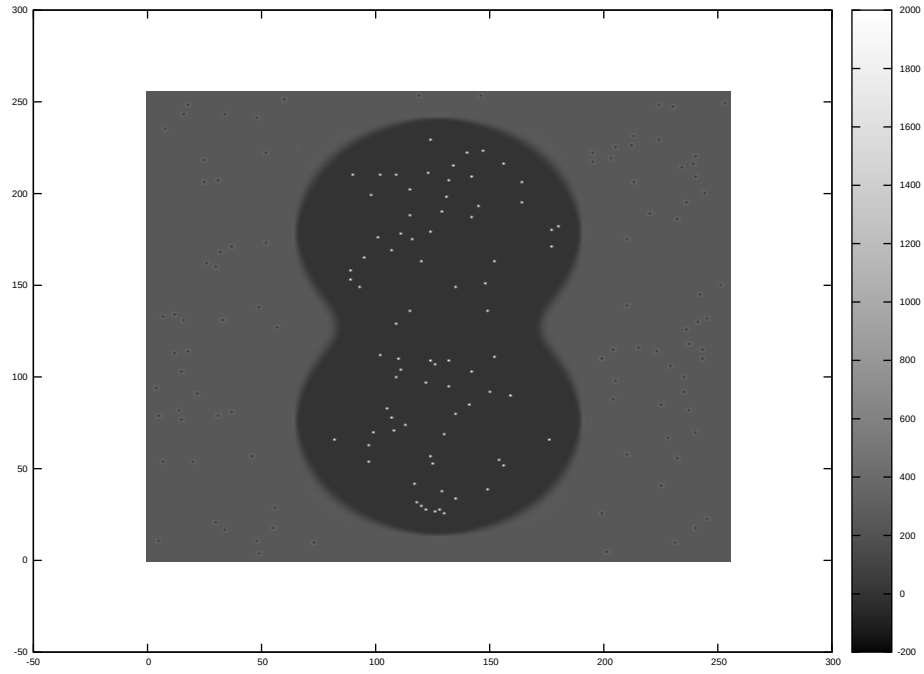

Figure 26: final shape after Level-Set Method Noise cancelling for 1% case

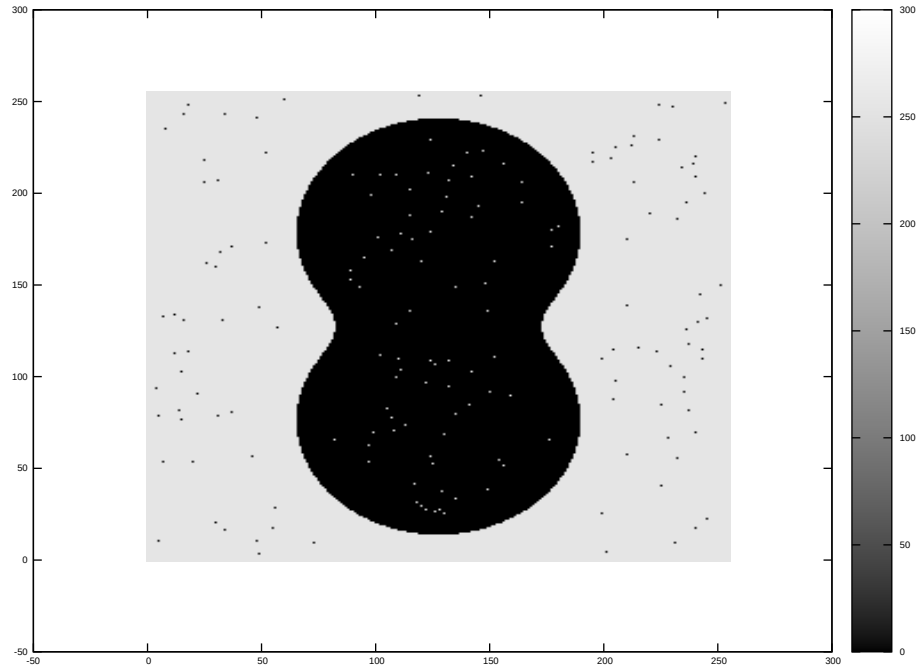

Figure 27: final shape after Level-Set Method cancelling and renormalization for 1% case

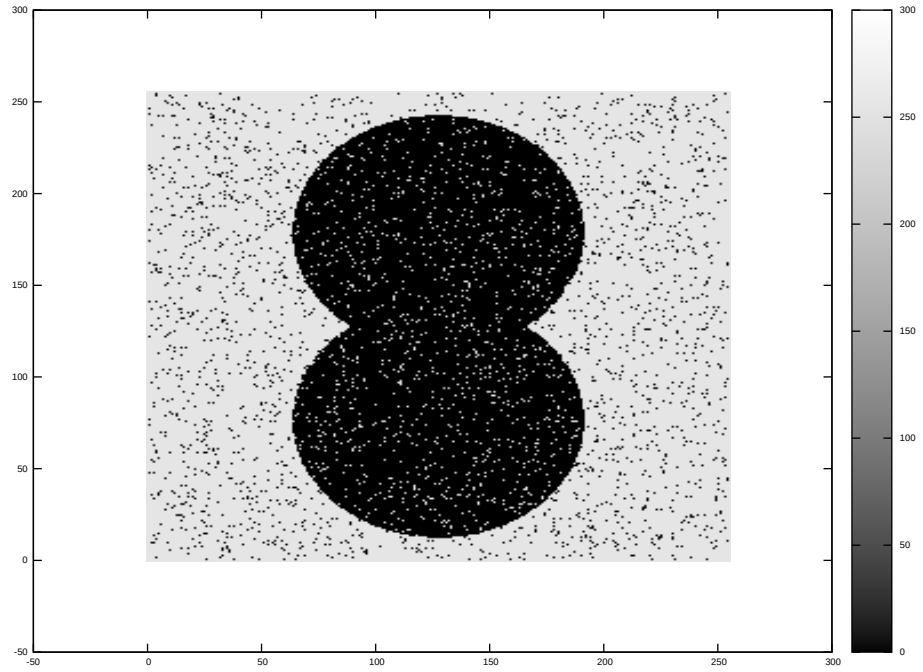

Figure 28: Initial shape after 10% putting noise

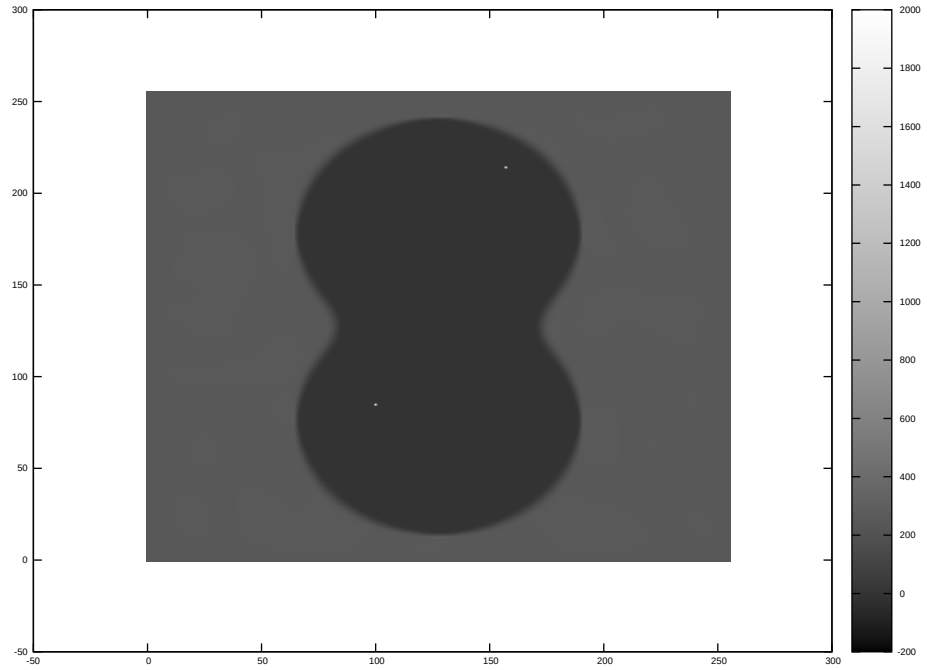

Figure 29: final shape after Level-Set Method Noise cancelling for 10% case

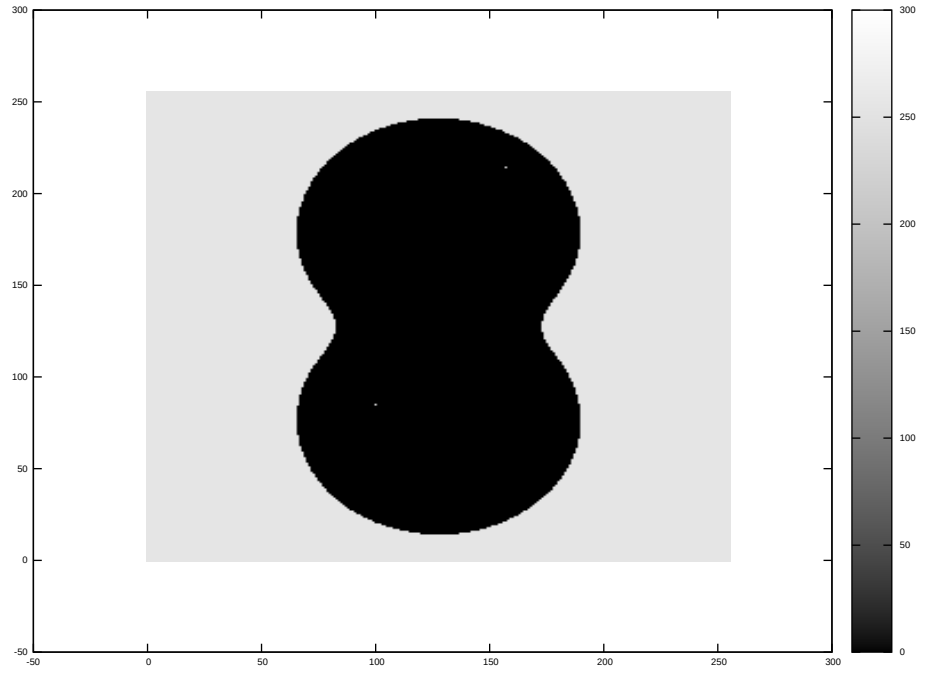

Figure 30: final shape after Level-Set Method cancelling and renormalization for 10% case

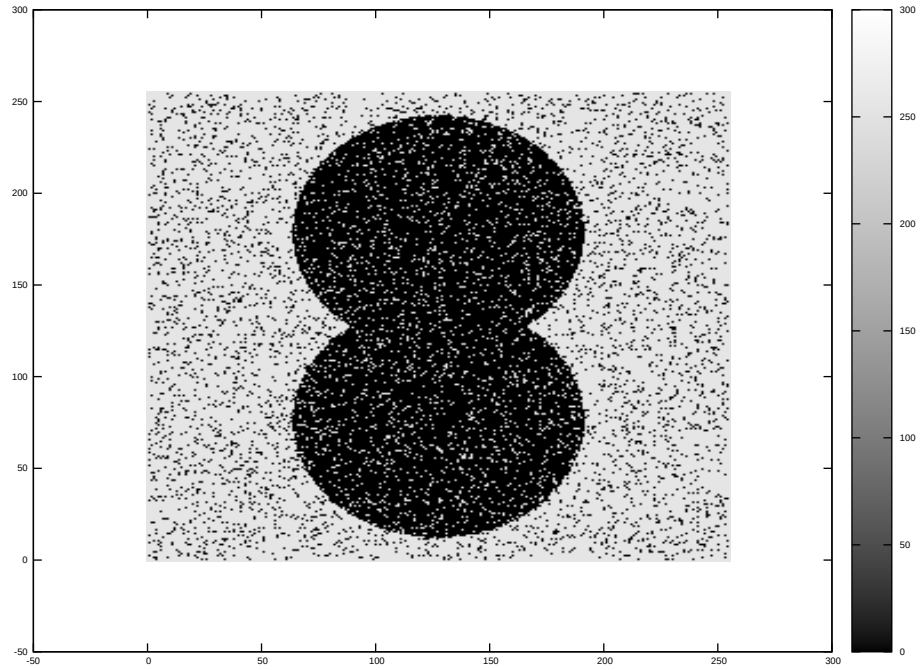

Figure 31: Initial shape after 20% putting noise

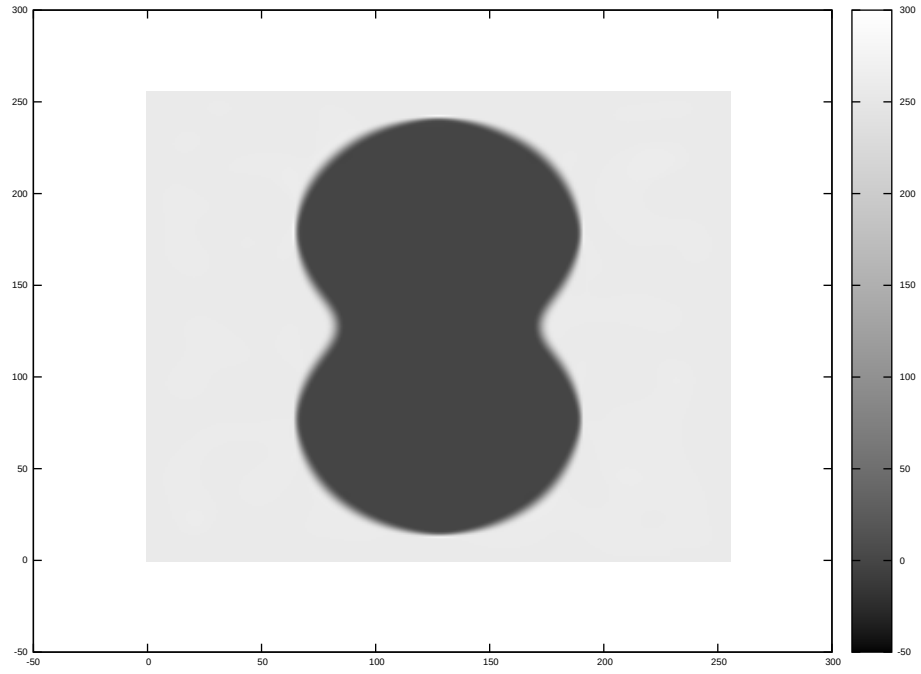

Figure 32: final shape after Level-Set Method Noise cancelling for 20% case

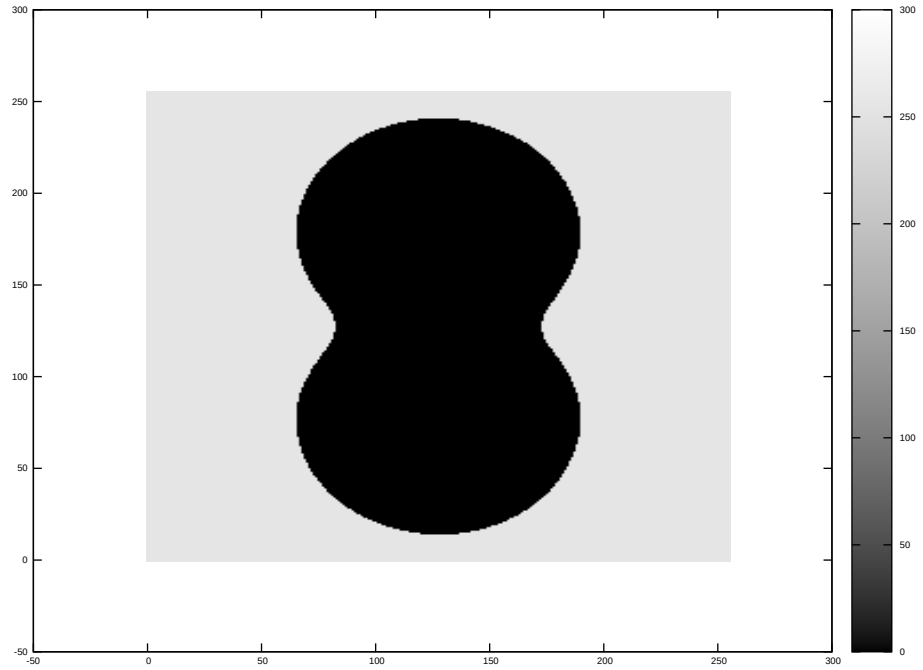

Figure 33: final shape after Level-Set Method cancelling and renormalization for 20% case

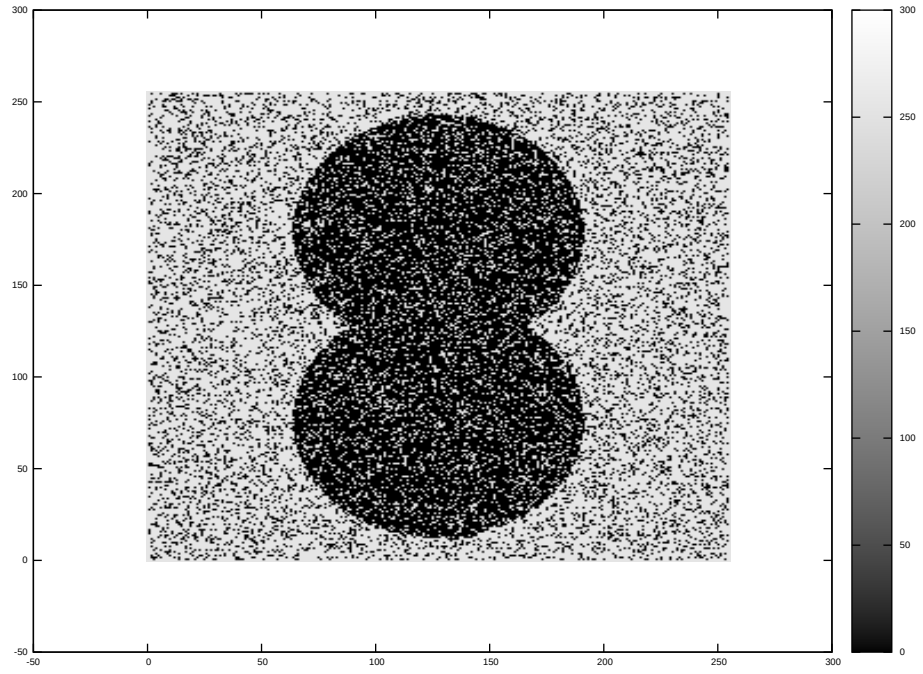

Figure 34: Initial shape after 40% putting noise

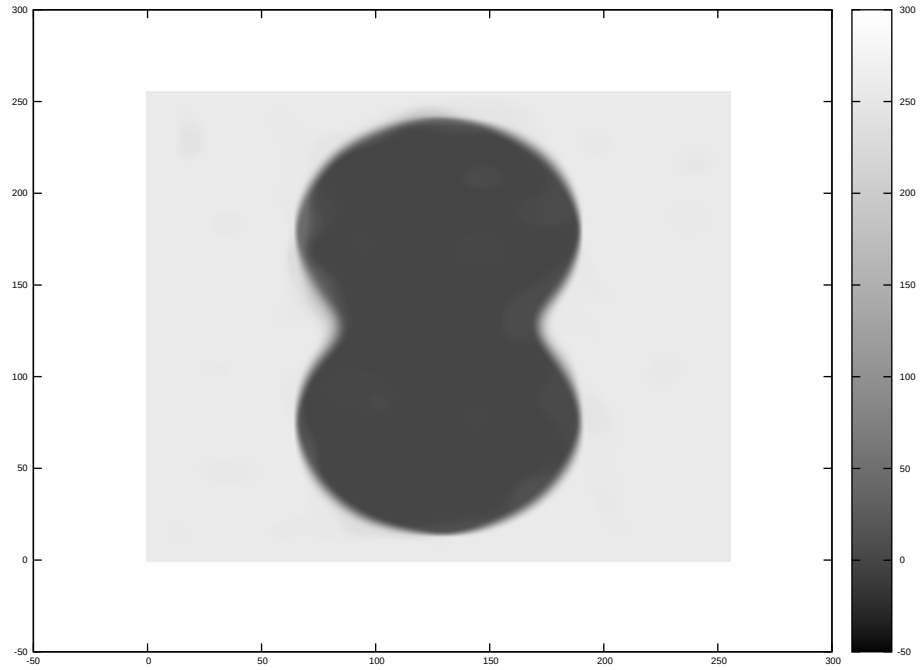

Figure 35: final shape after Level-Set Method Noise cancelling for 40% case

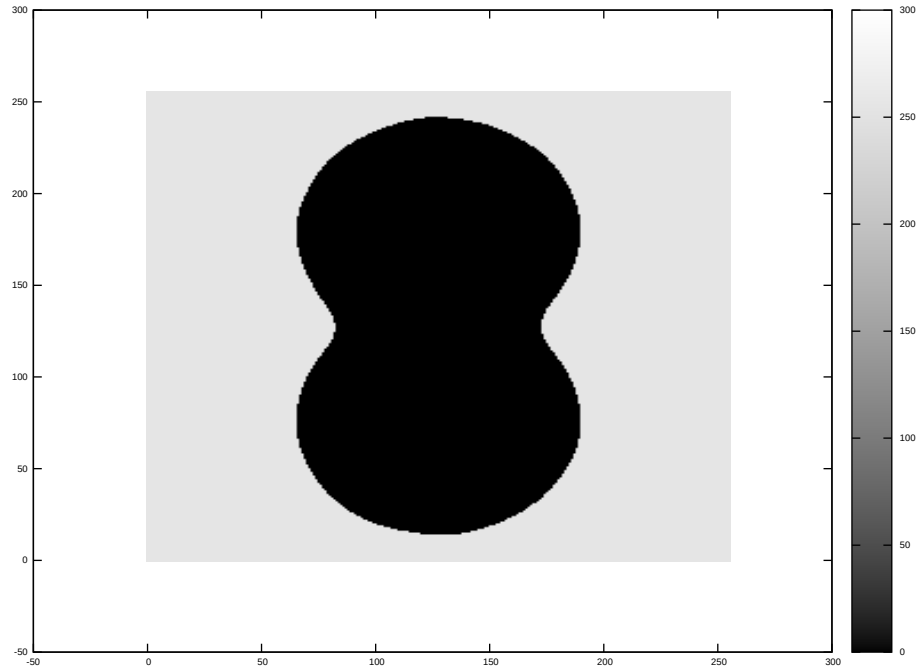

Figure 36: final shape after Level-Set Method cancelling and renormalization for 40% case

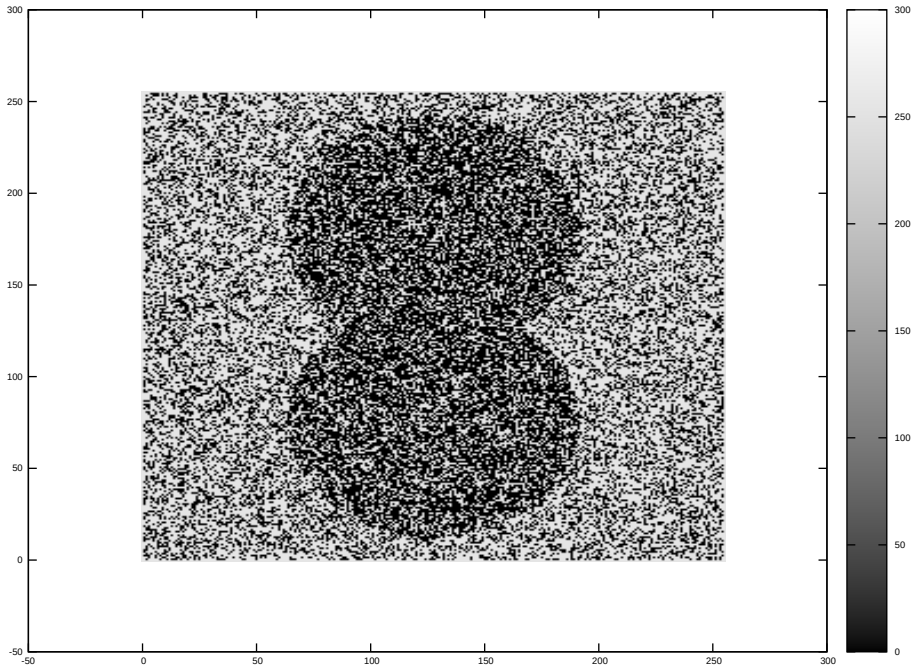

Figure 37: Initial shape after 70% putting noise

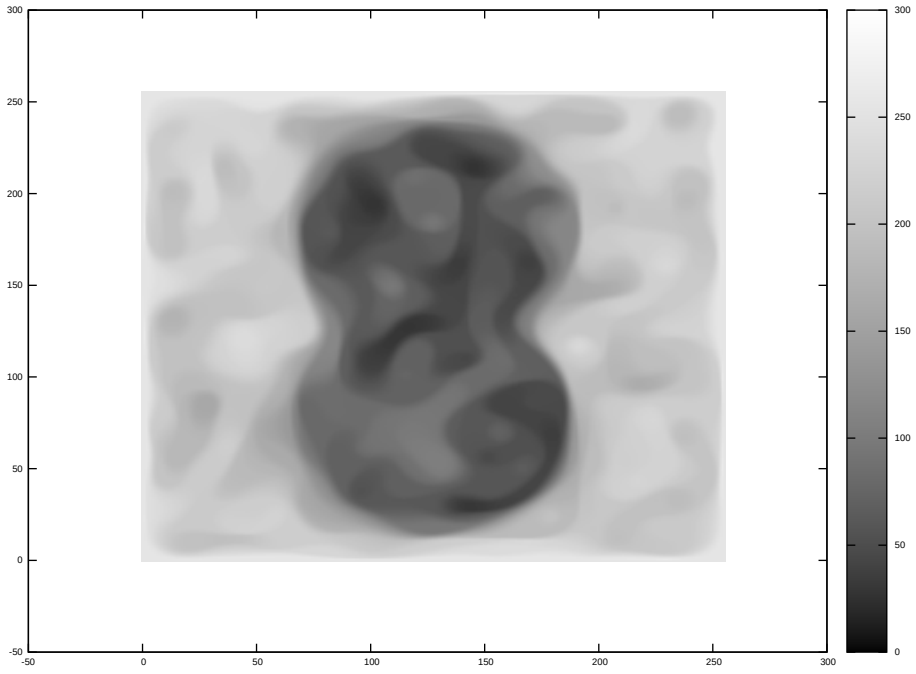

Figure 38: final shape after Level-Set Method Noise cancelling for 40% case

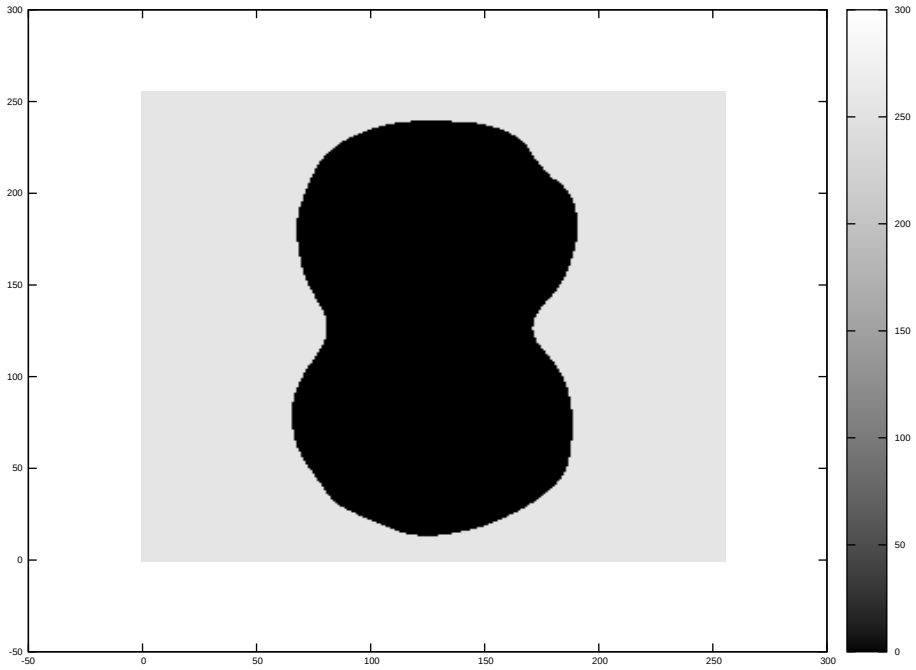

Figure 39: final shape after Level-Set Method cancelling and renormalization for 70% case

Part Four

In this part, we use the speed function base on the intensity of the image that we want to segment out.We put a circular curve in the middle of the image to grow to the boundary of the shape. The results is in the following figures. As it can be seen in Figure 47, with 20% noise, the method cannot find the boundary. Also as it can be seen in Figure 45, the image with 1% noise the boundary is founded, but still the method find the noises as the boundary, which is not true.

So basically, I put some noise in the case of 20% noise, then I cancelled these noise using Level-Set methods and then tried to find the boudnary. This way we could easily find the boudnary. This result can be seen in Figure

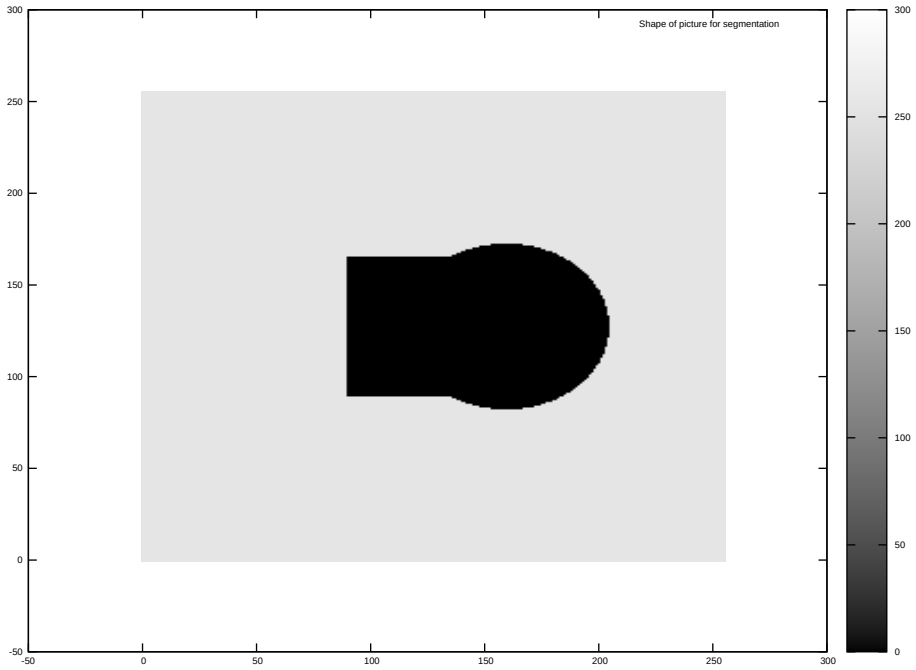

Figure 40: Image that we want to semgent out

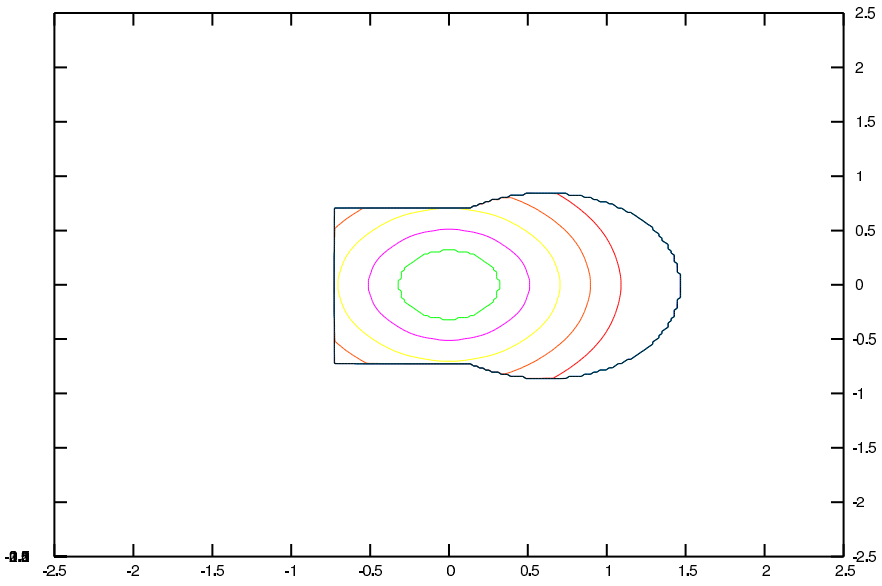

Figure 41: Segmentation using Level-Set method

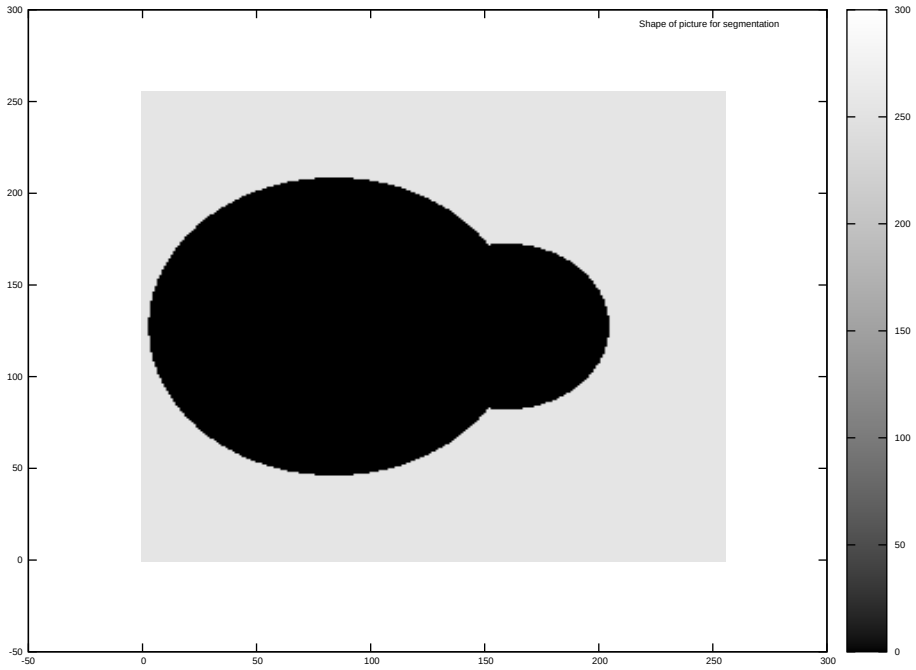

Figure 42: Image that we want to semgent out

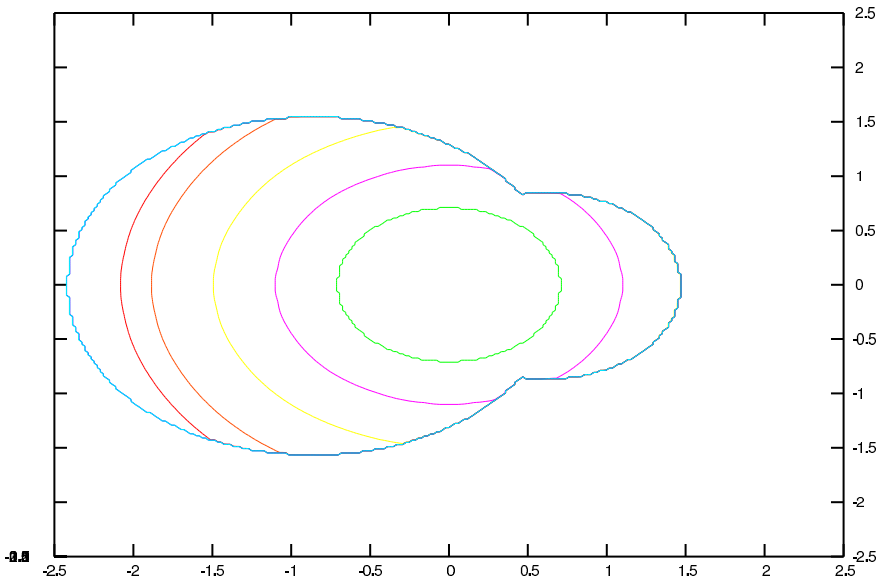

Figure 43: Segmentation using Level-Set method

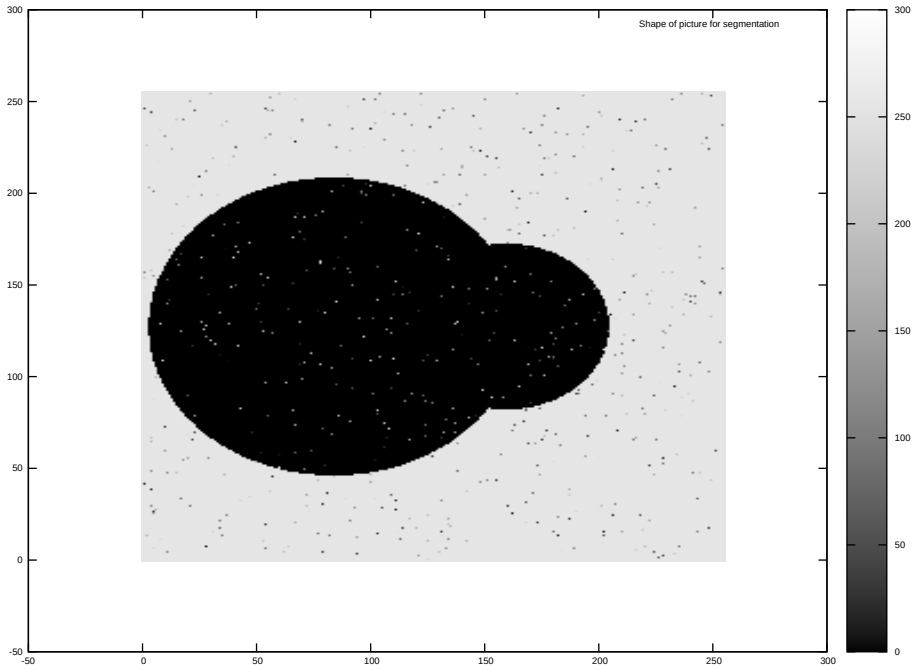

Figure 44: Image with 1% noise that we want to semgent out

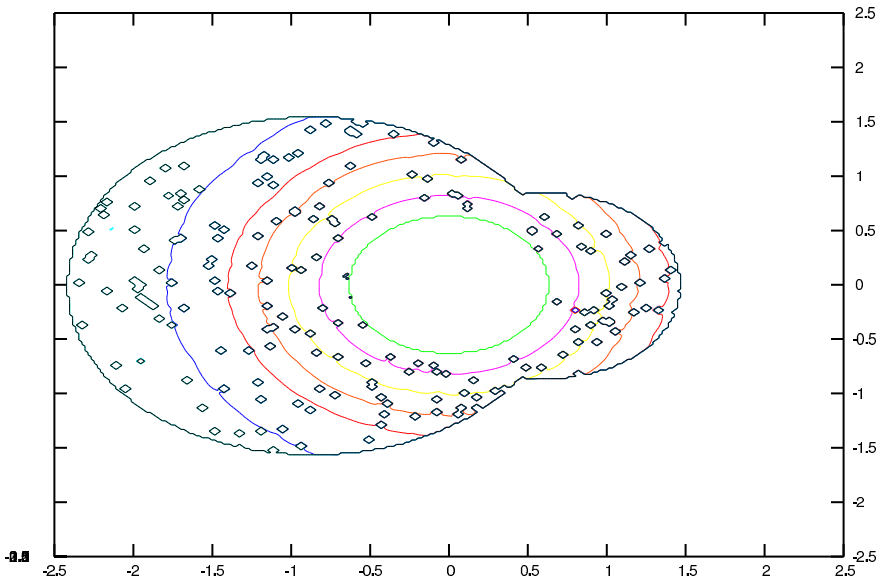

Figure 45: Segmentation using Level-Set method with 1% noise

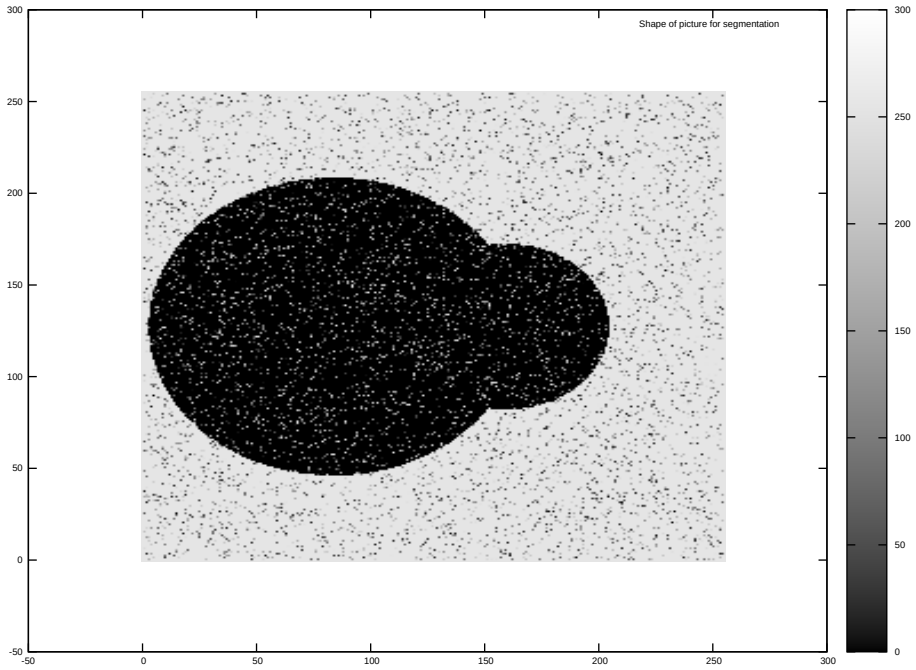

Figure 46: Image with 20% noise that we want to semgent out

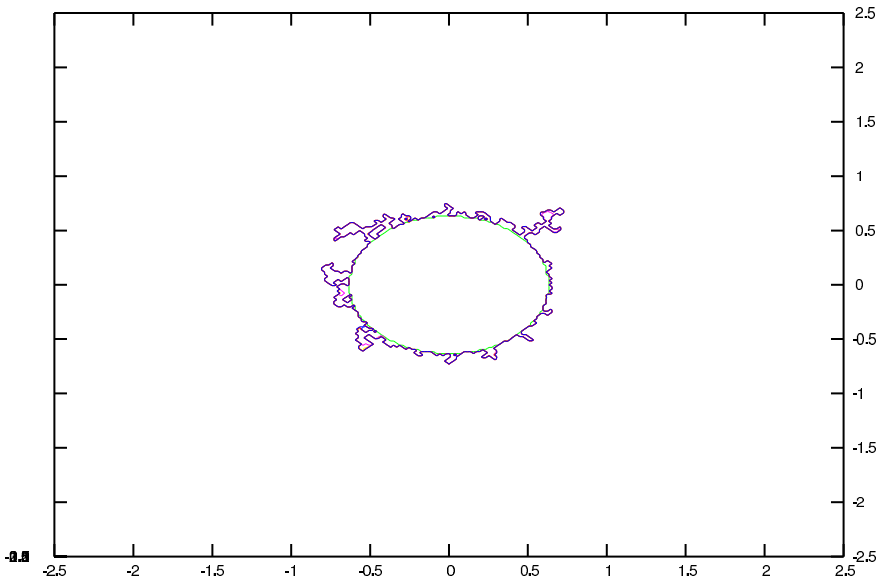

Figure 47: Segmentation using Level-Set method with 20% noise

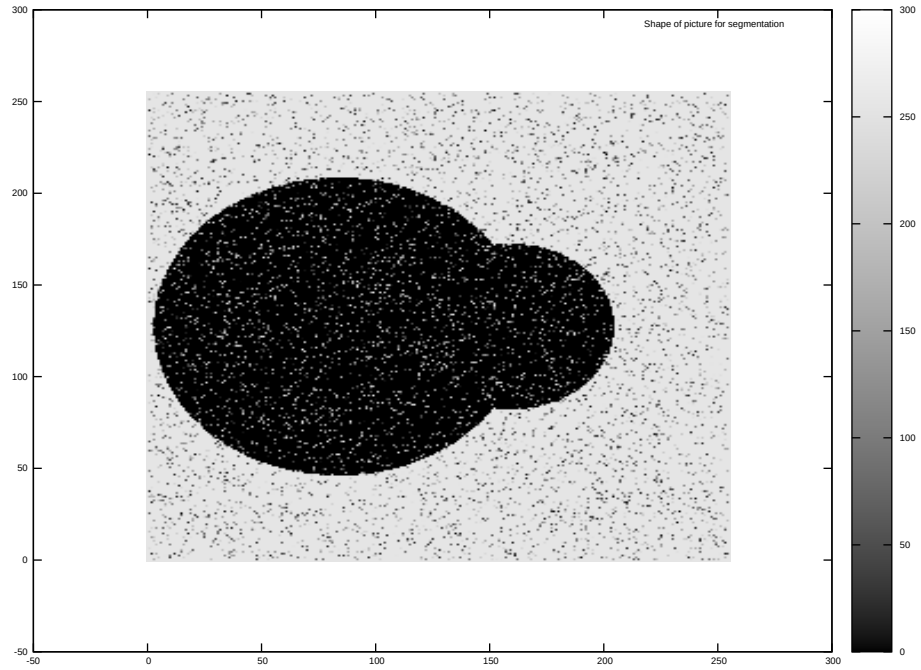

Figure 48: Image with 20% noise that we want to semgent out

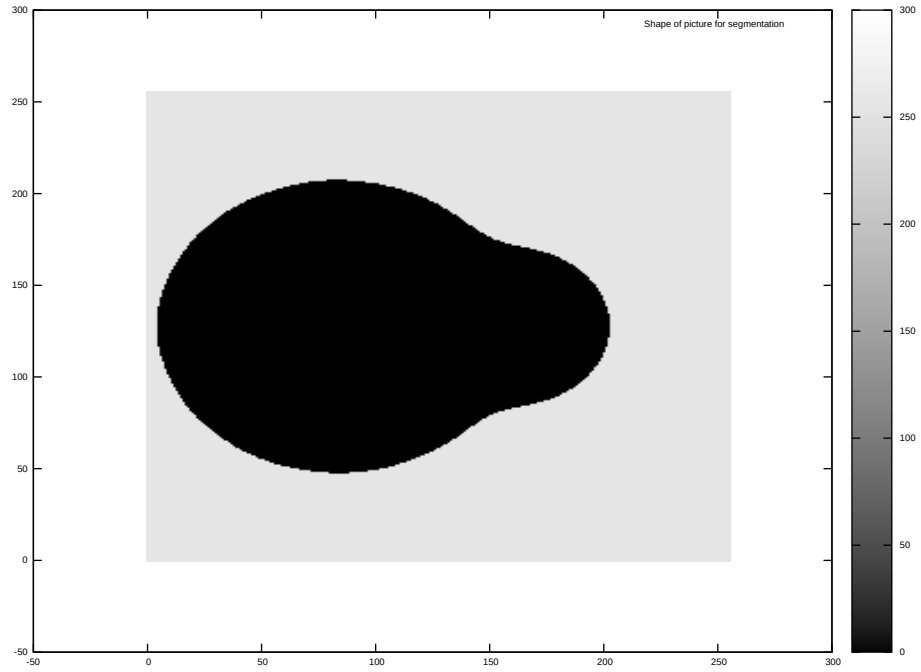

Figure 49: Noise cancelled Image with 20% noise, using Level-Set Methods

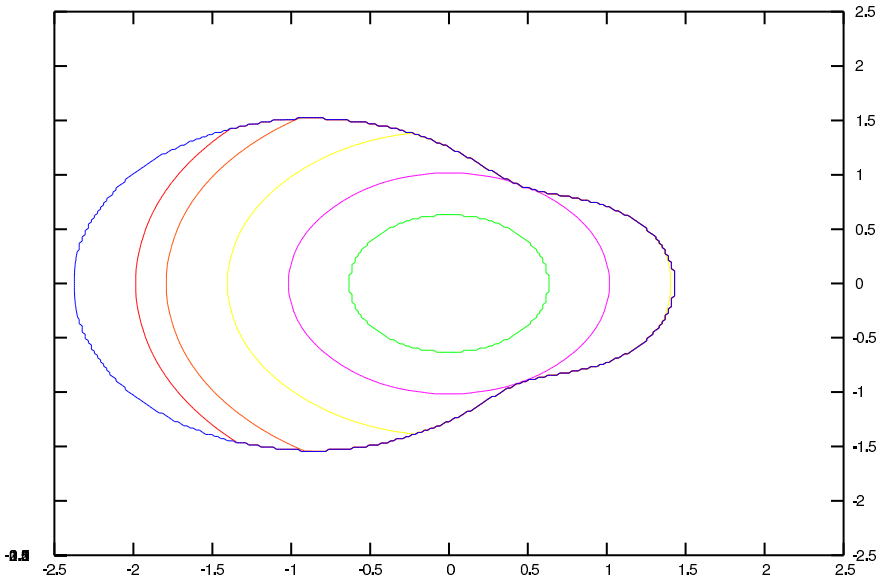

Figure 50: Image with 20% noise, Segmenting out after noise cancelation

Conclusion

In the problem set, we saw how to solve front propagation problems. Basically, we discussed boundary value problem, marker particle tracking and initial value problem. We saw that solving boudnary value problem requires that speed function to be either positive or negative. We found that in marker particle method the entropy condition is not satisfied and the method can not handle sharp corners that are formed in the simulation. Level-Set method has none of these problems and can be used to solve front propagation problems. We saw this method can be used to segment out part of an image or to cancel noise. Comparing to Gaussian methods for noise cancelling, Level-Set methods give better results, since this method shrinks the image rather than smoothing the intensity. We also found out how we can use intensity of the image as speed function of front propagation problem in order to segment out part of a picture.