Aynur Süne-21041

EE417 POST-LAB REPORT 2

1. Linear (Gaussian) Filtering

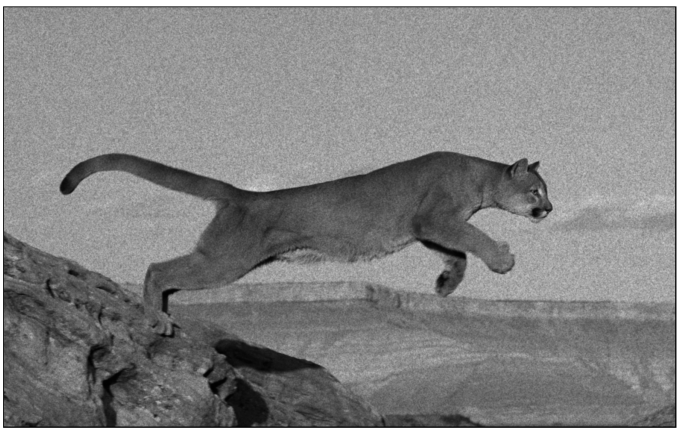
Gaussian filtering is a smoothing operation with a special filter kernel. Image is convolved with this kernel.

```
function [newimg] = lab2gaussfilt(img)

[r,c,color] = size(img);
if(color==3)
    img = rgb2gray(img);
end

newimg = zeros(size(img));
img=double(img);

filter = (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];
k=2;

for i=k+1:1:r-k-1
    for j= k+1:1:c-k-1
        subimg = img(i-k:i+k,j-k:j+k);
        newimg(i,j) = sum(sum(subimg.*filter));
    end
end

newimg = uint8(newimg);
imshow(newimg)
end
```

As in every function, first the image is converted to gray if not already. Then an empty matrix is created for new image that will be returned. Our given image is converted to double. Variable named ‘filter’ is the special filter kernel mentioned. Since we are given a specific matrix, windows created must be the same size with this filter matrix. This matrix is 5x5, since windows are 2k+1 x 2k+1, k must be equal to 2 In double for loop, for every pixel, a window with 2k+1 size is created and this window is convolved with the kernel filter and new pixel value is sum of filtered values. These values are new image’s pixel values. Returning image is converted back to integer.

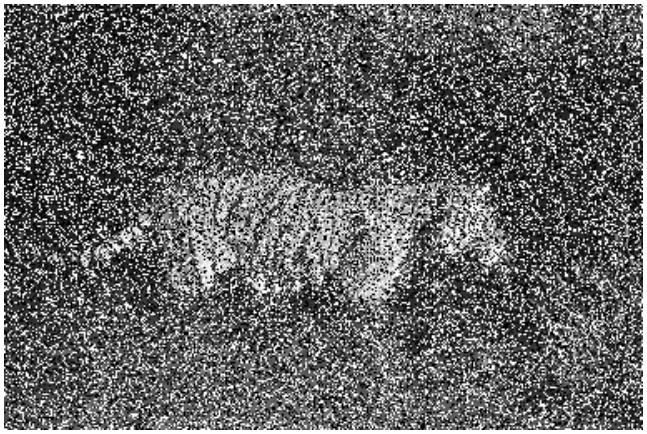
Original Image:

Gaussian Filtered Image:

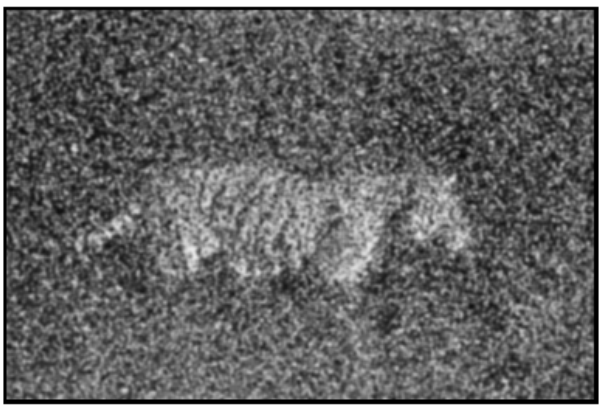


Aynur Süne-21041

Original Image:



Gaussian Filtered Image



Original Image:

Gaussian Filtered Image

Gaussian Filter made the image more blurry and smooth, and noise especially in the background of the first image is reduced. Second image is more explicit about noise reduction.

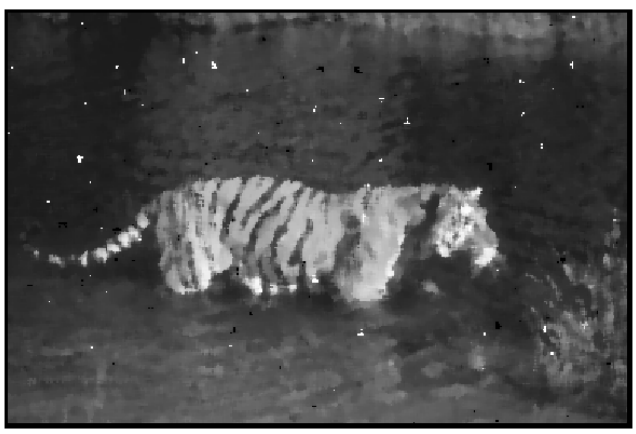
2. Non-linear (Median) Filtering

Median filtering is a non-linear operation because idea is to take the median element of the window and to do that, we need to sort the elements. Sorting is not a linear operation. That is why convolving cannot be used.

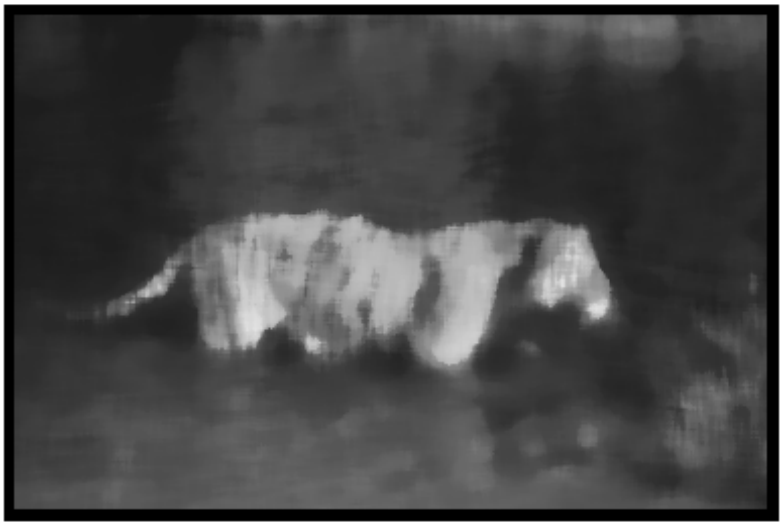
```
function [newimg] = lab2medfilt(img,k)

[r,c,color] = size(img);
if(color==3)
    img = rgb2gray(img);
end

newimg = zeros(size(img));
img=double(img);

for i=k+1:1:r-k-1
    for j= k+1:1:c-k-1
        subimg = img(i-k:i+k,j-k:j+k);
        newimg(i,j) = myMedian(subimg);
    end
end

newimg = uint8(newimg);
imshow(newimg)
end
```

Image is converted to gray if necessary, then new empty matrix with the same size is created to be returned as new image. Given image is converted to double. In double for loop, for each pixel, 2k+1 sized window is created and median element of that window is taken as new pixel value for the new image. Median operation is performed by 'myMedian' function written in the tutorial lab session. At the end, new image is converted back to integer.

Result below is the returned image when parameter k equals to 2. Same parameter with the Gaussian Filtered Image in above example, but this image is way clearer than Gaussian Filtered one. There is a huge difference between these two filters. White dots are places where window elements were mostly white noise pixels. When putting values in an array to pick the median, there is higher chance that the middle will be white or a close color, since array is sorted with respect to pixel values.



This is the output when k equal to 5. When window size got bigger, image become blurry.



3. Sharpening

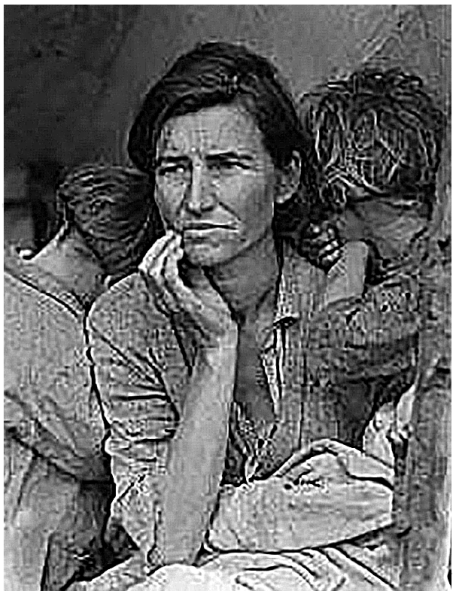
Sharpening is achieved by increasing contrast. Smoothed version of the given image is needed. For this purpose, some functions implemented earlier are used such as Box Filtering, Gaussian Filter, Median Filter.

```
function [newimg] = lab2sharpen(img,lamda, M,k)

[r,c,color] = size(img);
if(color==3)
    img = rgb2gray(img);
end

newimg = zeros(size(img));
smoothed = zeros(size(img));
img=double(img);

if(M==1)
    smoothed = lab1locbox(img,k);
end

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

if(M==3)
     smoothed = lab2medfilt(img,k);
end

smoothed=double(smoothed);
newimg = img+lamda*(img-smoothed);

newimg = uint8(newimg);
imshow(newimg)
end
```

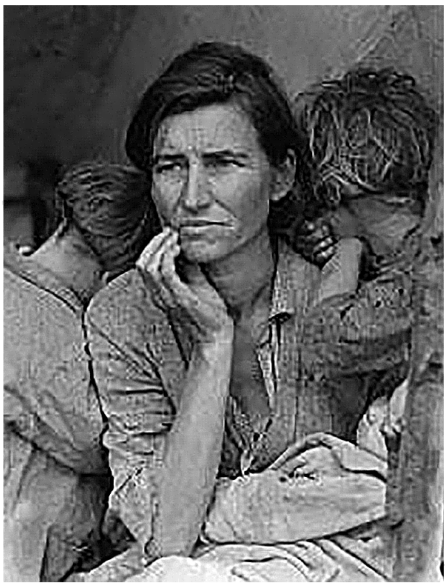
Aynur Süne-21041

As a start, image is converted to gray if not, then two new empty matrices are created for smoothed and returning image. Given image is converted to double. Parameter M is for specifying with smoothing filter to use. If M is given 1, Box Filtering function from first lab is used. If given 2, Gaussian Filter; if given 3, Median Filtering is used. After given image is smoothed and converted to double, formula given in the document is used. Smoothed image pixel values are subtracted from given image pixel values and result is multiplied by a parameter called lambda. Then given image's original values and the result is added for new image. These operations are conducted elementwise and does not require windowing operation. New image is later converted to integer values. Parameter k is for smoothing.

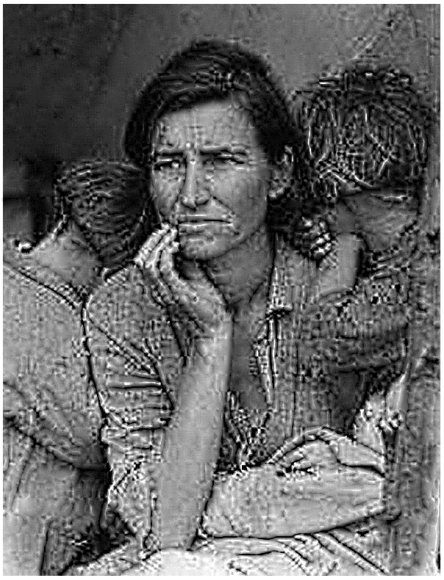
Box Filter with lambda=10, k=2



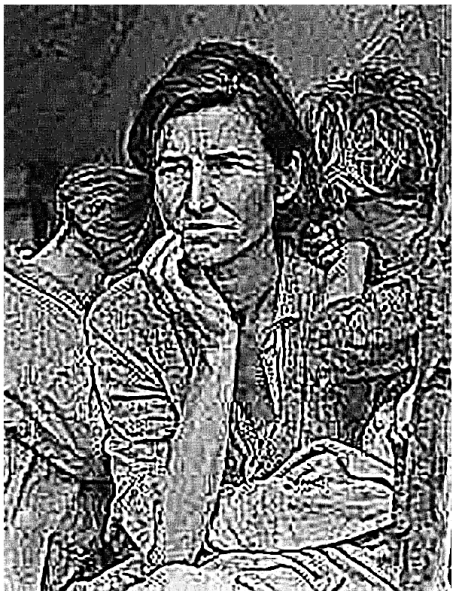
Gaussian Filter with lambda=10, k=2



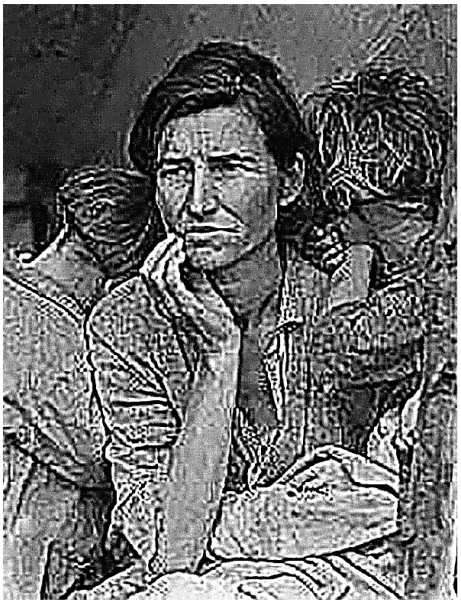
Median Filter with lambda=10, k=2



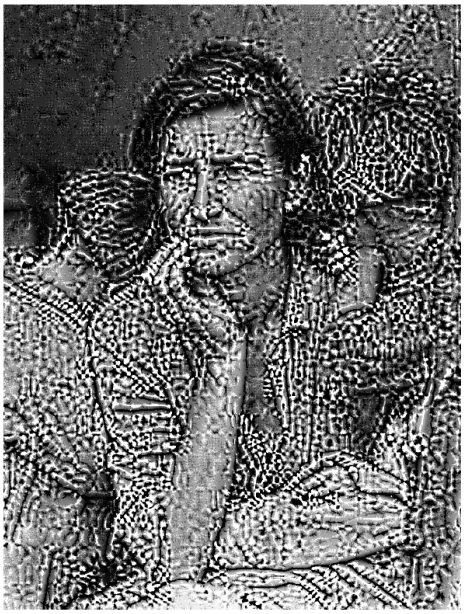
Box Filter with lambda=50, k=2



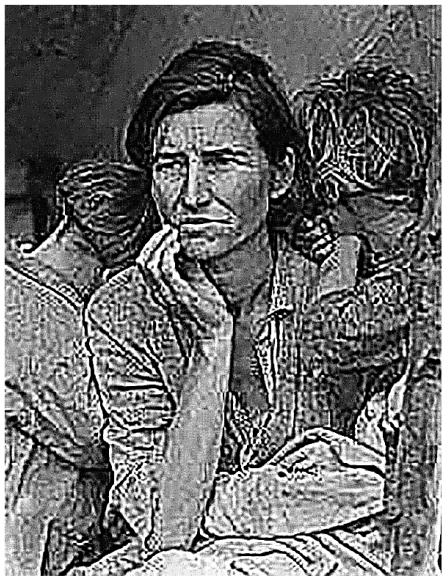
Gaussian Filter with lambda=50, k=2



Median Filter with lambda=50, k=2



Box Filter with lambda=50, k=5 *Gaussian Filter with lambda=50, k=5* *Median Filter with lambda=50, k=5*


With given parameters above; when lambda=10 and k=2, Gaussian Filter was better in terms of noise reduction, but Box filter seems much more sharpened and contrast than others and also noisy. When lambda=50 and k=2, contrast of Box Filter is higher than other two, but Median filter's pixel values are more uniform and image is clear, noise is high. Gaussian seems like a mix of other two filters; contrast is better than Median Filter and noise seems less than others. When lambda=50 and k=5; Box filtered image is again sharper and more contrast than others, noise is high as well. Median Filter is not that contrast but noise is really high. Gaussian Filter increased contrast a little bit and noise is lower than others, but image is not that sharpened, edges are not clear as others.

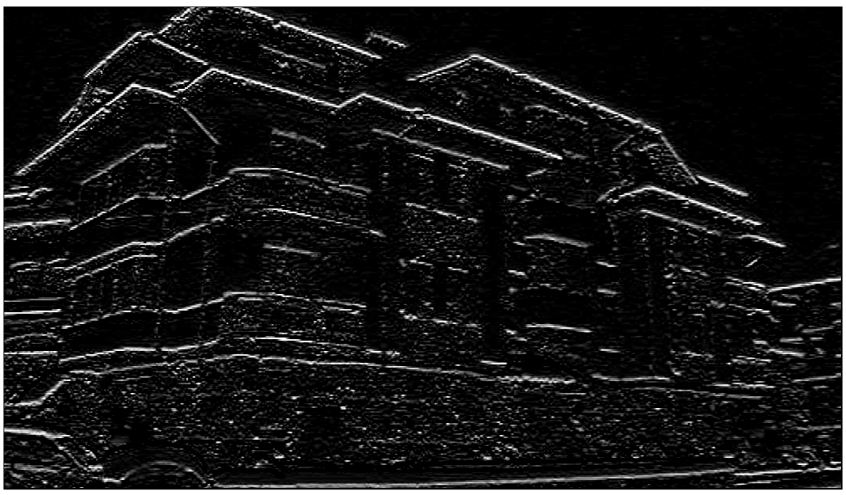
4. First Derivative

Sobel Filter is implemented with two given kernels, one for horizontal filtering and one for vertical filtering. Since those kernel filters are specified and 3x3, k parameter is 1. Sobel kernels are based on first derivative assumptions, first derivative with respect to x and y separately.

```
function [newimgx,newimgy] = lab2sobelfilt(img)

[r,c,color] = size(img);
if(color==3)
    img = rgb2gray(img);
end

newimgx = zeros(size(img));
newimgy = zeros(size(img));
img=double(img);

filterX = [-1 0 1;-2 0 2;-1 0 1];
filterY = [1 2 1;0 0 0;-1 -2 -1];
```

```
k=1;

for i=k+1:1:r-k-1
    for j= k+1:1:c-k-1
        subimg = img(i-k:i+k,j-k:j+k);
        newimgx(i,j) = sum(sum(subimg.*filterX));
        newimgy(i,j) = sum(sum(subimg.*filterY));
    end
end

newimgx = uint8(newimgx);
newimgy = uint8(newimgy);
imshow(newimgx)
imshow(newimgy)
end
```

Implementation is similar to Gaussian Filter, we multiply window with a filter and sum the elements to get the new image's pixel value. Difference is we have two different filters, named filterX and filterY, output is two different images each multiplied with these filters and used for edge detection.
filterY is used for horizontal filtering, output is above:



filterX is used for vertical filtering, output is above: