EE 417 Computer Vision

Berkant Deniz Aktaş

19515

Post-Laboratory Report

October 14, 2018

Sabancı University

Faculty of Engineering and Natural Sciences

Computer Science and Engineering

Linear Filtering (Gaussian Smooth)

In this question, we are asked to implement a gaussian smooth to an image. This can be achieved as `value = sum(sum(subImg.*k))` function. Idea is applying the gaussian kernel to the image. These numbers are not random and achieved from gaussian distribution. We can achieve a smoothed image by applying this kernel to subimage which is a (2k+1)x(2k+1) matrix.

```
function img2 = lab2gaussfilt(img)

[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end
img2 = zeros(size(img));
img = double(img);
K=2;
k = (1/273)*[1 4 7 4 1; 4 16 26 16 4; 7 26 41 26 7; 4 16 26 16 4; 1 4 7 4 1
];
for i=K+1:1:row-K-1
    for j=K+1:1:col-K-1
         subImg = img(i-K:i+K,j-K:j+K);
         value = sum(sum(subImg.*k));
         img2(i,j)= value;
    end
end

img = uint8(img);
img2 = uint8(img2);
figure;
subplot(2,2,1);
imshow(img);
title('Original Image')
subplot(2,2,2);
imhist(img);
title('Original Image Histogram')
subplot(2,2,3);
imshow(img2)
title('Gaussian Smoothed Image')
subplot(2,2,4);
imhist(img2)
title('Gaussian Smoothed Image Histogram')

end
```

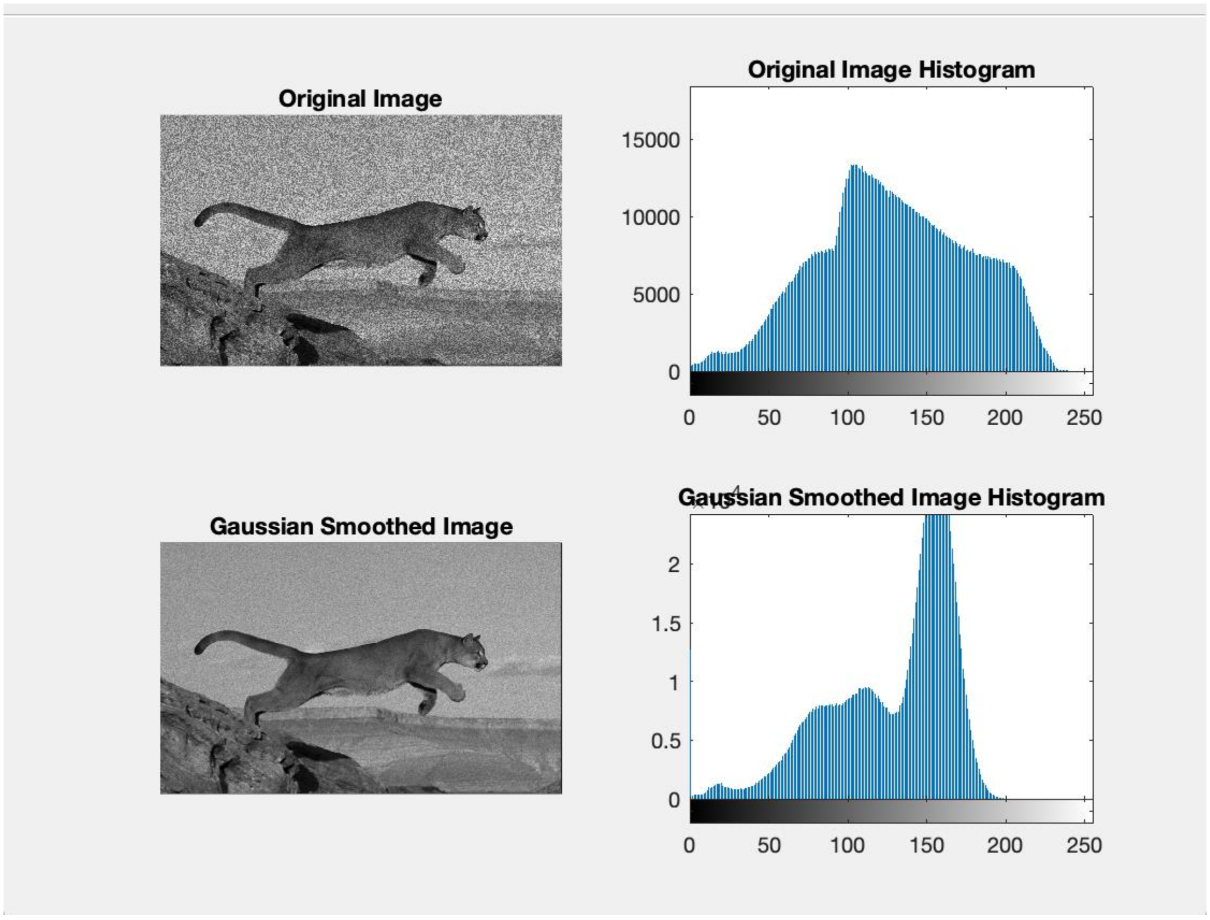

Figure 1. Gaussian Filter with K = 2

In this example, we used 5x5 window and applied gaussian smoothing to ‘jump.png’. Figure 1. shows the new image, which has less noise comparing with first image.

Non-linear Filtering (Median Filter)

In this question, we are asked to implement a median filter. Aim is creating a (2k+1)x(2k+1) matrix and taking the median of the pixel values then changing this values with corresponding pixel. Aim is again reducing the noise and smoothing the image.

```
function img2 = lab2medfilt(img,K)
```

```
[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end
img2 = zeros(size(img));
img = double(img);

for i=K+1:1:row-K-1
    for j=K+1:1:col-K-1
         subImg = img(i-K:i+K,j-K:j+K);
         value = median(subImg(:));
          img2(i,j)= value;

    end
end

img = uint8(img);
img2 = uint8(img2);
figure;
subplot(2,2,1);
imshow(img);
title('Original Image')
subplot(2,2,2);
imhist(img);
title('Original Image Histogram')
subplot(2,2,3);
imshow(img2)
title('Median Filtered Image')
subplot(2,2,4);
imhist(img2)
title('Median Filtered Image Histogram')

end
```

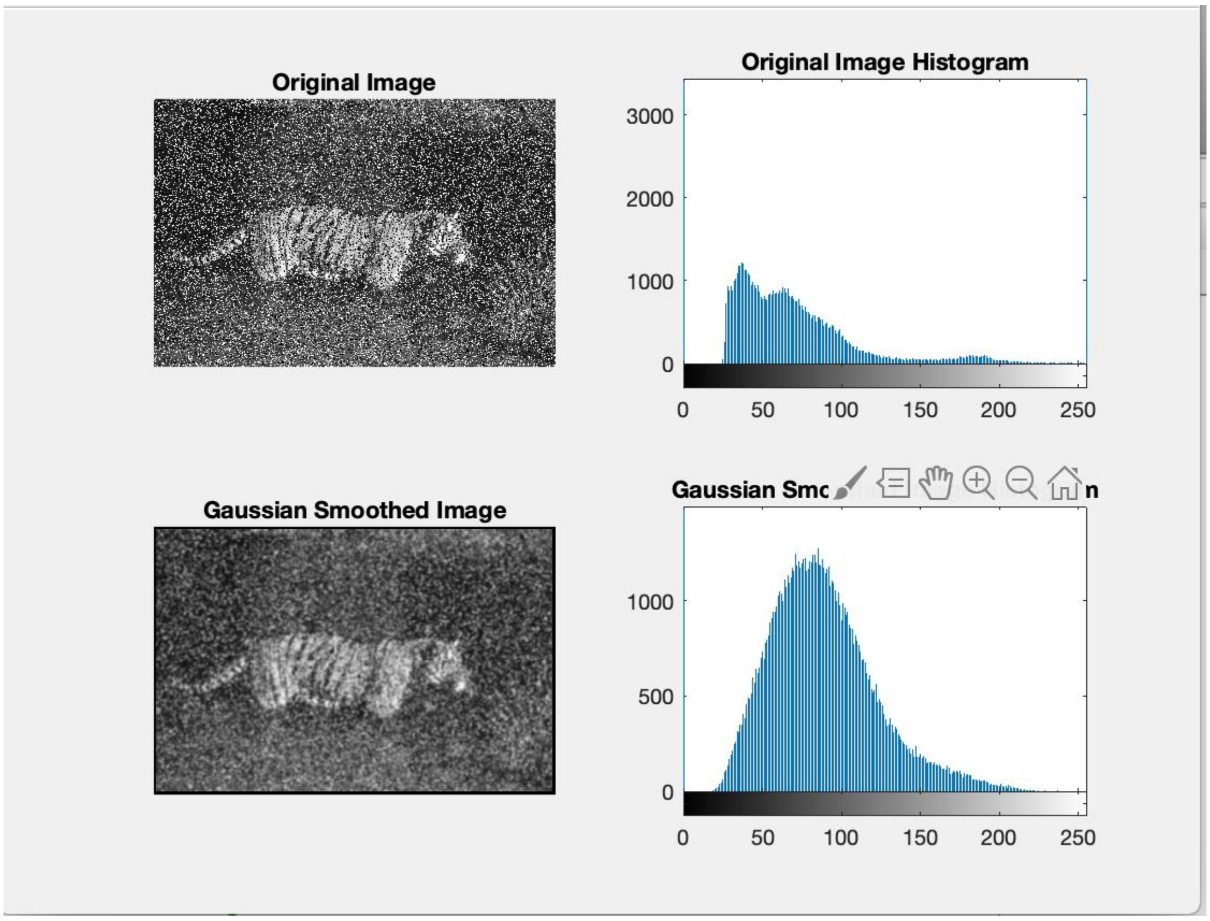

Figure 2. Gaussian Filter with K = 2

Figure 2. shows a tiger image which taken from web and has 50 noise. After applying Gaussian filter we can achieve Figure 2. and we can achieve Figure 3. with using median filter.

As its shown in the figures, both has their noise reduced but median filter made picture smoother.

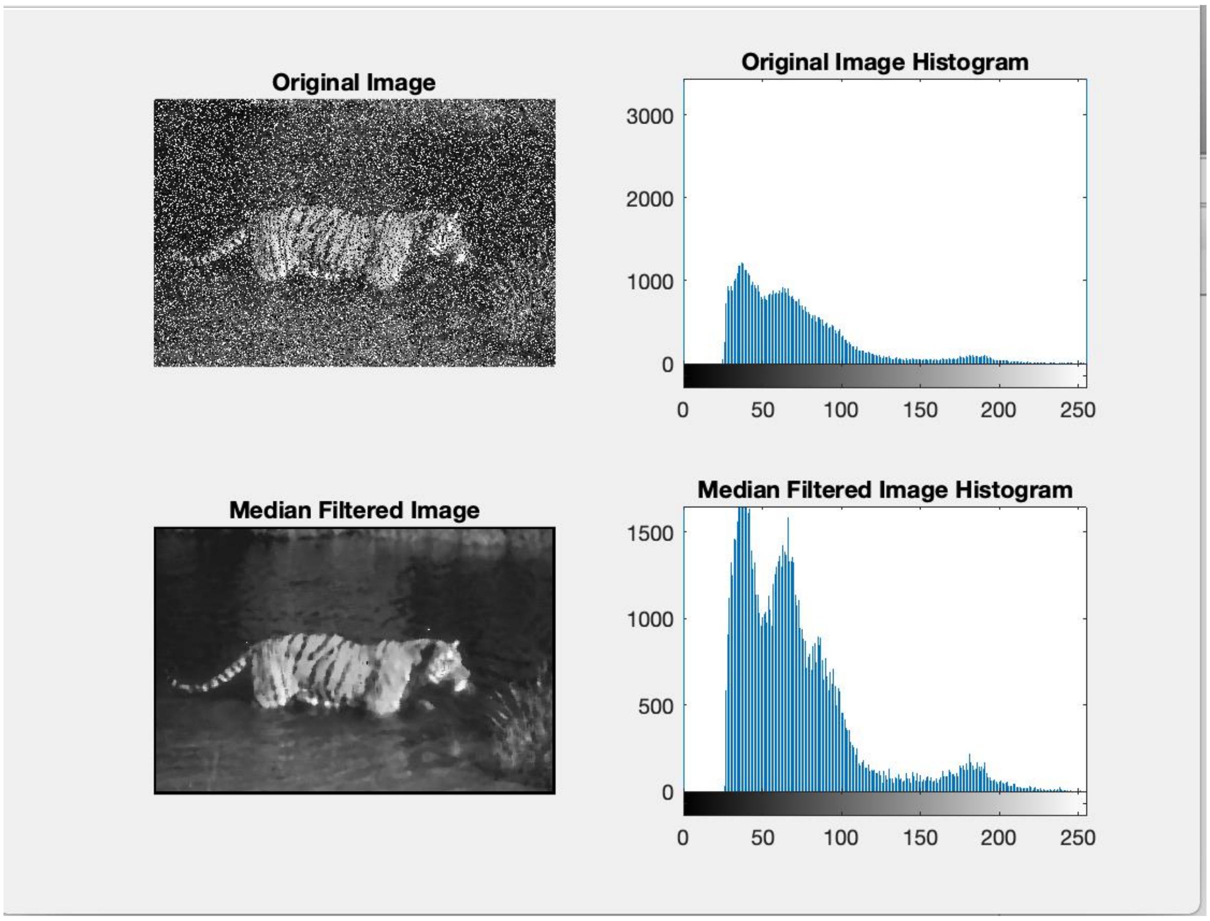

Figure 3. Median Filter with K = 2

Sharpening

In this question, aim was sharpening the image but we reduced the noise by smoothing the picture before sharpening the image. Function takes three parameters; image, lambda and M. M stands for smoothing option box filtering, gaussian smoothing and median filtering respectively.

P.S: Implemented box filtering inside of the for loops because script file was in LAB1 folder.

```matlab
function [img3, img4] = lab2sharpen(img,lamba,M)

[row, col, ch] = size(img);

if(ch==3)
    img = rgb2gray(img);
end
img2 = zeros(size(img));

K=2;
if(M == 1)
    for i=K+1:1:row-K-1
        for j=K+1:1:col-K-1
            subImg = img(i-K:i+K,j-K:j+K);
            meanValue=mean(subImg(:));
            img2(i,j)= meanValue;
        end
    end
end

if(M==2)
    img2 = lab2gaussfilt(img);
end

if(M==3)
    img2 =  lab2medfilt(img,10);
end

img = double(img);
img2 = double(img2);

img4 = img2;
img3 = img - lamba.*(img-img2);

img3 = uint8(img3);
img2 = uint8(img2);
img = uint8(img);

figure;
subplot(1,3,1);
imshow(img);
title('Original Image')
subplot(1,3,2);
imshow(img2);
title('Smoothed Image')
subplot(1,3,3);
imshow(img3);
title('Sharpened Image')

end
```

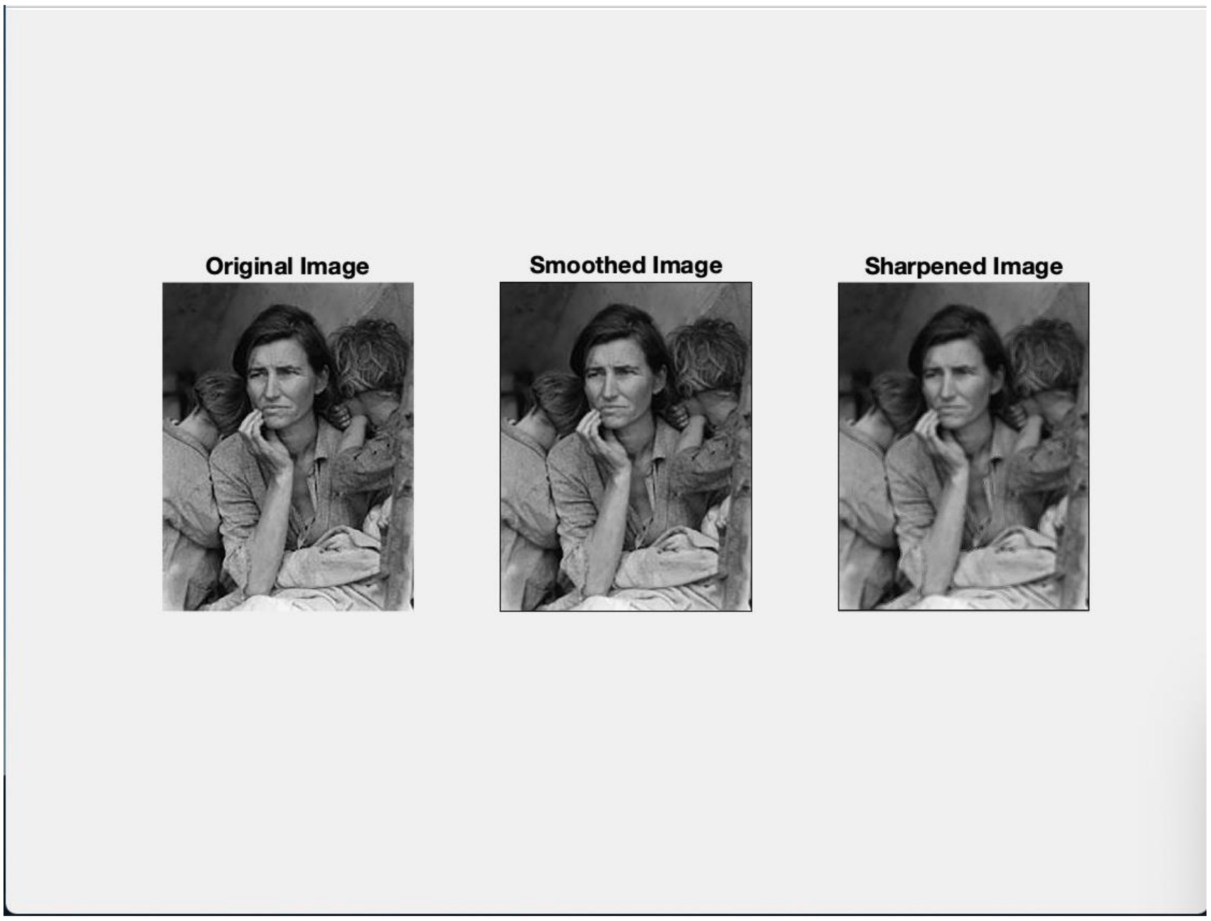

Figure 4. Smoothed with Gaussian Filter, K = 2; Sharpened with Lambda = 5

Figure 4,5,6 shows the ‘mother.png’’s sharpened versions and smoothed versions.

Used window size (2k+1)x(2k+1) = 5 for K = 2; and used Lambda = 5 for all of three methods.

Sharpened image is different in all methods because of the difference of smoothed image.

Different parameters will be tested in following sections.

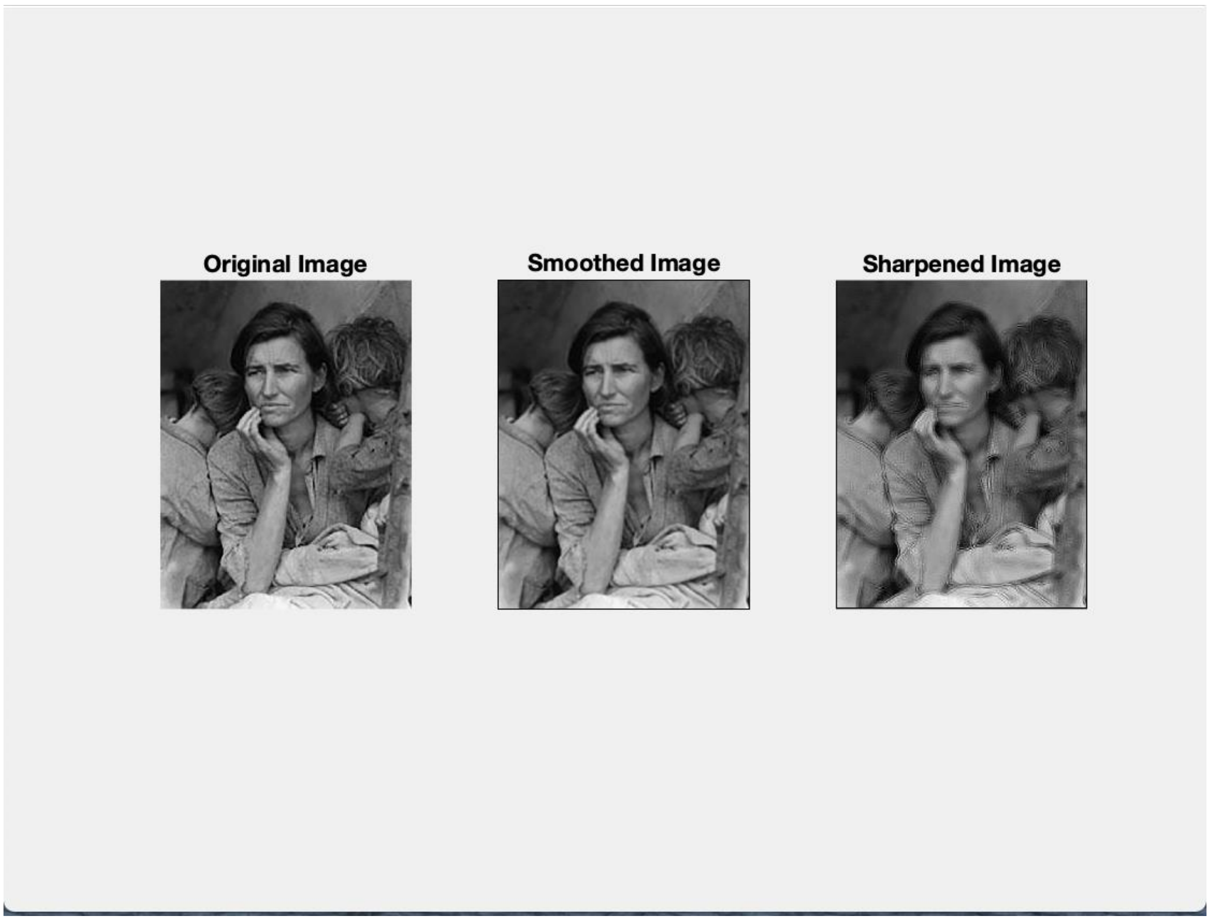

Figure 5. Smoothed with Box Filter, K = 2; Sharpened with Lambda = 5

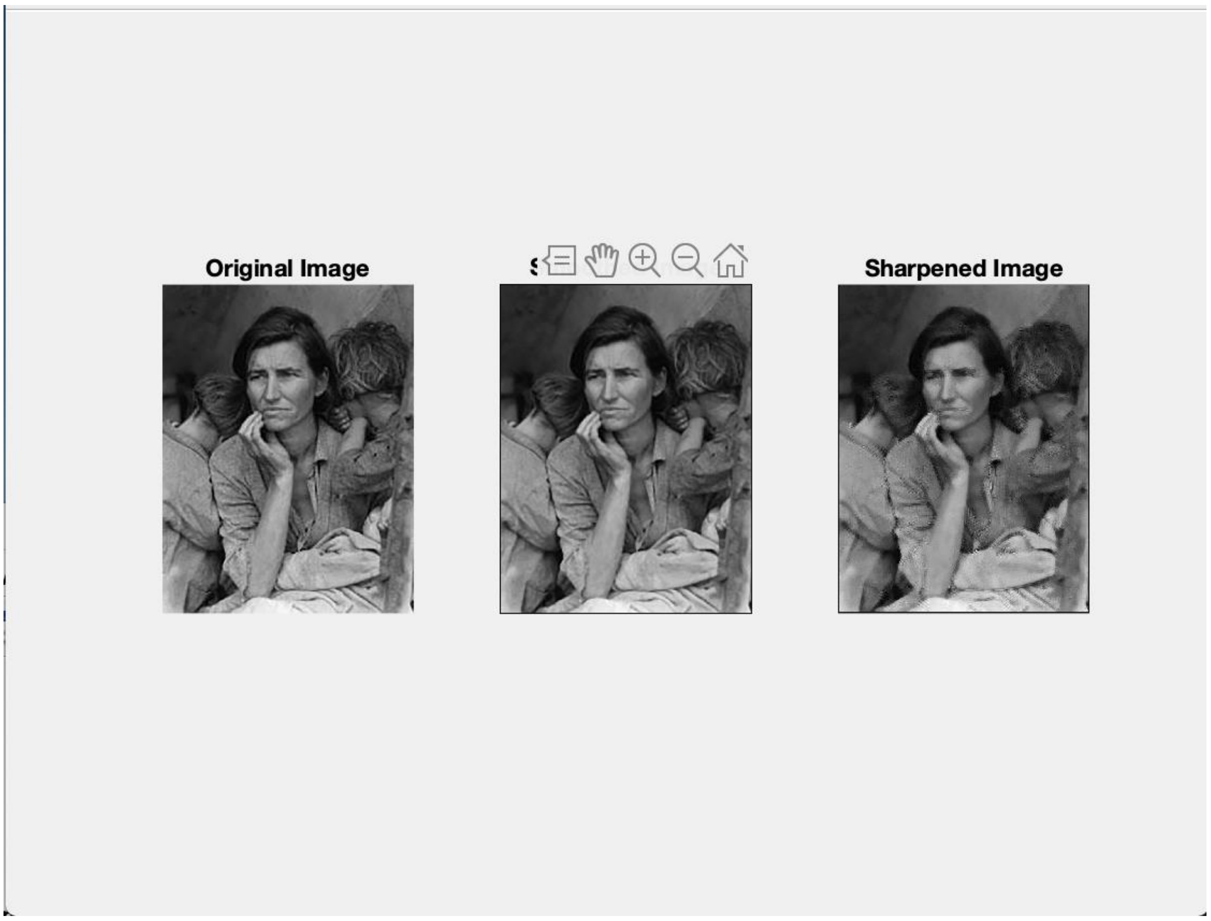

Figure 6. Smoothed with Median Filter, K = 2; Sharpened with Lambda = 5

First Derivative (Sobel Filter)

In this part, we are asked to detect the edges using Sobel Filter which basicly takes the first derivative.

1	2	1
0	0	0
-1	-2	-1

Y Filter (Horizontal)

This mask is used for horizontal filtering. As one can notice middle horizontal row is made from zeros for Horizontal masking. Similar logic applies the Vertical Mask, middle column is made from zeros.

-1	0	1
-2	0	2
-1	0	1

X Filter (Horizontal)

```
function [img2,img3] = lab2sobelfilt(img)

[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end

img = double(img);
K=1;
x = [ -1 0 1 ; -2 0 2; -1 0 1 ];
y = [ 1 2 1 ; 0 0 0; -1 -2 -1 ];
for i=K+1:1:row-K-1
    for j=K+1:1:col-K-1
         subImg = img(i-K:i+K,j-K:j+K);
         valueX = sum(sum(subImg.*x));
         valueY = sum(sum(subImg.*y));
         img2(i,j)= valueX;
         img3(i,j)= valueY;
    end
```

```
end

img = uint8(img);
img2 = uint8(img2);
img3 = uint8(img3);
figure;
subplot(1,3,1);
imshow(img);
title('Original Image')
subplot(1,3,2);
imshow(img3);
title('Sobel Horizontal Image')
subplot(1,3,3);
imshow(img2)
title('Sobel Vertical Image')


end
```

After applying X and Y filters to our image we can see the edges detected. Horizontal mask detects the edges in a horizontal manner as it is shown in the Figure 7. and Figure 8. detects vertical edges.

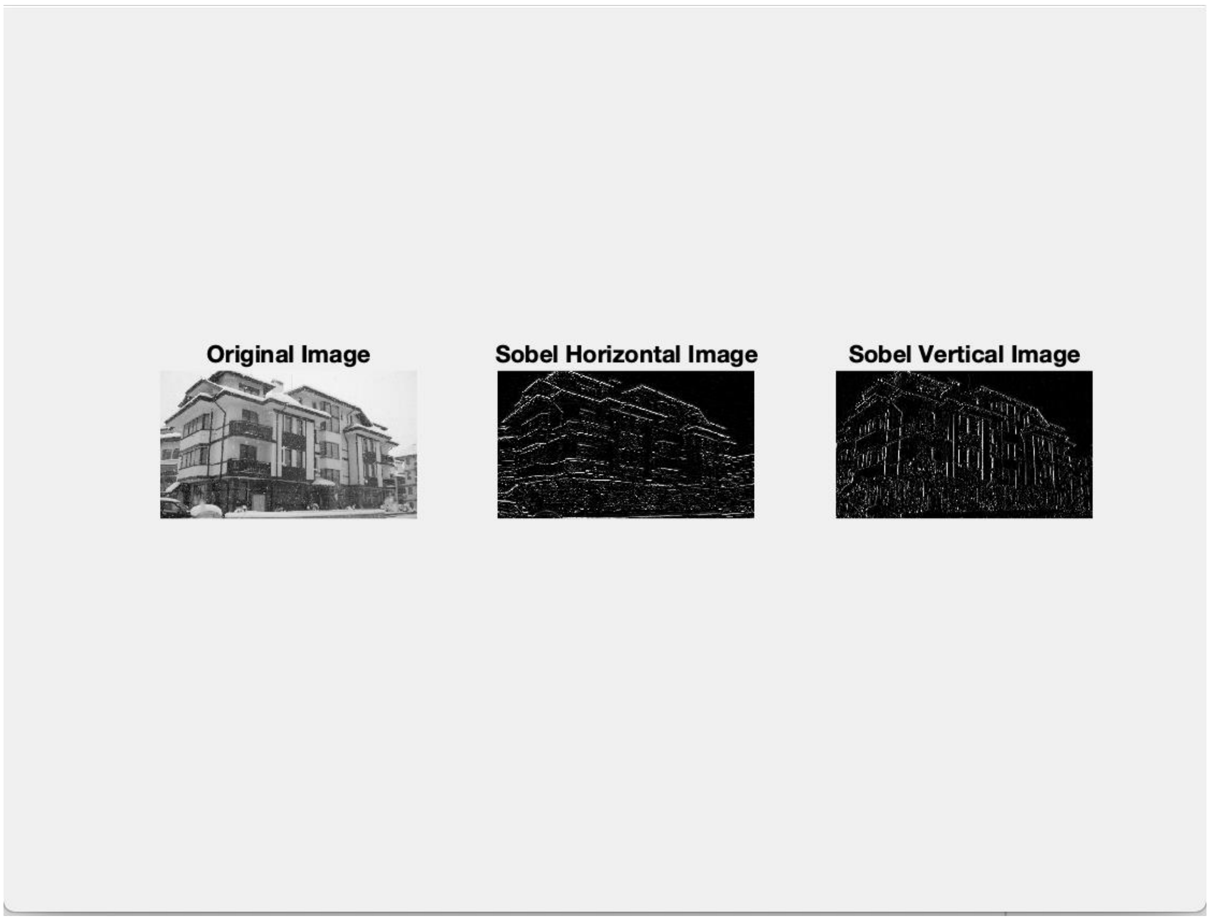

Figure 7. Sobel filter with K = 1

Playing with Filters

In this part, I played with the parameters of filters and sharpening. Aim was playing with window size with changing K and changing the constant lambda value to see its effects on the image.

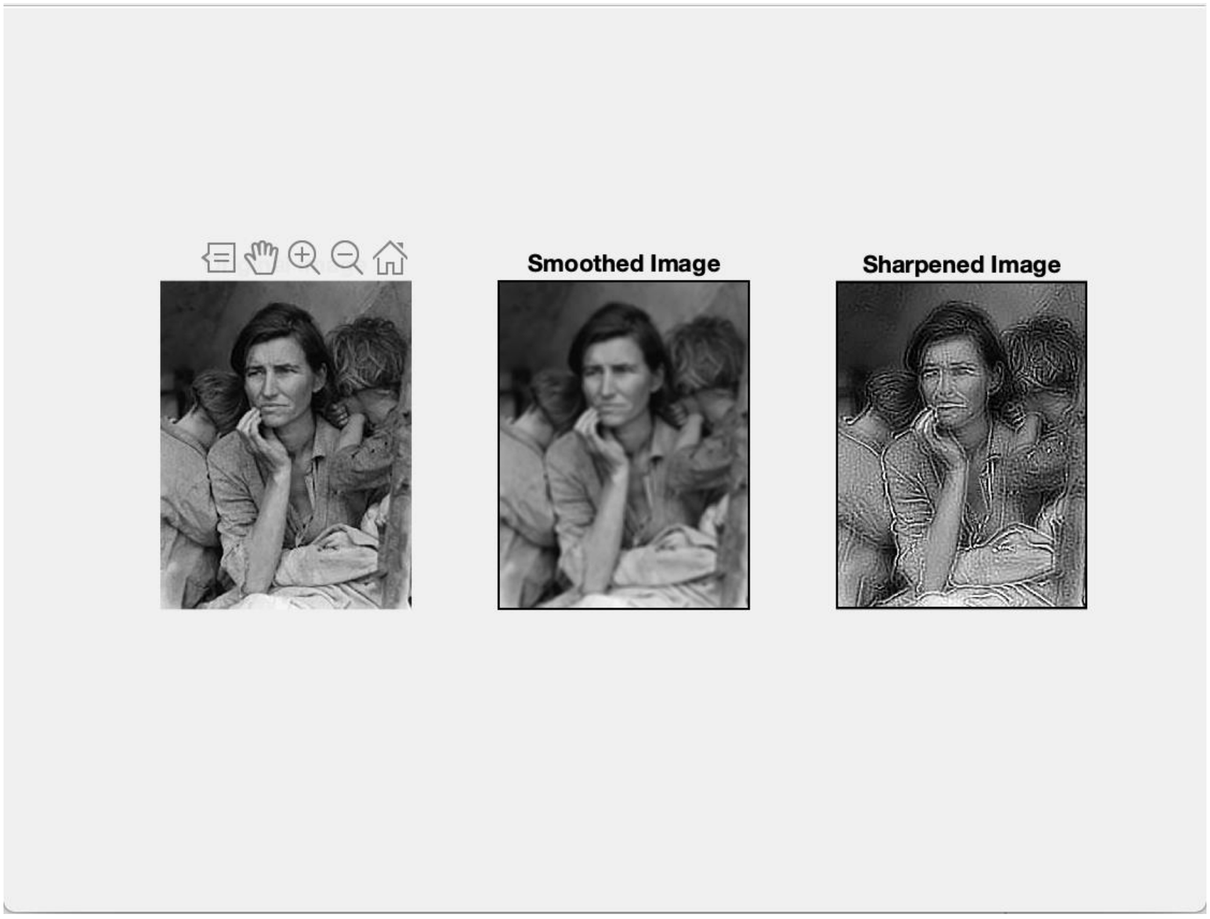

Figure 8. Smoothed with Box Filter, K = 5; Sharpened with Lambda = 5

Figure 8. shows the image with window size 25 and the lambda value 5 using box filter smoothing. If we increase the lambda value directly to 50, as its shown in the Figure 9., we can see the that image is sharpened too much that we can see lots of edges in the image. So we can conclude that if lambda increases, box smoothed image gets more sharpened and edges will be visibile.

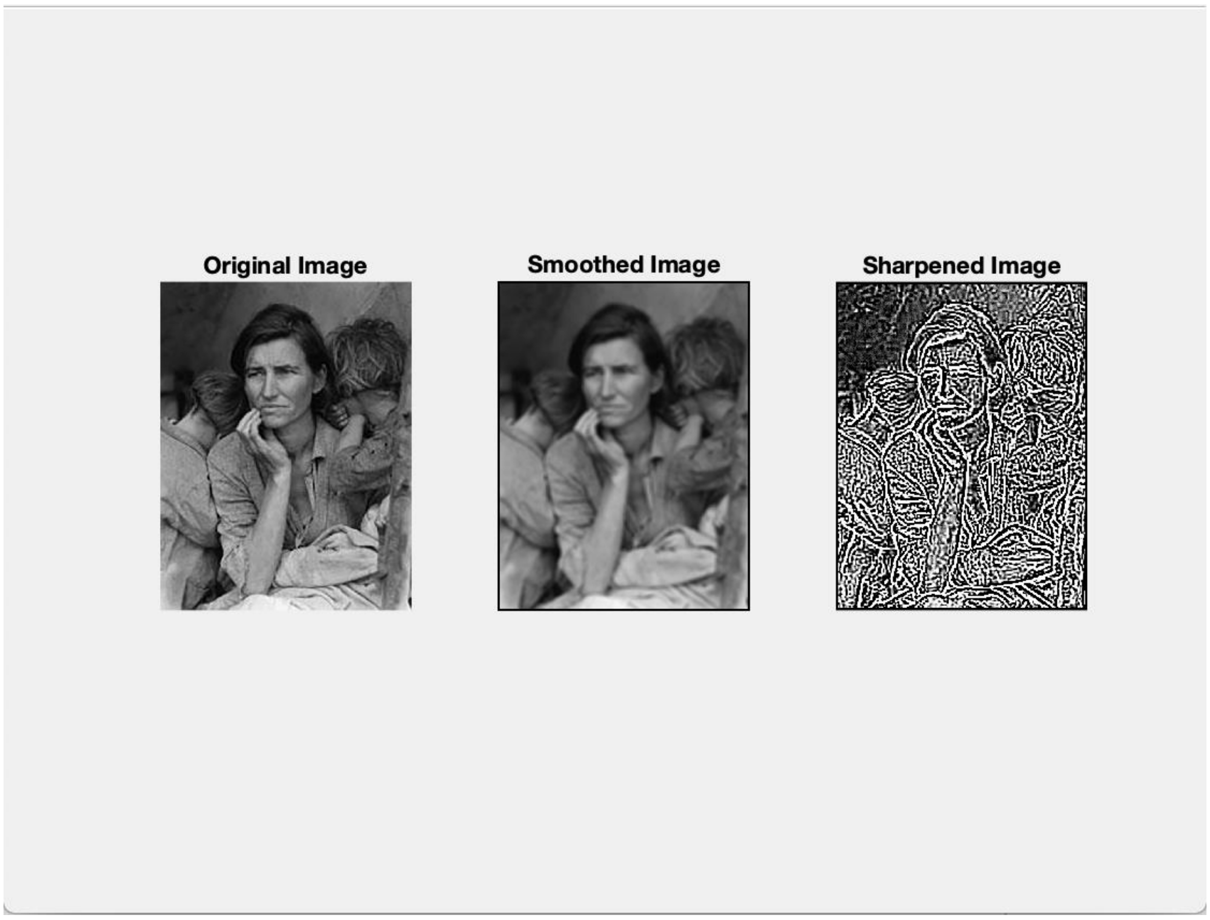

Figure 9. Smoothed with Box Filter, K = 5; Sharpened with Lambda = 50

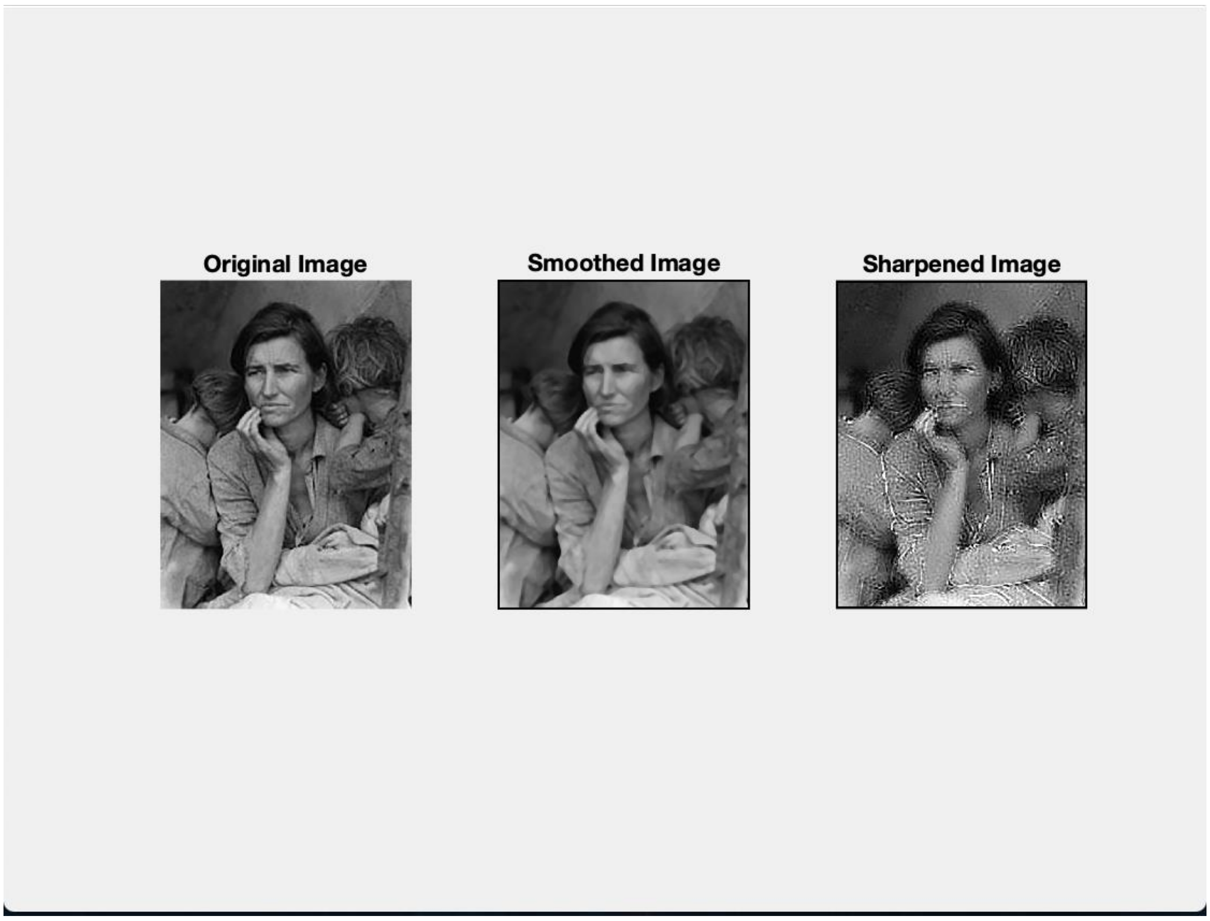

Figure 10. Smoothed with Median Filter, K = 5; Sharpened with Lambda = 5

In this part, I changed the window size of an image which is median filtered then sharpened. Comparing the smoothed image in Figure 10 and Figure 11, we can conclude that having a bigger window size results losing the details of the image after smoothing. Since the Figure 10's image is more smooth and has less details, sharpening it will results us an image with thick edge lines. It is because of smoothing of image. After smoothing that much closer pixels have similar intensity values and this leds to losing details. Thus, sharpening it gives us an image which has thick sharpened edge lines.

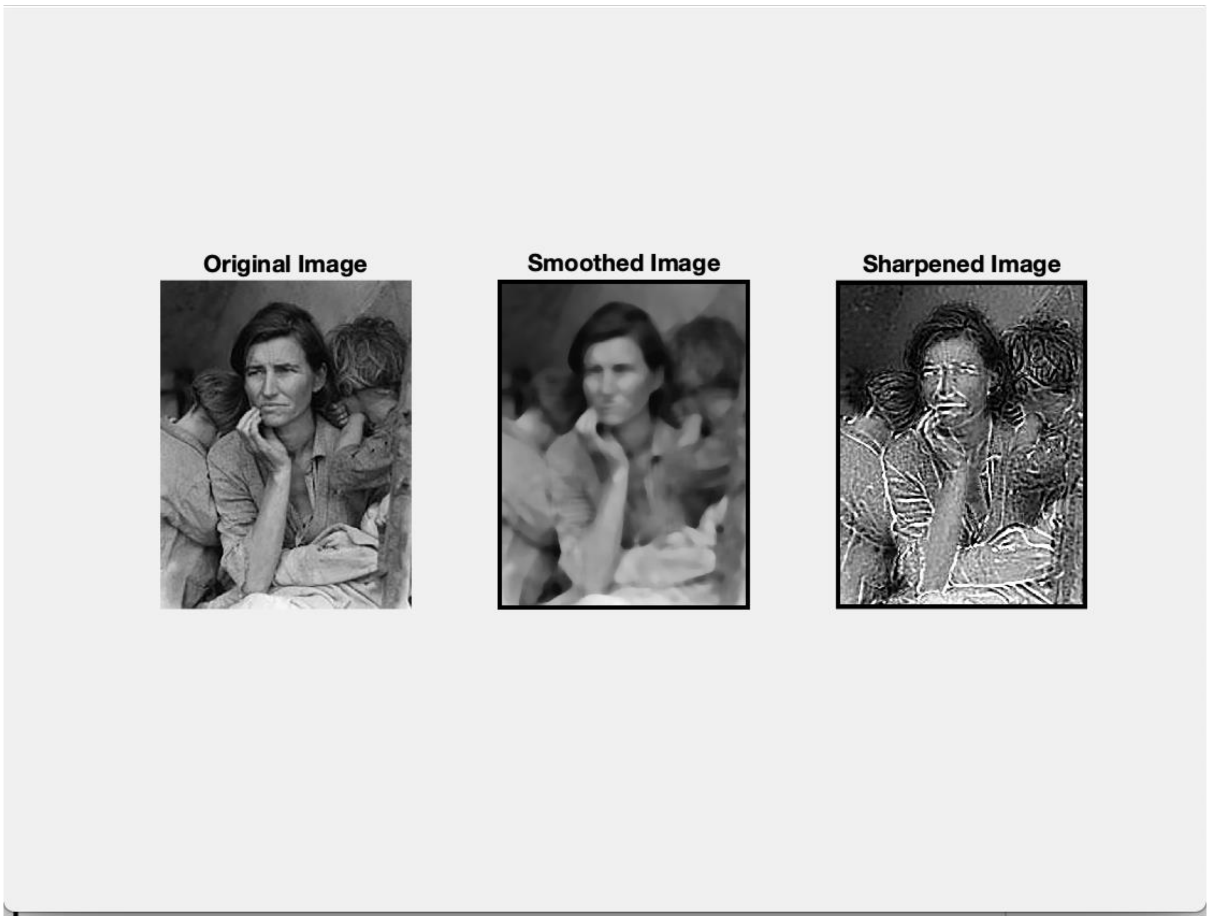

Figure 11. Smoothed with Median Filter, K = 10; Sharpened with Lambda = 5

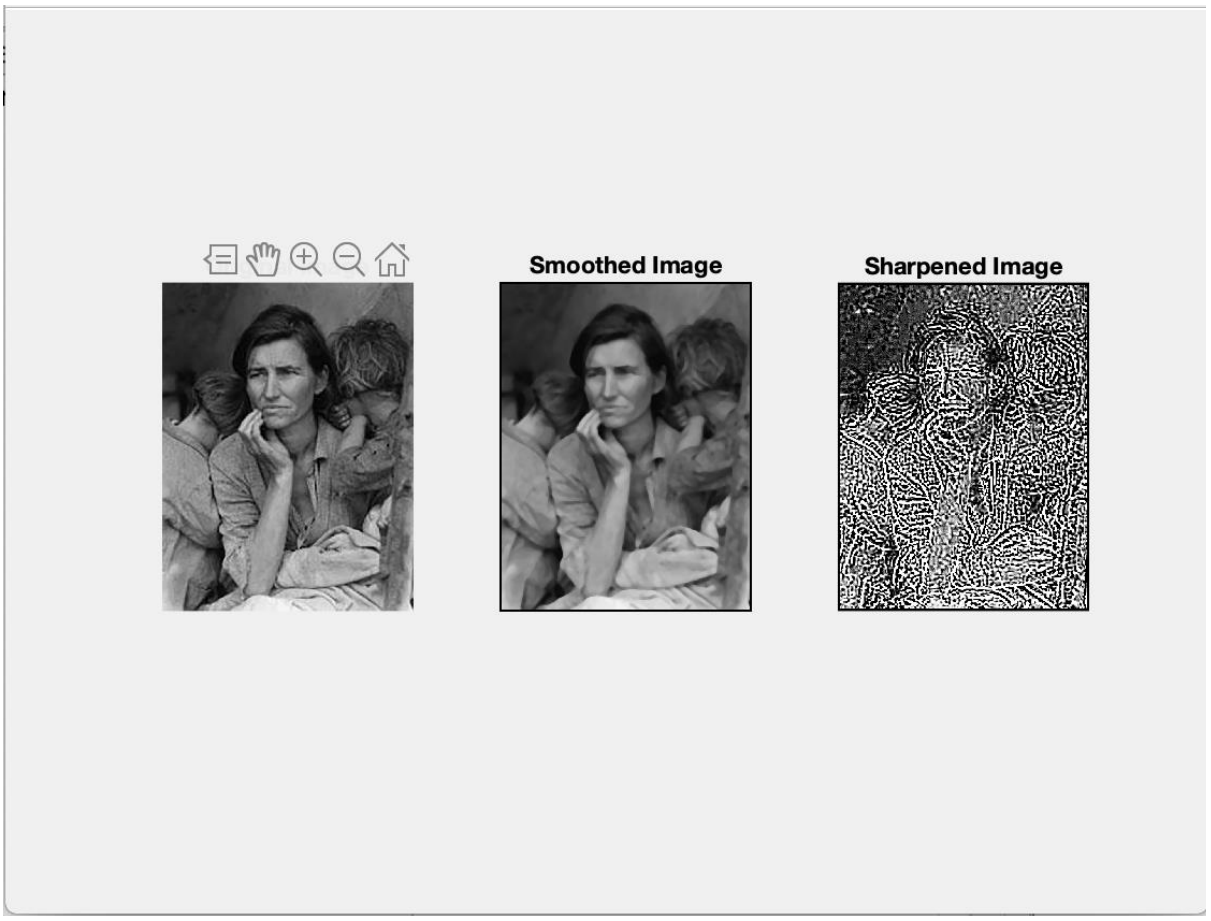

Figure 12. Smoothed with Median Filter, K = 5; Sharpened with Lambda = 50

In this part, I tried to increase lambda value and compare the Figure 12 with Figure 10. Both has same window size but their lambda value is different. This led us have a lots of small edges. After this example I tried increasing K and lambda value simultaneously and the result was having more and more white lines. As lambda increases, the number of white lines increasd and as window size increase the density of white lines increased.