Seth Balodi
ECE 4560
Digital Image Processing
Project 2 : Edge Detection
Spring 2017

ABSTRACT

Edge detection is a very interesting application of Digital Image Processing and allows for many cool things to happen in technology. Windows Hello can use face detection software which uses edge detection to separate objects in a view and snapchat has many filters that it uses that have some form of an edge detection software running in some capacity. In Project II, we were to choose two different methods of edge detection, either the Canny Edge Detector or the Multiscale LOG Detector. Based on our choice, we were to output certain intermediate steps as well as the final edge detection pictures that showed similar values to the built in MATLAB function “edge”, which can take parameters ‘zerocross’ (for Multiscale LOG Detection) or ‘canny’ (obviously for the Canny Detection). THe following report will outline my procedure and the results I achieved through my implemented algorithm.

APPROACH

I decided to go with the Multiscale LOG Detector as I was more familiar with gaussian and laplacian filters and so less research was needed to implement the algorithm. This technique is also known as the Marr-Hildreth Edge Detector which uses a threshold that measures intensity changes at different scales. A sudden change in intensity will cause a peak or a trough to occur. This particular edge detector requires a differential operator as well as the ability to be tuned to act at any scale. The filter takes a gaussian and a laplacian filter and convolves them. After this, it is a simple matter of finding the zero crossings of the image produced from the convolution of the gaussian and laplacian image. The zero crossing represent where the image goes from one intensity to another, or from a grayscale image, from black to white in this case. Using various levels of sigma through a gaussian filter, different results are achieved. It is important to threshold the maximum value so that a proper result can be determined, as without it, not enough, or too many, zeros could be identified and mess up the edge detection. Fine tuning this is key, but in the class lecture notes, a 4% threshold was shown.

I began by reading in the grayscale file and converting it to double for extra precision. I then set up a for loop that would try all of my results with different values of sigma and then subplot them for debugging and correctness, and also for the final submission. I used the algorithm for the gaussian filter using the gaussian filter equation. I experimented initially with using the ‘fspecial’ and ‘imfilter’ functions but the results of the values of their matrixes were very close and therefore it did not matter which one I used. I ended up using my own algorithm, which I then normalized to ensure that the filter would produce an overall sum of 1.

I took this blurred image and convolve it with the original image, making sure that the edges of the matrix did not convolve externally with the zeroes from the gaussian filter. I then plotting this intermediate calling it “Blurred Picture”. This result would then need to be filtered with a laplacian filter with the matrix ‘ H = [-1 1; 1 -1]’. Taking the image and making sure the positive values are displayed with the proper threshold and also in grayscale was next, followed

by running down the other parts of the specification. The thresholded zero values are shown displayed in all gray (128), the positive and negative values are shown in white and black respectively (255 and 0) and finally all the zero crossings are shown. The experimental results of my algorithm are shown in the section below.

.

EXPERIMENTAL RESULTS

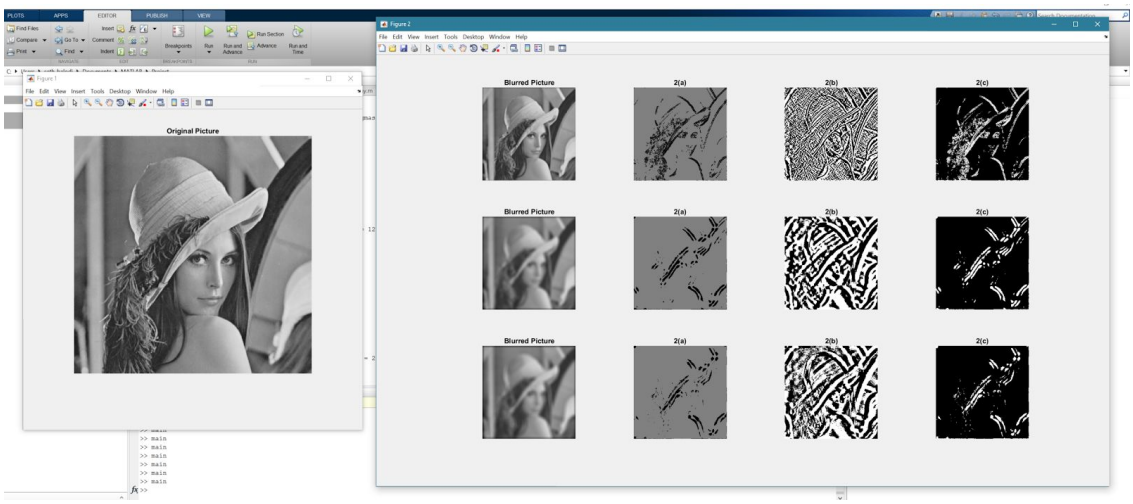

Figure 1: Overall Results

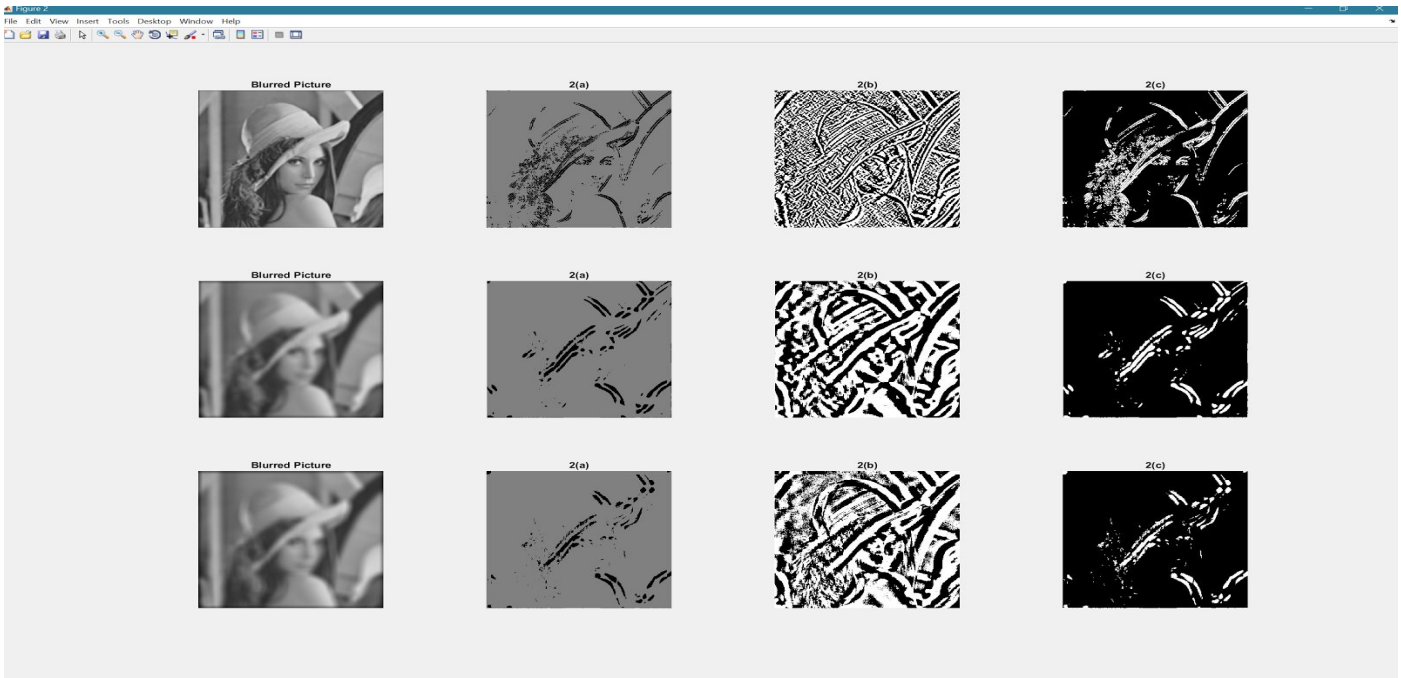

Figure 2: Close up final results

Seth Balodi
ECE 4560
Digital Image Processing
Project 2 : Edge Detection
Spring 2017

Editor - C:\Users\seth_balodi\Documents\MATLAB\Project\main.m

main.m

```
clear;

lena = imread('lena.png');
lenaDouble = double(lena);

figure
imshow(lena)
title('Original Picture')
sigmas = [2 8 16];
figure;

for i=1:length(sigmas)

    % Step 1
    % Filter the image by Gaussian lowpass filter
    N = 25;
    [X, Y] = meshgrid(-N/2:N/2-1, -N/2:N/2-1);
    G = 1/(2*pi*sigmas(i)^2)*exp(-(X.^2 + Y.^2)/(2*sigmas(i)^2));
    G = G/sum(G(:));

    bluredImage = (conv2(lenaDouble, G, 'same'));
    subplot(3,4,4*(i-1)+1);
    imshow(uint8(bluredImage))
    title('Blurred Picture')

    % Filter image with Laplacian filter
    H = [-1 1; 1 -1];
    laplacian = conv2(bluredImage, H, 'same');
    logImage = laplacian;
    logImage(abs(laplacian) < .05*max(laplacian(:))) = 128;

    subplot(3,4,4*(i-1)+2);
    imshow(uint8(logImage))
    title('2(a)')

    edgeImage = zeros(size(bluredImage));
    edgeImage(laplacian > 0) = 255;
    subplot(3,4,4*(i-1)+3);
    imshow(uint8(edgeImage))
    title('2(b)')

    zeroImage = zeros(size(bluredImage));
    zeroImage(abs(laplacian) > .05*max(laplacian(:))) = 255;
    subplot(3,4,4*(i-1)+4);
    imshow(uint8(zeroImage));
    title('2(c)')
end
```

Figure 3: Code implementation