CSUMS Second Semester

Kerenne Paul
Department of Mathematics
Student – University of Massachusetts Dartmouth

Advisors: Saeja Kim and Sigal Gottlieb
Department of Mathematics
Professors – Unversity Massachusetts Dartmouth

05/12/11

Abstract

The following project focuses on the use of Radial Basis Functions (RBFs) in Edge Detection in both one-dimensional and two-dimensional images. We used a 2-D iterative RBF edge detection method. In order to have a better understanding of how the method works, we varied the point distributions and the shape parameter in the Matlab code provided by Professor Gottlieb. We also quantify the effects of the accuracy of the edge detection on 2-D images. In addition, we used radial basis functions of different nature such as the Inverse multi-quadric function and the Gaussian function. The result when running the program was an edge map of a specific one-dimensional or two-dimensional image that was incorporated in the code. The accuracy and the clarity of the edge map vary for each type of function in accordance to the epsilon value that best fits the respective RBF.

1 Introduction

During my second semester in CSUMS, professor Gottlieb introduced me to a project that dealt with Radial Basis Functions and their application in finding edge detection for both one-dimensional and two-dimensional images. She presented me along with my other group member a Matlab code which we would use for the project. Her code originally used the Multi-quadric RBF

$$\phi = \sqrt{(x - x_i)^2 + \epsilon_i^2}$$

Understanding and working with the project required us to familiarize ourselves a little more with the program matlab and also investigate on how some parts of the codes were formulated and how they work.

2 What exactly are Radial Basis Functions?/ Problem

Radial basis functions use the relationship of radial distances between selected points on a given function, called center-points, and a special matrix calculation in order to figure the missing lambda coefficients so that we can

approximate intermediary data points and develop a new estimated function which will fit the position and shape of the initial function pretty accurately.

In our project, we started by working with the multi-quadric function that was initially in the code. Our task was to change around the epsilon and see at what value we get a more accurate view of the edge map. We defined accurate in terms of which edge map produced less noise in the background and which resembled the original image the most. For the multi-quadric function, the value of epsilon that worked best was at 0.1

We later wanted to get a better view of the edge map so we decided to play around with the point distribution. We changed it from the original number of points, 125 to 160 but the result was not really pleasant. We later discovered that we had to change the number of points which was listed at the beginning of the code in order for us to see a full view of the images. Other factors such as the dimensions of the images that we were using in the codes came into play when getting a good view of the edge map.

```
N = 284; NR = 500; xR = linspace(-1,1,NR);

tmp = linspace(-10,10,N);

x = tmp';

P = double(imread('stars.jpg'));
%P = double(imread('test1.tiff'));
P=P(:,:,1);
P=P(1:N,1:N);
imagesc(P), colormap(gray(N))
```

The next step in our project was to investigate on how the other two Radial Basis Functions would work in our code. First we started with the inverse multi-quadric:

$$\phi = \frac{1}{\sqrt{(x - x_i)^2 + \epsilon_i^2}}$$

Then taking the derivative of that, we ran the code and it gave us an edge map that was pretty similar to one we got for the multi-quadric one. Just like

the for the multi-quadric, we had to play around with the shape parameter and we found that it worked best at epsilon= 0.69.

```
%%%%%%% this is the code for the inverse multi-quadric RBF
   M(ix,iy) = 1/sqrt( (x(ix)-x(iy))^2 + (eps(iy))^2);
  if M(ix,iy) == 0
    MD(ix,iy) = 0;
  else
  MD(ix,iy) = -(x(ix) - x(iy))/sqrt( ((x(ix)-x(iy))^2 + (eps(iy))^2)^3);
  end
```

We later applied the same method for the Gaussian function:

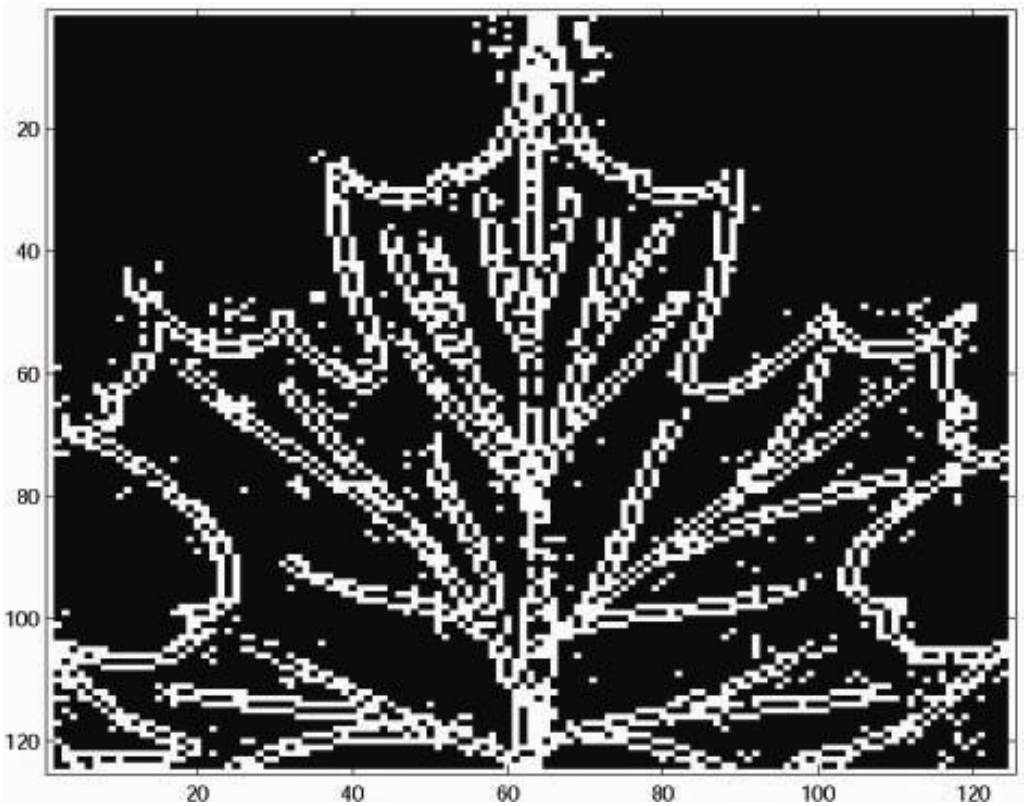
$$\phi = e^{(-\epsilon^2 * (x - x_o)^2)}$$

We incorporated it along with its derivative into the code and we found that the best epsilon value for it is at 3.

```
%%%%%%%% This is the code for Gaussian RBF
  M(ix,iy) = exp(-((eps(iy))^2)*((x(ix)-x(iy))^2));
  if M(ix,iy) == 0
     MD(ix,iy) = 0;
  else
  MD(ix,iy) = -2*((eps(iy))^2)*(x(ix)-x(iy))*exp(-((eps(iy))^2)*(x(ix)-x(iy
  end
```

These are the edge maps that we got for different images respectively for the multi-quadric function at epsilon= 0.1, the inverse multi-quadric at epsilon= 0.69 and the Gaussian function at epsilon =3.

Multi-Quadric at epsilon = 0.1



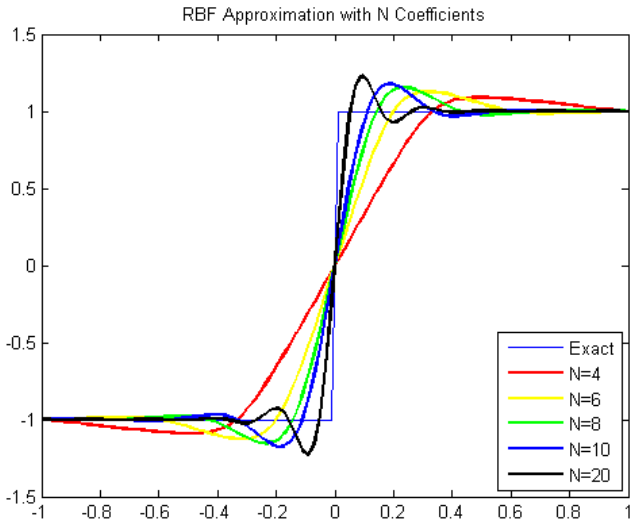
Inverse Multi-Quadric at epsilon = 0.69

Gaussian at epsilon = 3

While playing around with the code, we also ran it from the x-direction and the y-direction individually. The end results were pretty similar and it did not make much of a difference.

3 MQ RBF

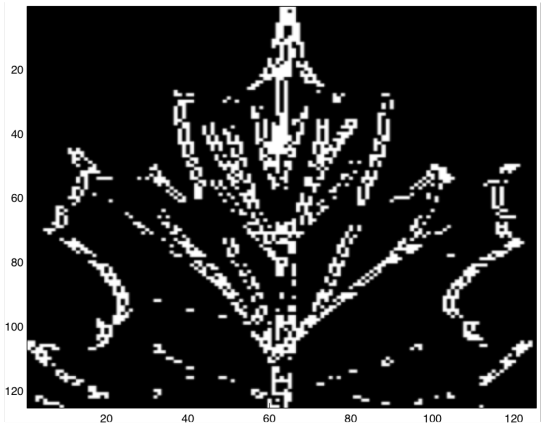
In approaching our project, we mainly focused on varying the code itself and analyzing the edge maps. We kind of neglected a major piece of the work. Our project uses a Multi-quadric RBF which plays the role of an approximator in the code. It uses a sumation of MQ RBF's with a certain value of epsilon evaluated anywhere from 1 to N coefficients. In one case we used this sumation to approximate a common step function beginning with 4 coefficients then increasing until we reach 20.



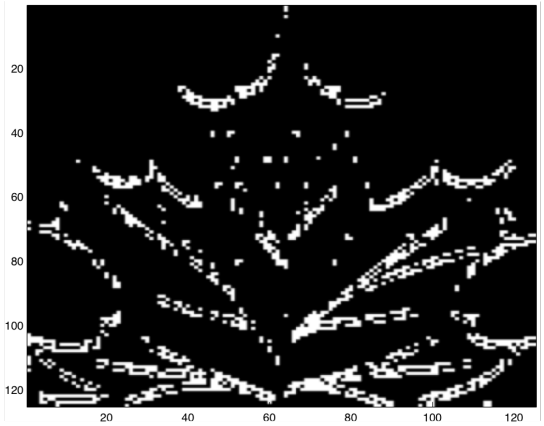
As we increase the number of coefficients, our approximation begins to look more and more similar to the actual function we are approximating. This is known as the Gibbs Phenomenon that I studied last semester in CSUMS. Each time this phenomenon occurs, our TwoD Example code draws out the edges for the specific image that is being mapped.

4 Techniques

Different techniques that we used in our project include changing the epsilon in order to figure out what value gives us a better edge map. We started by increasing the value of epsilon, that did not work out pretty well, so we reduced it. We eventually found a value that fit best. At a point we tried using a negative value for epsilon; that was before we realized that using a negative value is not really relevant, finding that epsilon is squared in the original RBF's. Other techniques that we used were changing the point distribution in order to get a better view of the edge-map and also running the code from different directions x and y individually.



Multi quadric x-direction



Multi quadric y-direction

Multi quadric Total (x-direction+y-direction)

5 Progress

My partners and I have worked towards accomplishing what had been asked of us at beginning of the semester. We have a way better understanding of how the code itself works. There is however a part of our project that is not quite clear to us. The involvement of matrix in the nature of RBFs and our project is still an area that we have to get more familiar with.We neglected to work on the part of our project that dealt with the Gibbs phenomenon, but we gained a better understanding of it last week after a meeting with professor Kim.

6 CSUMS

My second semester in CSUMS was great. I enjoyed working with my partners Chris and Tian. I gained a more understanding of how to use the program matlab and I also have mastered the LATEX program through CSUMS; all of which I will be able to use in the future for various tasks. This course has also enabled me to apply some of the skills I learned in my previous math classes. The in-class presentations that we had to do helped me better my public speaking skills and also the events that we had such as Sigma Xi and the Undergraduate Research Conference were enjoyable and again helped me better my public speaking skills. The roadblocks that Ive encountered, such as minor troubles with matlab and screwing up my code served as reinforcers in understanding my project.

7 References

1. Vincent Durante, Jae-Hun Jung. An iterative adaptive multi-quadric radial basis function method for the detection of local jump discontinuities. Appl. Numer. Math. 57 (2007) 213-229