Computer Vision
Programming Assignment 2

Results and Report

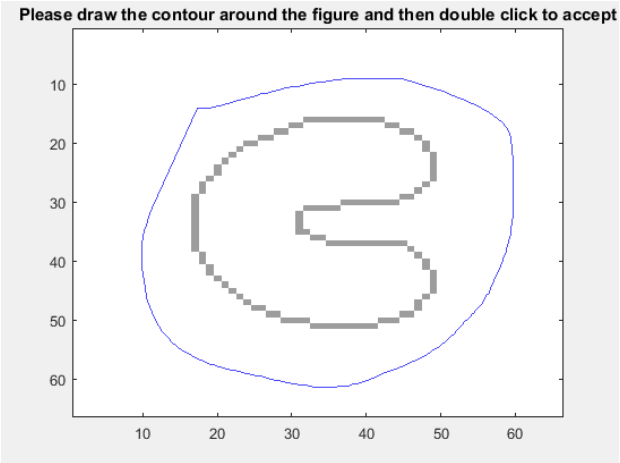
Question 2:

a) Code attached in gvf23.m file

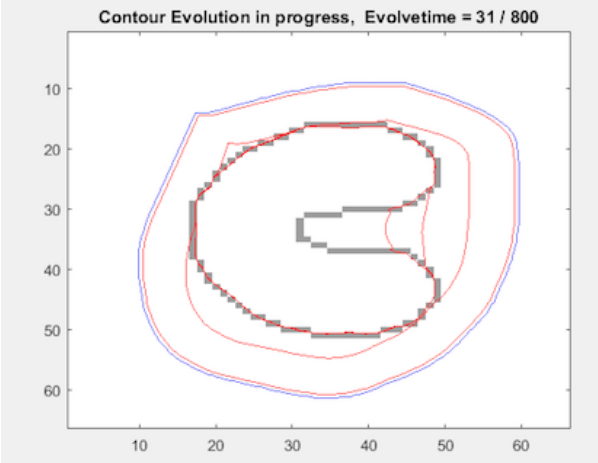
b) Snapshots:

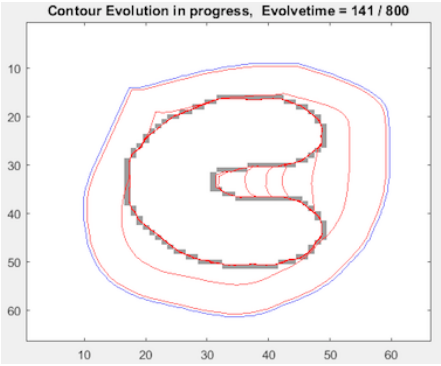
Figure 1: pacman.pgm

Initialization of contour



Intermediate Stages:





Final contour around the image:

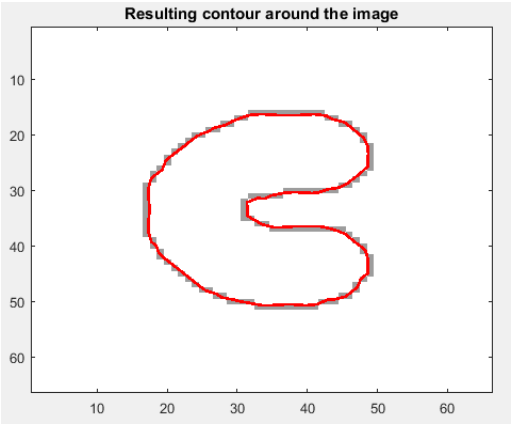
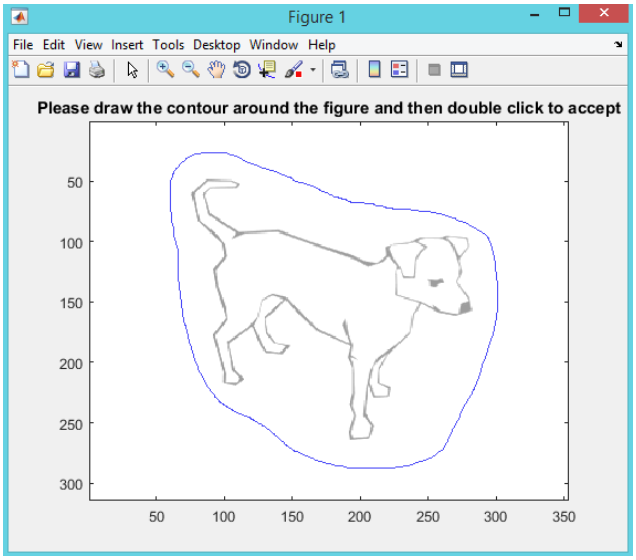
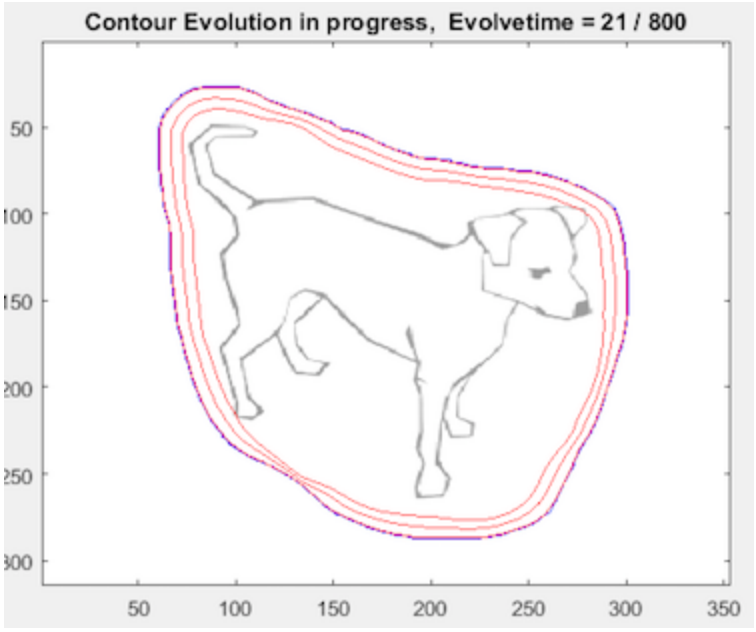


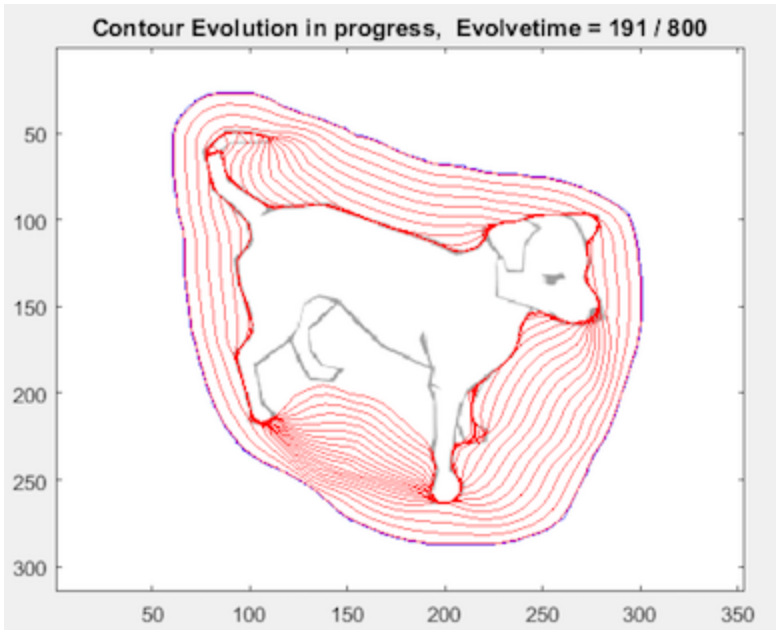
Figure 2: doggy.jpg

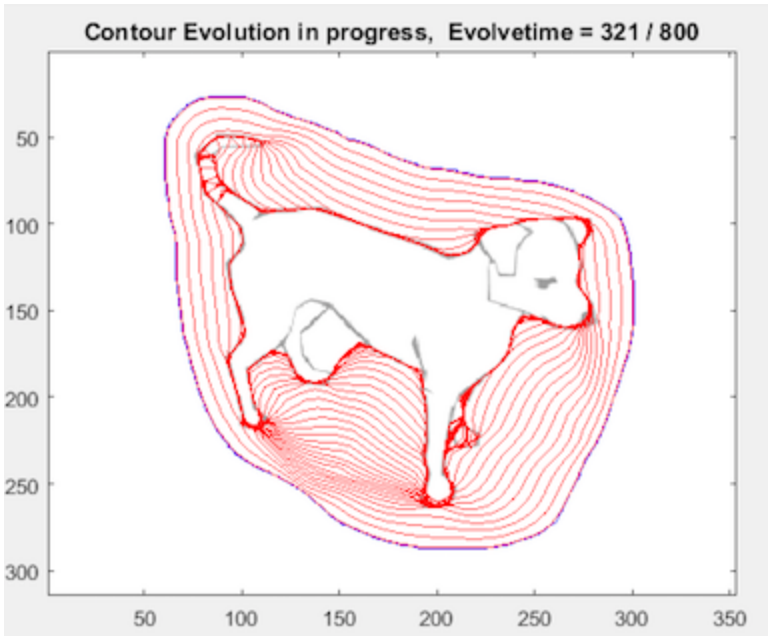
Initialization of contour

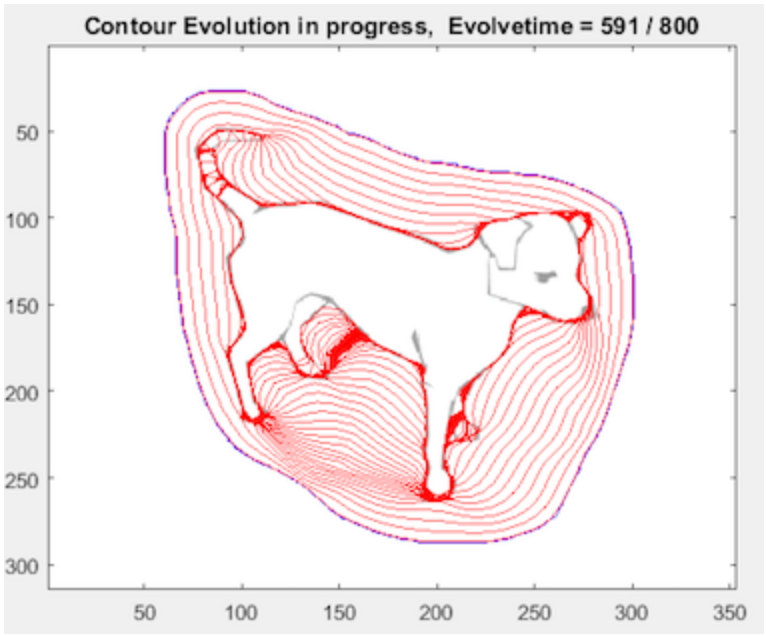


Intermediate Stages:

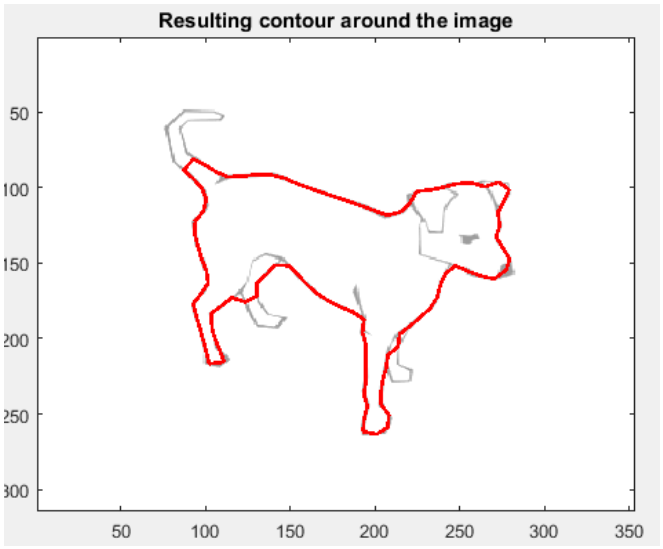








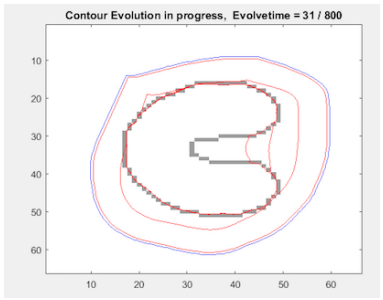
Final contour around the image:

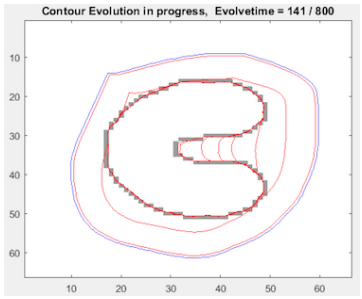


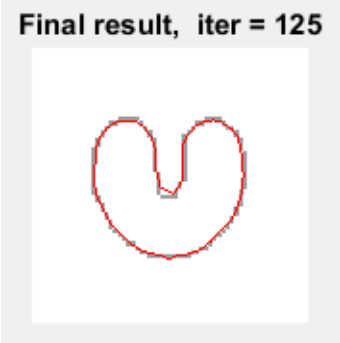
2. c) Comparison of original GVF code as downloaded vs my code:

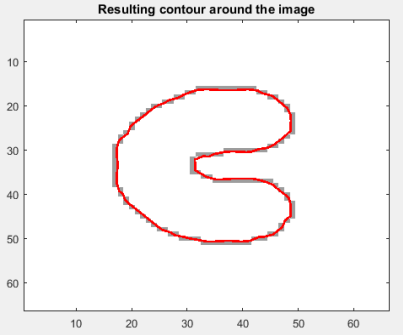
Downloaded code:

My code:









Report:

I have tried to implement the functionality of the GVF active contour as much as possible like described in the paper. The equations I used to form the Gradient Vector field matrices are equation 14a and 14b from the paper.

$$u_t(x,y,t) = \mu\nabla^2 u(x,y,t) - [u(x,y,t) - f_x(x,y)] \cdot [f_x(x,y)^2 + f_y(x,y)^2] \quad (14a)$$

$$v_t(x,y,t) = \mu\nabla^2 v(x,y,t) - [v(x,y,t) - f_y(x,y)] \cdot [f_x(x,y)^2 + f_y(x,y)^2]. \quad (14b)$$

Once we have the gradient vector field matrices it gives us how each point around the image would move towards the image. So once we have the contour from the user we multiply the current position of the points in the contour with the corresponding GVF matrix value at that point which gives us the location of the new point. Thus the contour evolves till it reaches the point where there is no change in the Gradient Vector field matrix.

I have used the function interp1 to resample the curve after each evolution. Also I used interp2 to find the corresponding value of the gvf with the points of the contour.

I noticed the following thing about the regularization parameter mu. With a low mu value the contour evolves faster but gets stuck at insignificant changes in the image and is not able to reach the actual edge of the intended figure. A high value of mu makes the contour ignores the insignificant changes in the image and converges to the intended edge but takes a lot more time as the contour evolution happens slowly. With the images I used there was no irregularities around the image hence a low mu value worked well so as to be well defined around the edges. I used a mu value of 0.1. I used a constant kappa value of 0.6 as curvature flow in my equation.