Alejandro Makarem CS455
Assignment 2

Problem Statement (part 1)

- Description: Unsharp masking, Sobel operator, Laplacian of Gaussian operator, and Scale Space filter can be used for edge detection and enhancement. Apply these operators to a gray scale image to generate the edge image. In this assignment, you will implement these algorithms, and get various details of edge information in a given gray scale image.

- Implementation: Apply the Unsharp Masking method to enhance the edges of images(f1, f2) and obtain the enhanced image E1 and E2;

Algorithm design

I started with this project by creating the Unsharp Mask function. This function created a 2 dimensional array that will be used as a mask. This mask will later be the basis of the algorithm by performing a convolution filter against the image. This is done by grabbing every pixel in the image and multiplying each neighbor pixel by its corresponding pixel inside the filter. So I created a function called Unsharpmask(). This function contained the main filter:

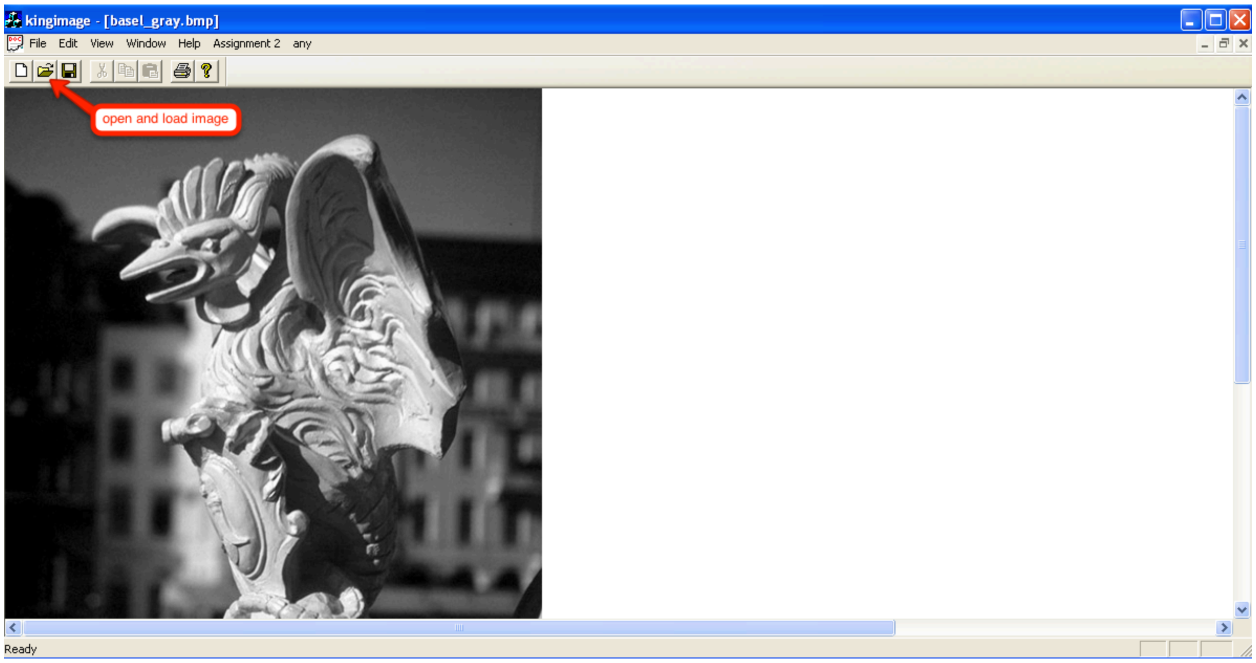
```
double filter[3][3] =
                {
                        -1, -1, -1,
                        -1, 9, -1,
                        -1, -1, -1
                };
```

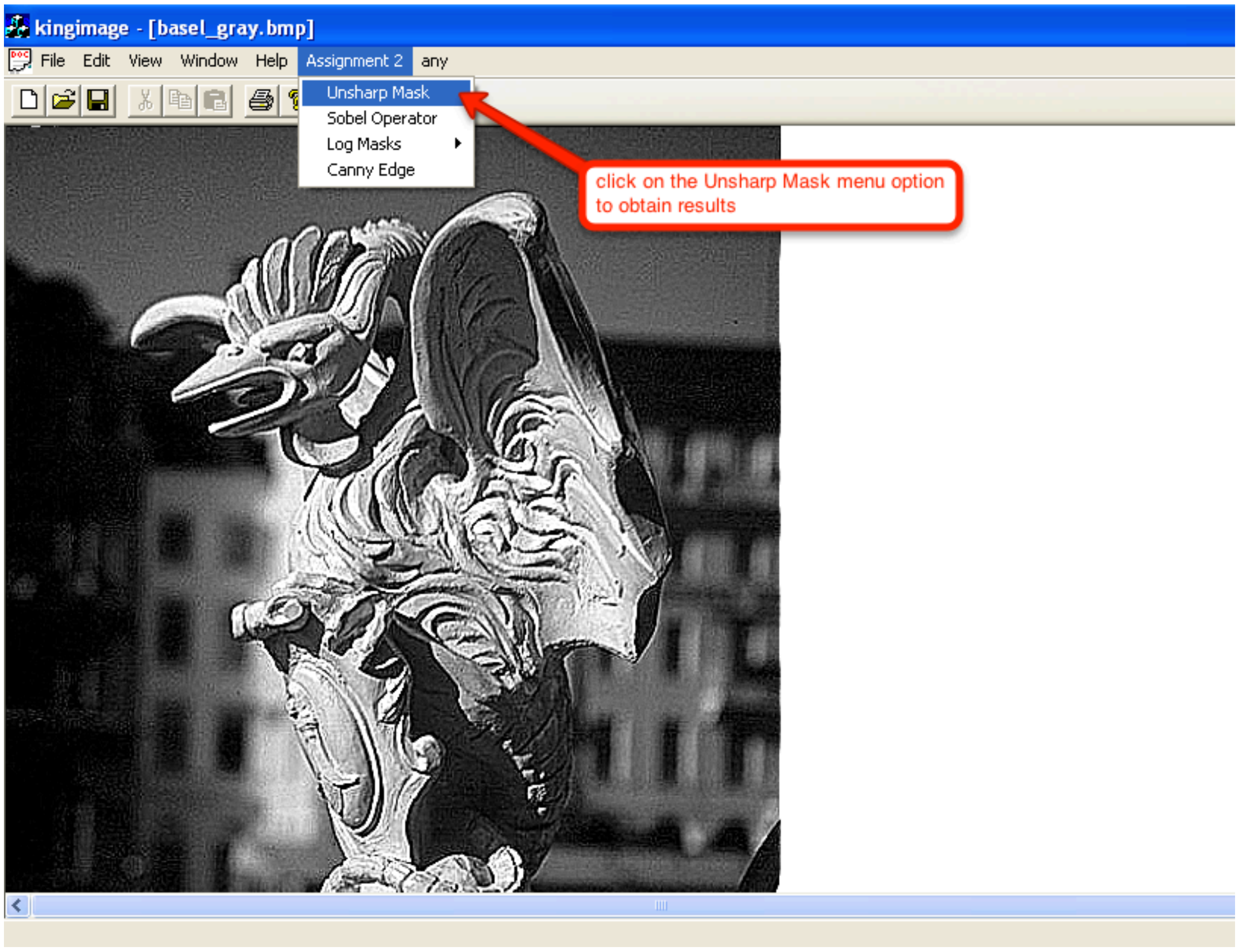
This filter acts as a shortcut of the formula: fsharp(x,y) = f(x,y) + k * g(x,y) where k is a constant. Basically it represents adding a blurring of the image and then subtracting the original that way the edges will be more sharp.
After applying this matrix to every pixel in the image, I assigned the result to a temporary matrix that will only serve to hold the results. This is due to the fact that if it affects the original image then all the following results would have been affected by the previous one, which leads to an incorrect interpretation of the algorithm and false results.

Manual

Load each image into the interface by opening the file. Click on the menu “Assignment 2” on the menu bar, which expands, and click on the “Unsharp Mask” menu. This will apply the algorithm to the grayscaled image.

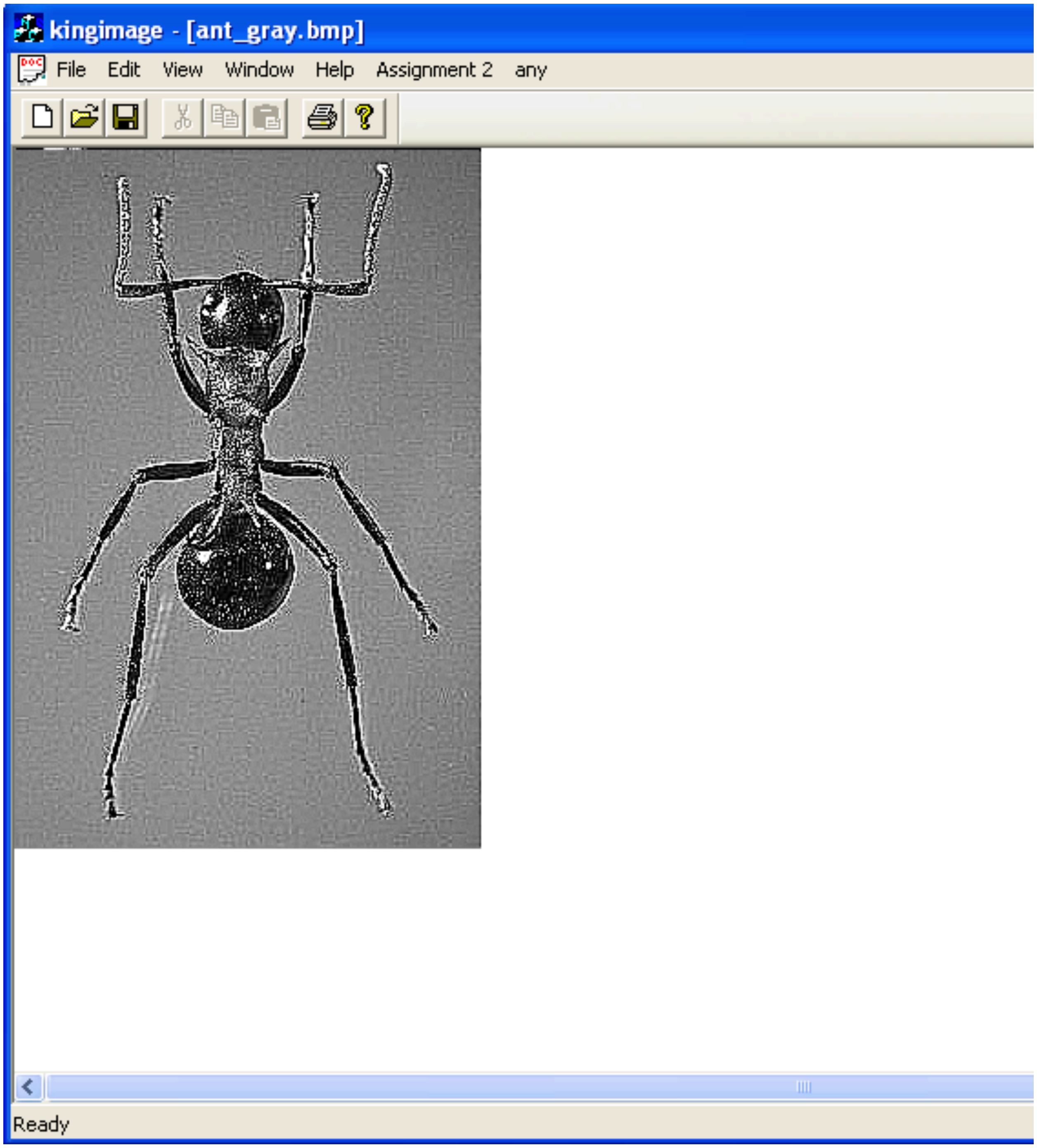
Alejandro Makarem CS455
Assignment 2


kingimage - [basel_gray.bmp]
File Edit View Window Help Assignment 2 any
open and load image
Ready


kingimage - [basel_gray.bmp]
File Edit View Window Help Assignment 2 any
Unsharp Mask
Sobel Operator
Log Masks
Canny Edge
click on the Unsharp Mask menu option to obtain results

Alejandro Makarem
Assignment 2

CS455



Alejandro Makarem CS455
Assignment 2

Problem Statement (part 2)

- Description: Unsharp masking, Sobel operator, Laplacian of Gaussian operator, and Scale Space filter can be used for edge detection and enhancement. Apply these operators to a gray scale image to generate the edge image. In this assignment, you will implement these algorithms, and get various details of edge information in a given gray scale image.

- Implementation: Apply the Sobel operator to the two images (f1, f2) and generate the edge image Es1 and Es2.

Algorithm Design

Very similar to the first part of the assignment, this part also included creating and implementing a convolution filter that would be applied to the image. I created the function Sobel() where included all the computations. There were actually 2 filters that had to be applied, one for the X values and another for the Y values.

```
double filter[3][3] =
{
        -1, 0, 1,
        -2, 0, 2,
        -1, 0, 1
};
double filtery[3][3] =
{
        1, 2, 1,
        0, 0, 0,
        -1, -2, -1
};
```

After these two filters would be applied to every pixel (so 2 filters per pixel), then the results had to be added together for both filter results. Then you applied the formula: |G| = sqrt(Gx^2 + Gy^2)
so basically my algorithm formula looks like:

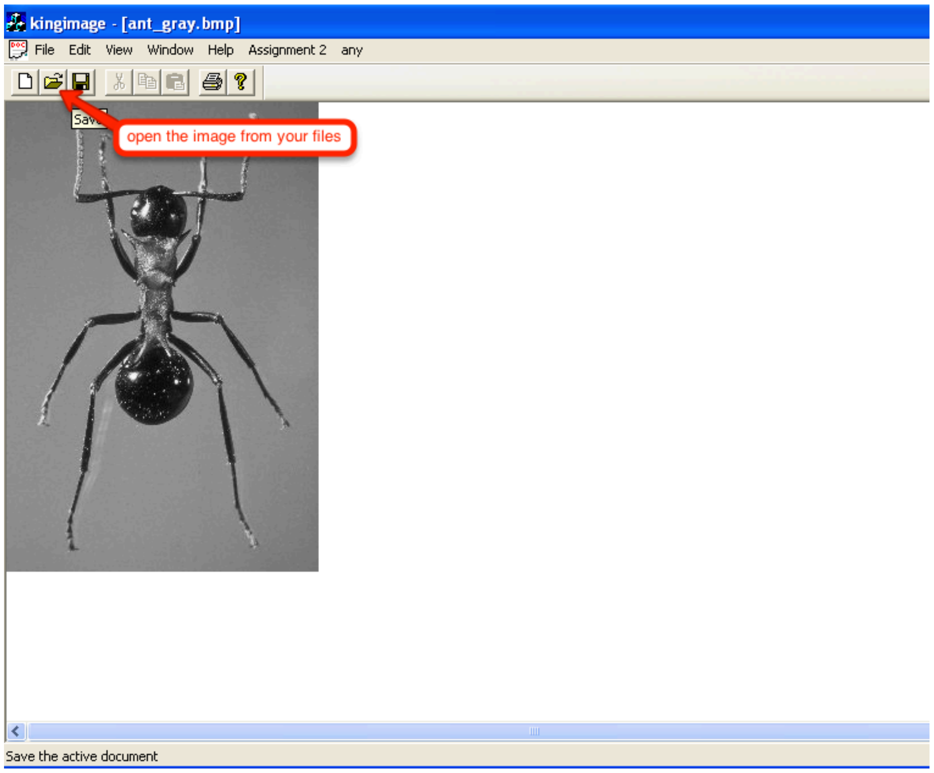
```
calc = sqrt((calcx*calcx) + (calcy*calcy));
```

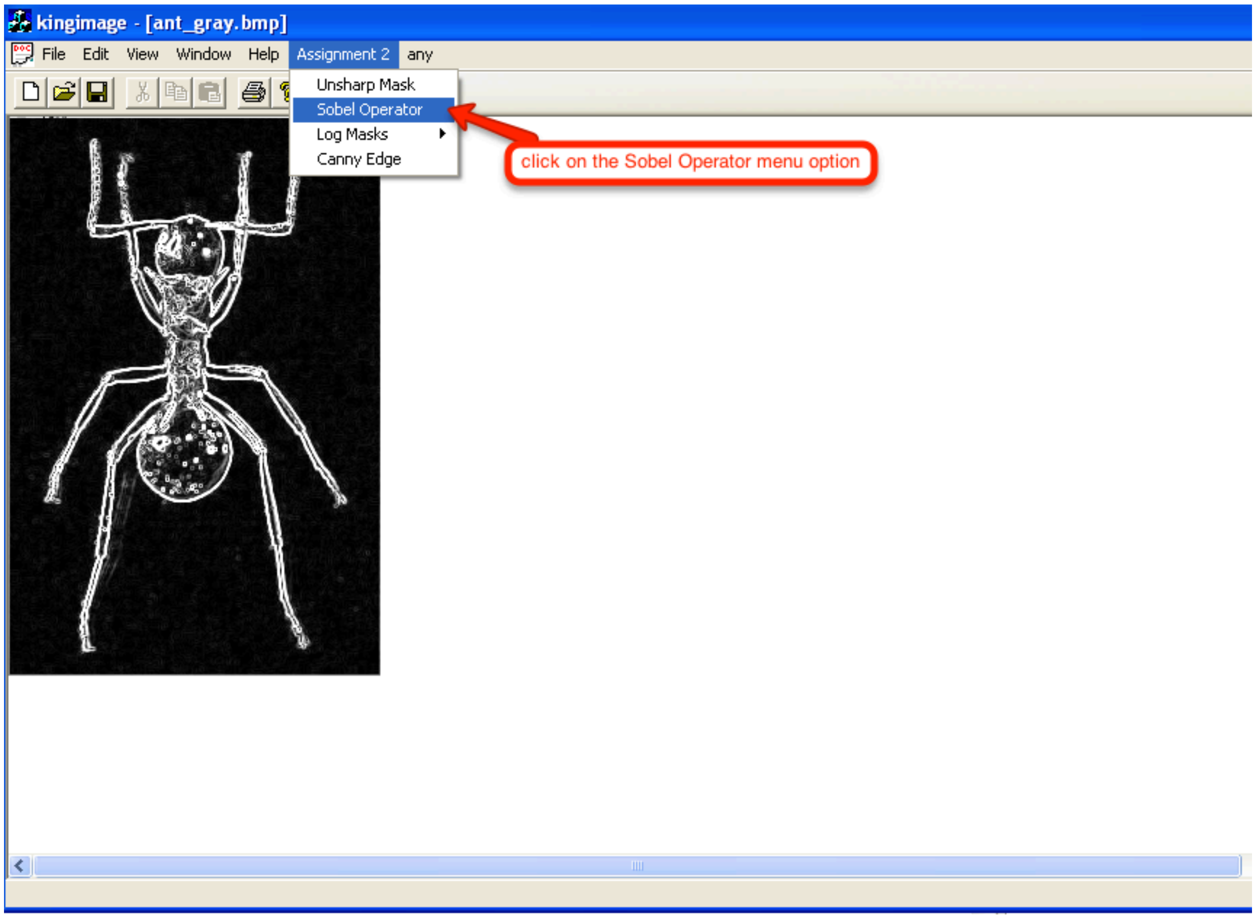
Then I finished off the function by saving the results into a temporary array that would not influence the results of other computations and then copied those into the original image once the computation was done. I must admit that my code is very repetitive and I could have created functions to make it more readable and cleaner, but for the sake of simplicity I needed to see every single computation being made which resulted in a large function.

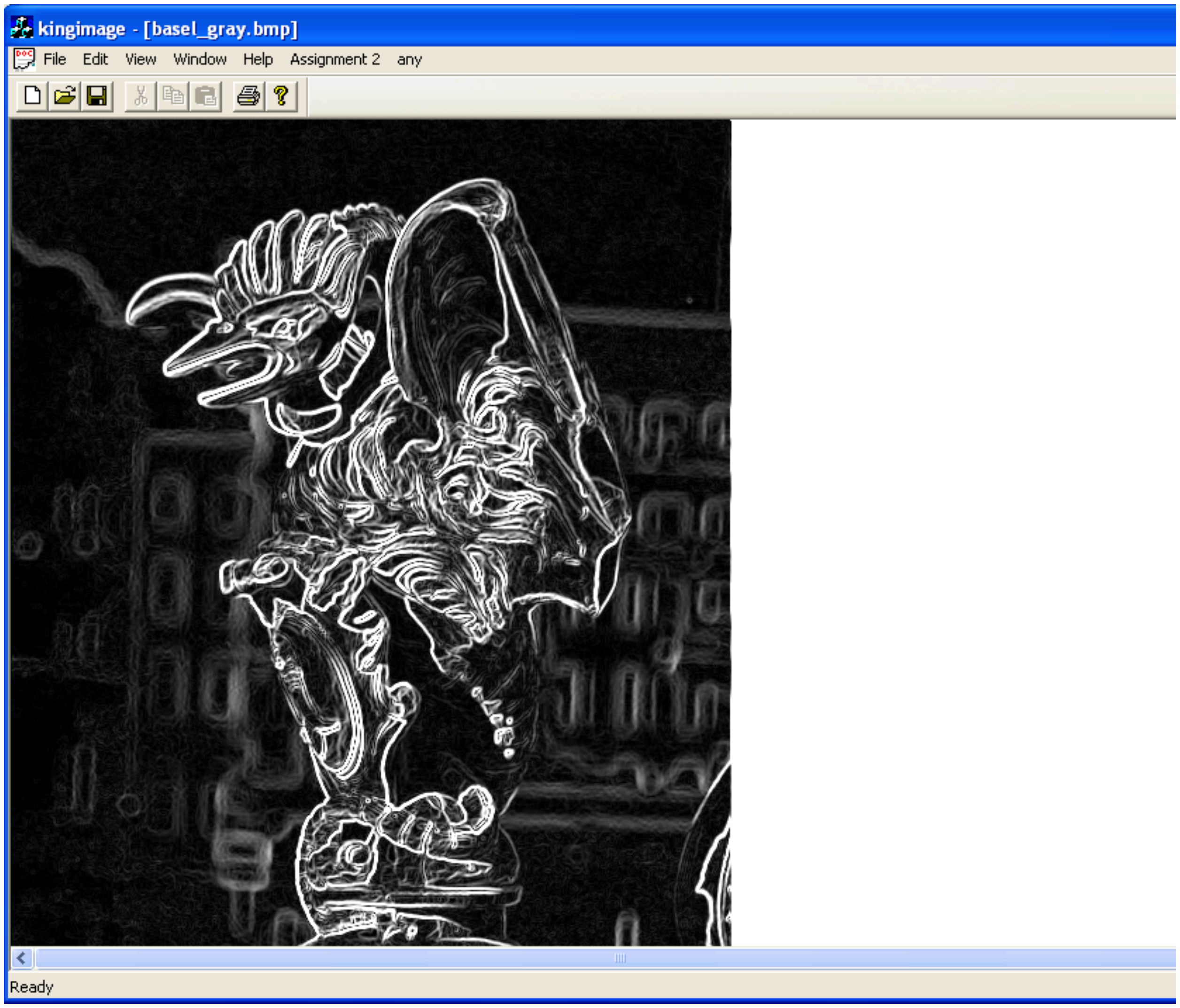
Manual

Same as part 1. Open each of the files by accessing your computer, and then click on the menu “Assignment 2” and the submenu item “Sobel Operator” which will apply the algorithm to the currently opened image.

Alejandro Makarem CS455
Assignment 2


kingimage - [ant_gray.bmp]
File Edit View Window Help Assignment 2 any
Save
open the image from your files
Save the active document


kingimage - [ant_gray.bmp]
File Edit View Window Help Assignment 2 any
Unsharp Mask
Sobel Operator
Log Masks
Canny Edge
click on the Sobel Operator menu option


kingimage - [basel_gray.bmp]
File Edit View Window Help Assignment 2 any
Ready

Problem Statement (part 3 and 4)

- Description: Unsharp masking, Sobel operator, Laplacian of Gaussian operator, and Scale Space filter can be used for edge detection and enhancement. Apply these operators to a gray scale image to generate the edge image. In this assignment, you will implement these algorithms, and get various details of edge information in a given gray scale image.

- Implementation: Generate LOG Mask1 7*7 (sigma = 1.4) and Mask2 11*11 (sigma=5.0).Apply Mask1 and Mask2 to intensity images (f1, f2), obtain two sets of edge images (E1_1, E1_2) for f1, and (E2_1, E2_2) for f2.

Algorithm

I started by searching for the values to fill the filter with. I needed to find a filter that would satisfy a size of 7*7 and a sigma of 1.4. This first part was not too bad. I created a 7*7 2 dimensional matrix that would hold these values and then I applied the formula to fill the results. This formula consisted in the following:

```
one = -(1/(3.14*sigma*sigma*sigma*sigma));
two = 1-((((di-k2)*(di-k2))+((dj-k2)*(dj-k2)))/(2*sigma*sigma));
three = exp(-(((di-k2)*(di-k2))+((dj-k2)*(dj-k2)))/(2*sigma*sigma));
coeff[di][dj] = one*two*three*NORM;
```

where:
k = 7
k2 = k/2
sigma = 1.4
di-k2 = -3, -2, -1, 0, 1, 2, 3
dj-k2 = -3, -2, -2, 0, 1, 2, 3
NORM = 25 (a normalizer for the results can be changed easily)

This formula filled the filter as needed to the apply the convolution on every pixel, this worked perfectly because all of the elements inside the matrix could add up to 0. Similar to problem 1 and 2, I proceeded to do the computations but this time inside a loop (more convenient for larger matrices). For every calculation made, I saved the results into a temporary matrix so the results would not affect each other. Then I copied the results to the resulting image.

For the second mask of 11*11 it was more complicated because the sigma was very high. In order for me to normalize the results this time, I had to manipulate the signs of the resulting matrix in order for it provide the correct results. The matrix was initially just shooting out negative numbers, making the whole image black. But I changed the initial row and column plus the center 5*5 cells into positive numbers so they counteract each other. There was a very small margin of error of about 1-2% difference between the positives and the negatives, but it does not affect the resulting image.

The main issue I had throughout all of the algorithms was a common one. I could not manage to change the border pixels of the images because the filter size would not allow me to. So for example, a filter with 3*3 dimensions would not allow me to change 3 of the border pixels all around the image.

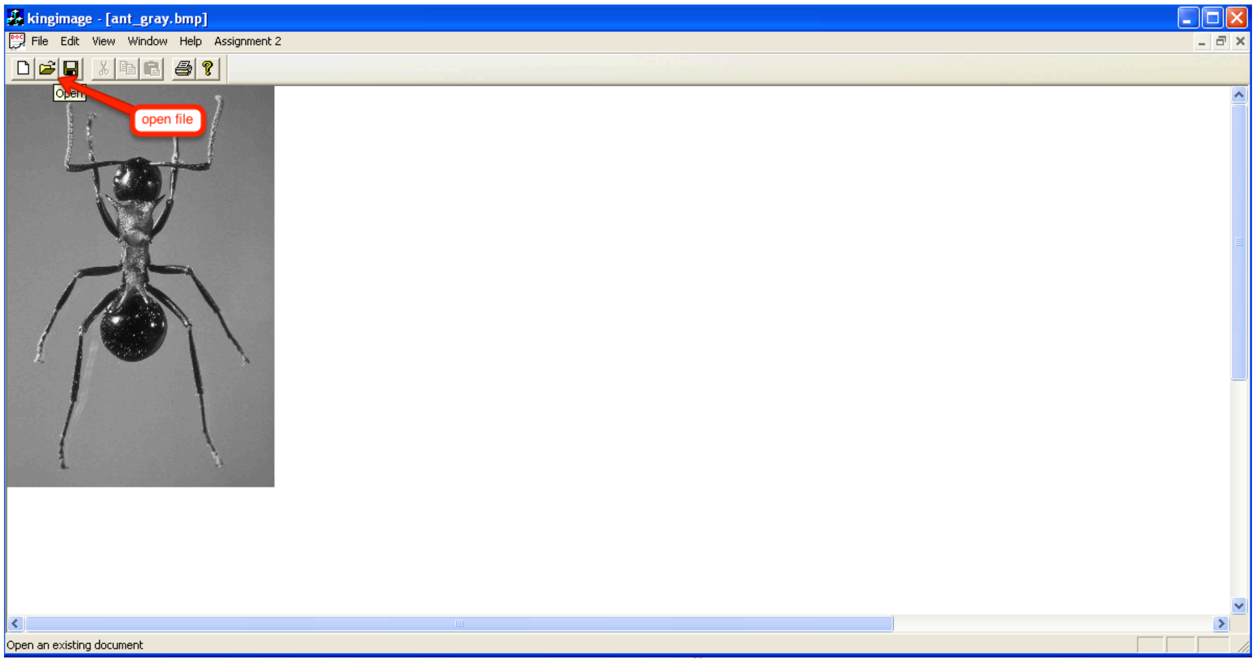
- Explain which images have more detailed edge information, and explain why.

I definitely think that the ant image had more detailed edge information, because it is consistent, high and bold contour changes and a small margin of error. There is a problem with the 'torso' of the ant because its color is very similar to that of the background and the average of the neighboring pixels are too similar to its color. Also with mask 11*11 on the picture there is more accurate creation of the filter which leads to better display of the photo. But much better than the basel picture. The examples below don't show a very accurate depiction of how they really show.

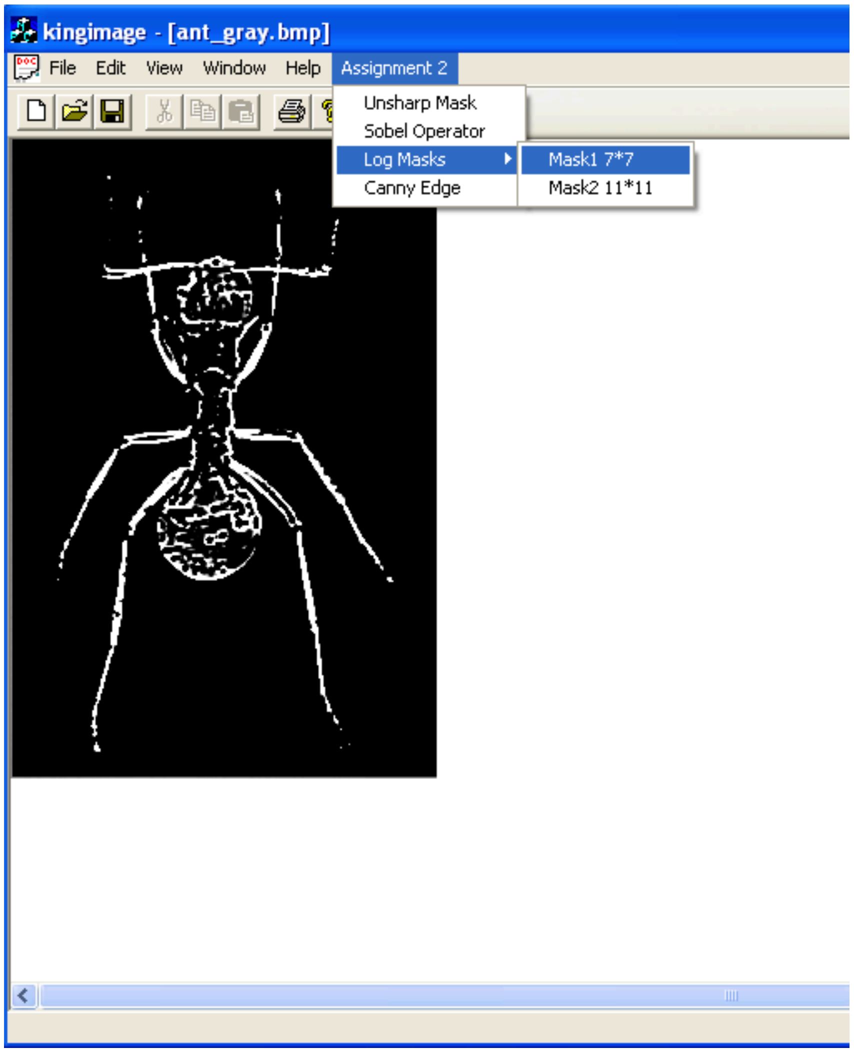
Alejandro Makarem CS455
Assignment 2

Manual

Same as other parts of the program. Click on the menu “Assignment 2” then on the submenu “Log Masks” which expands to 2 options. You have to reopen the image every time you want to apply a new filter.

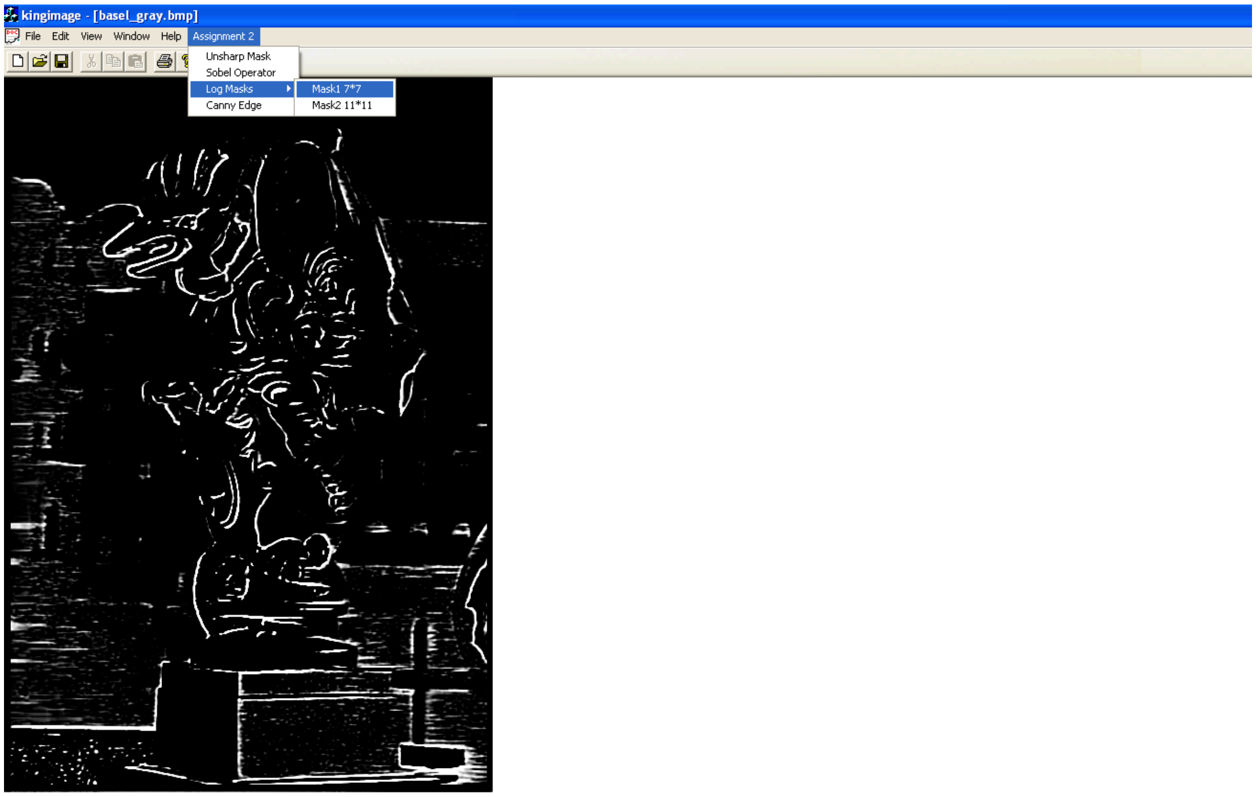

kingimage - [ant_gray.bmp]
File Edit View Window Help Assignment 2
Open
open file
Open an existing document

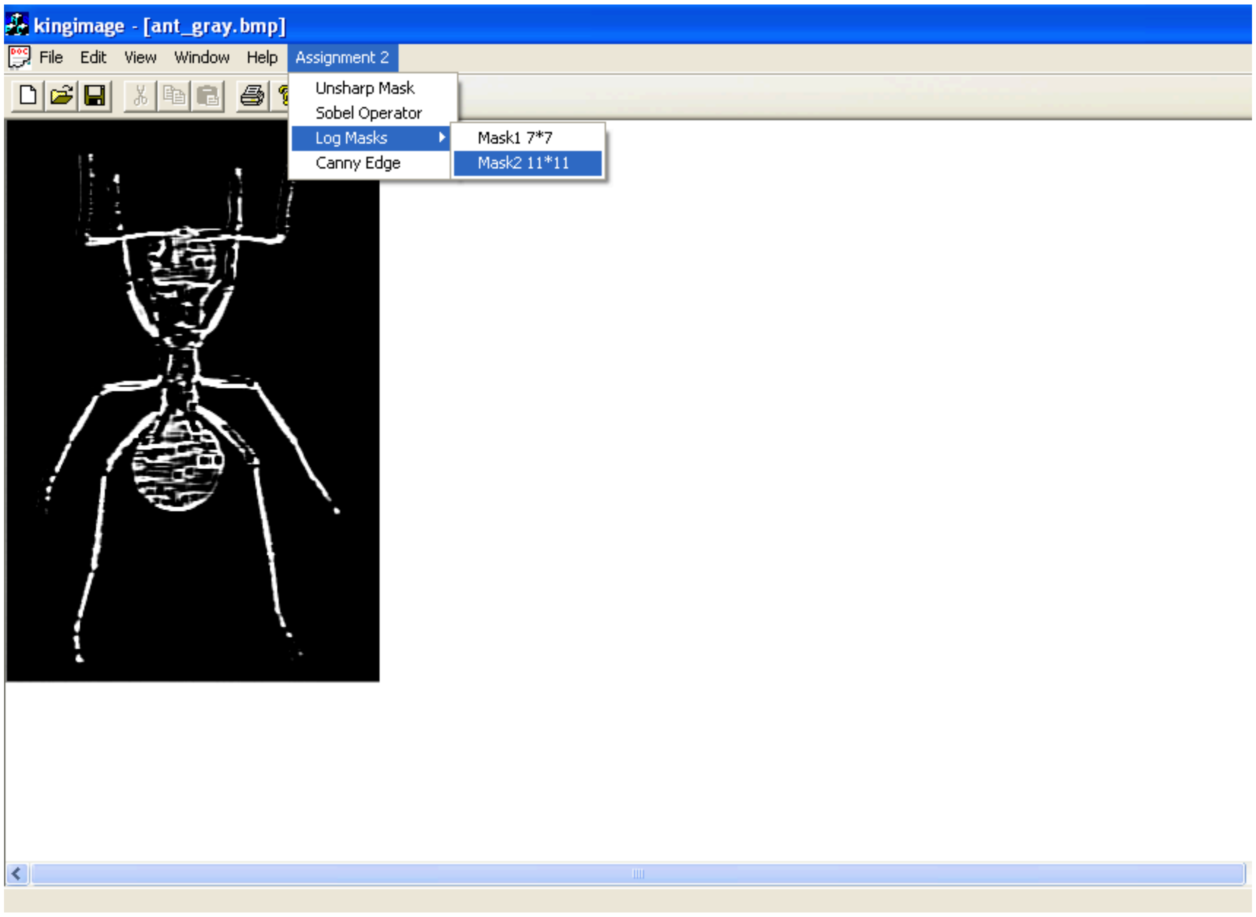
Alejandro Makarem
Assignment 2



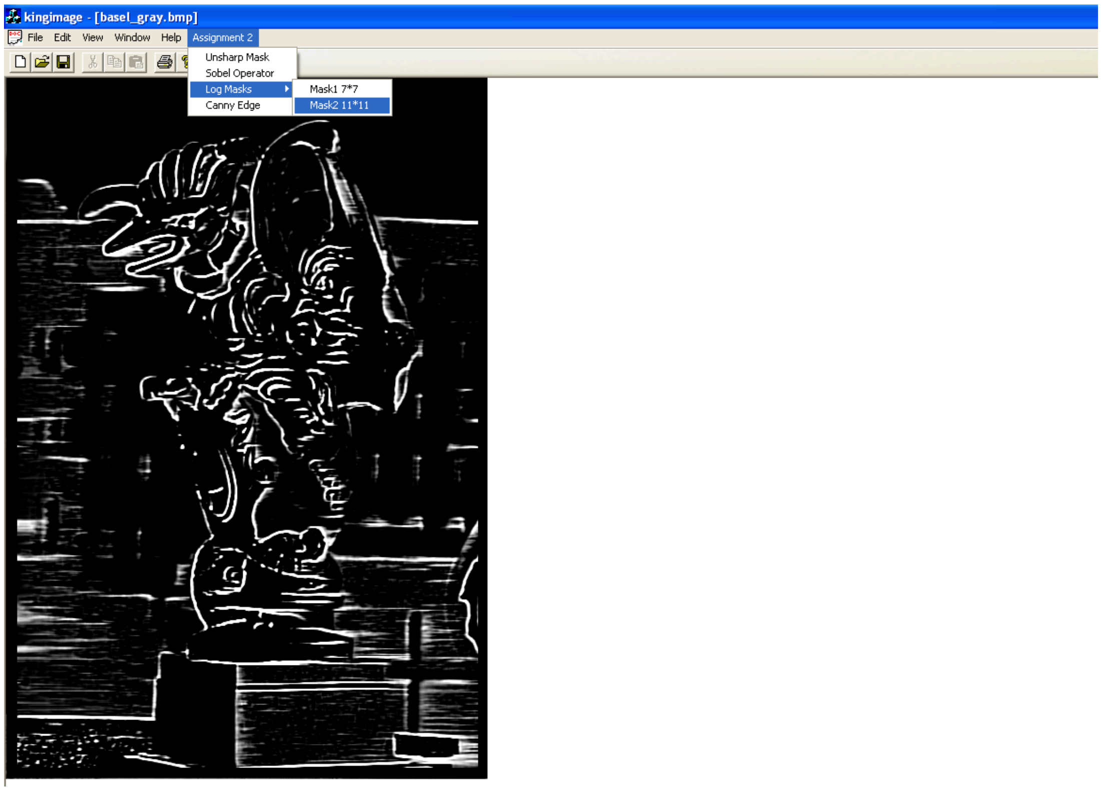
Alejandro Makarem
Assignment 2

CS455


kingimage - [basel_gray.bmp]
File Edit View Window Help Assignment 2
Unsharp Mask
Sobel Operator
Log Masks
Canny Edge
Mask1 7*7
Mask2 11*11


kingimage - [ant_gray.bmp]
File Edit View Window Help Assignment 2
Unsharp Mask
Sobel Operator
Log Masks
Canny Edge
Mask1 7*7
Mask2 11*11

Alejandro Makarem CS455
Assignment 2


kingimage - [basel_gray.bmp]
File Edit View Window Help Assignment 2
Unsharp Mask
Sobel Operator
Log Masks
Canny Edge
Mask1 7*7
Mask2 11*11

BONUS:

The zero crossing points. Works in conjunction with the LoG filter. It is supposed to detect when the graph of the image passes 0 and when it's above a certain threshold. I did this by making sure that every time the image has passed on 0 a white pixel is created on that spot, by creating a temporary variable that compares every new result with the previous result, and if the signs are different then it means it passed through 0. And for the threshold, if you increment the NORM constant enough, logically there will be an automatic threshold set for your image.