Image Processing Laboratory

Experiment 3: Spatial Filtering

Report

Group-6

Submitted by

Deep Kiran Shroti (12EC35013)
Vishnu Dutt Sharma (12EC35018)

Contents

1 Introduction

The objective of this experiment is to perform following *Spatial Filtering* operations on an image:

1. Mean Filter
2. Median Filter
3. Gradient Filter
4. Laplacian Filter
5. Sobel Filter

Each of the above filter helps in modification or extracting features.

- *Mean Filter*: This filter is used for high-frequency noise reduction and image smoothing.
  3x3 Kernel =
  $$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$
- *Median Filter*: This is a non-linear filter used for noise reduction. It finds particular use in Salt & Pepper noise reduction.
  For applying it, we find the median of all the values covered my mask and assign it to the center. Unlike mean filter, it preserves edges.
- *Gradient Filter*: As evident by the name, this filtering operation is used to find directional change in intensity in the image. This helps in detection of edges in the image. Edge detection helps in feature extraction and sharpening of the image.
  3x3 Kernel for vertical gradient filter =
  $$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$
  3x3 Kernel for horizontal gradient filter =
  $$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$
- *Laplacian Filter*: Laplacian filter works on the principle of Laplace operator i.e. it uses second-order derivative instead of first order derivative to detect edges.
  3x3 Kernel =
  $$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- *Sobel Operator*: This operator is also used for edge detection. Sobel operator finds more edges or make edges more visible as compared to gradient operator because in Sobel operator we have allotted more weight to the pixel intensities around the edges.
  3x3 Kernel for vertical Sobel operator=

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

  3x3 Kernel for horizontal Sobel operator=

$$\begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

2 Algorithm

We have created a data structure for mask which stores 'size' of the mask and the kernel as a square matrix whose side is equal to the 'size' of the mask. Now for each operator we pick up required kernel and multiply it with the image.

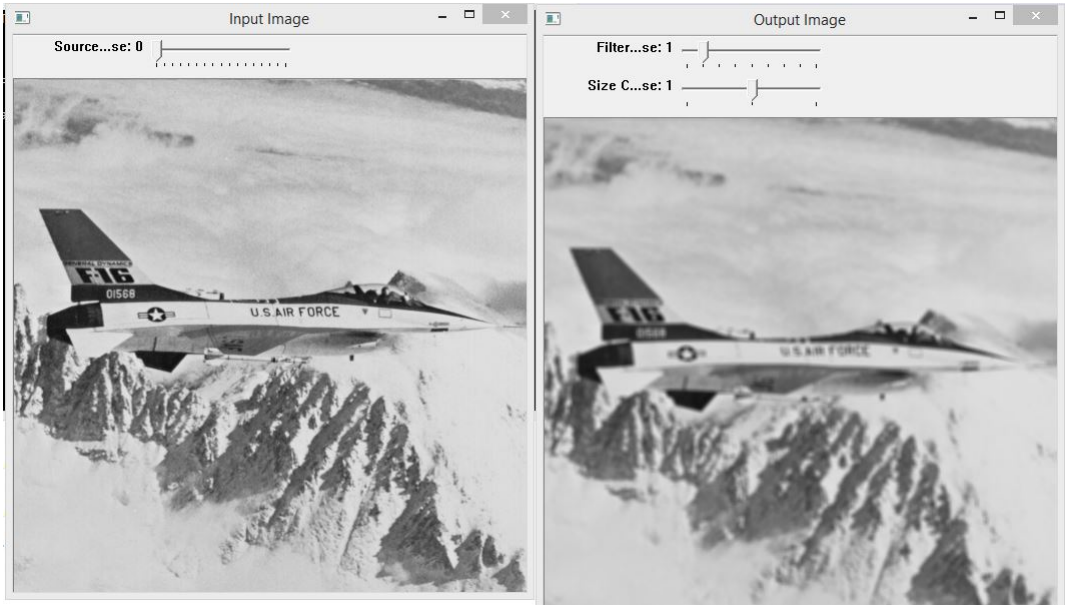
Code for filter application is as follows:

```
Mat general(Mat input, Frame *window){
  // count: number of pixels operated upon
  // sum: value at center pixel after
  // application of the filter
  int i, j, count, sum;
  i = j = count = 0;
  // Get image size
  int new_size = (window->side)/2;

  // Create output image
  Mat output = input.clone();


  for(i = 0; i < input.cols; i++){
    for(j = 0; j < input.rows; j++){
      sum = 0;
      count = 0;
          // Iteration over the masked image
      for(int i1 = -new_size; i1 <= new_size; i1++){
        if(((i + i1) >= 0 ) && ((i + i1) < input.cols)){
          for(int j1 = -new_size; j1 <= new_size; j1++){
                       // Mask application
            if(((j + j1) >= 0) && ((j + j1) < input.rows)){
             sum += window->data[i1 + new_size][j1 + new_size]
                              * input.at<uchar>(i + i1, j + j1);
             count += abs(window->data[i1 + new_size][j1 + new_size]);
            }
          }
        }
      }

          // Limiting the output values
          if(sum > 255)
                 sum = 255;
          else if(sum < 0)
                 sum = 0;

          output.at<uchar>(i, j ) = sum;
          //cout<< sum <<endl;
    }
```

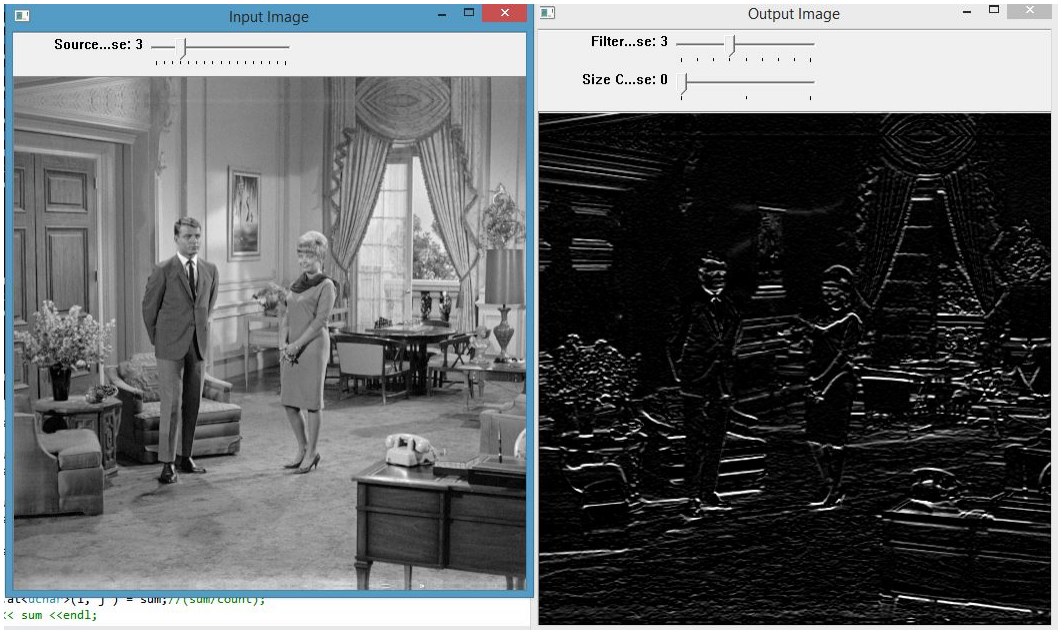
```
    }

    return output;
}
```

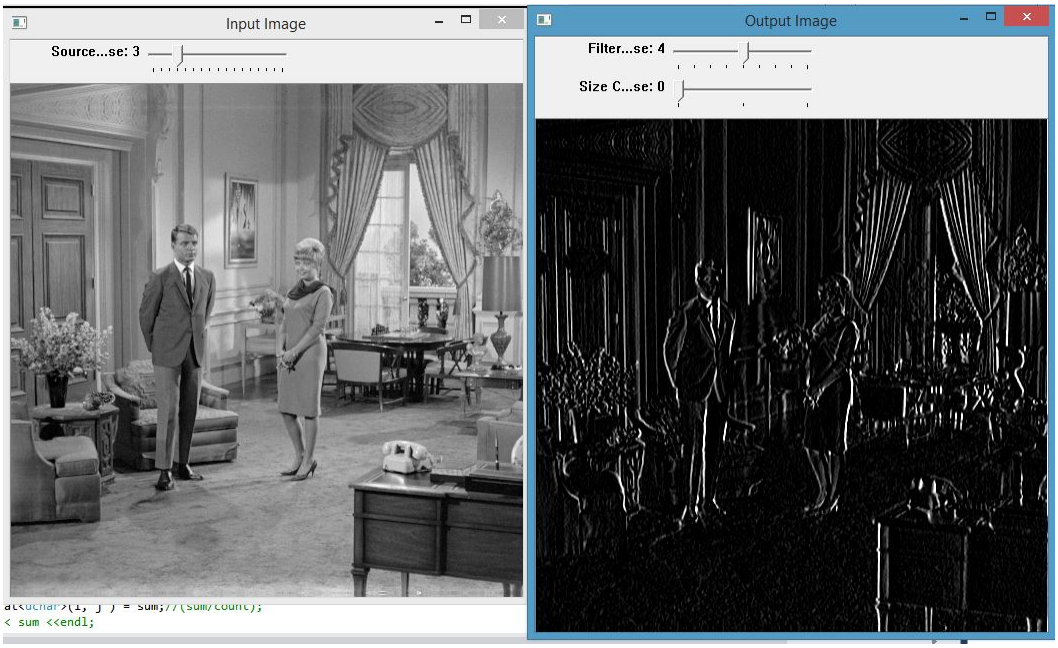
3 Output Results

Following are the results for filtering in the order of filters mentioned above:



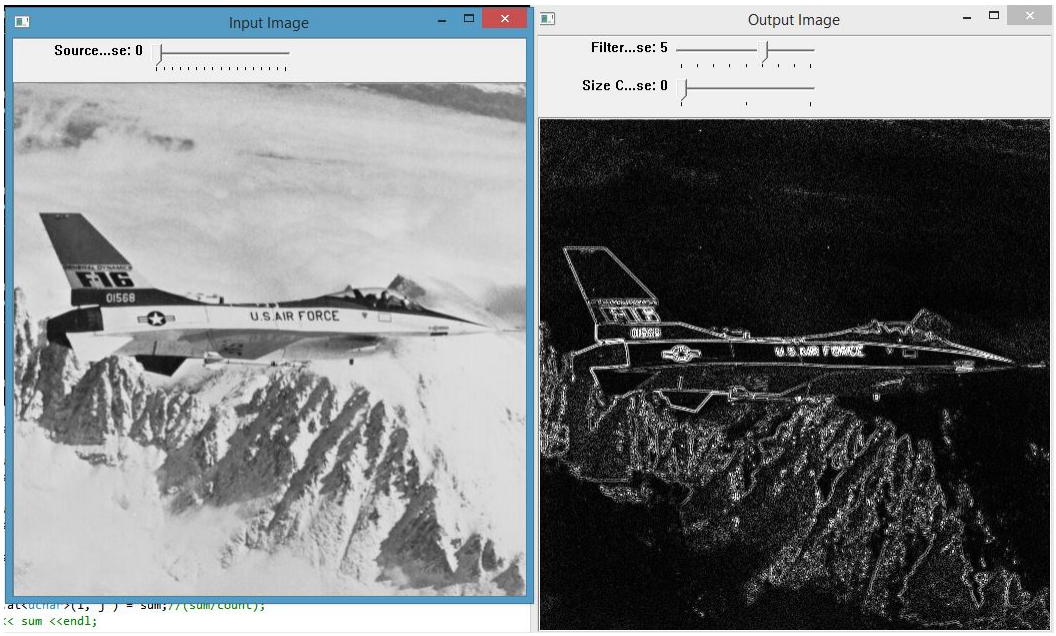

Input Image
Source...se: 9
Output Image
Filter...se: 2
Size C...se: 1

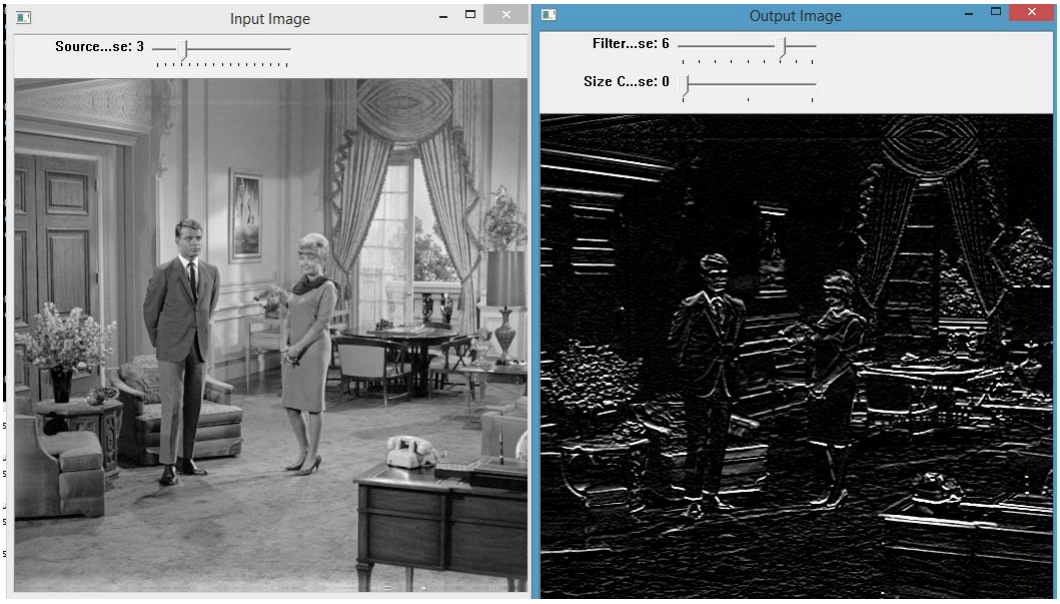

Input Image
Source...se: 3
Output Image
Filter...se: 3
Size C...se: 0

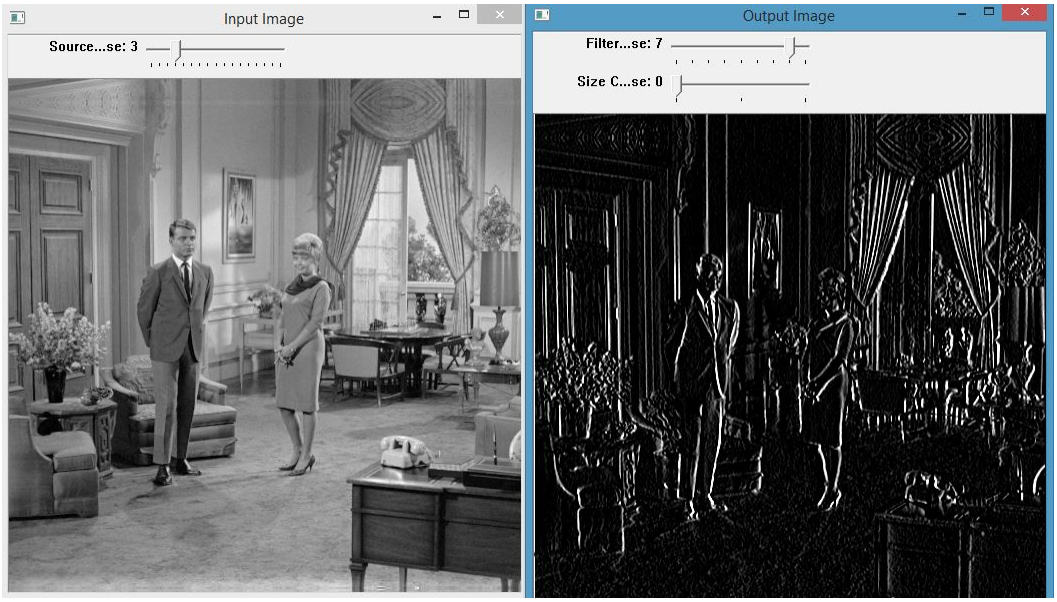


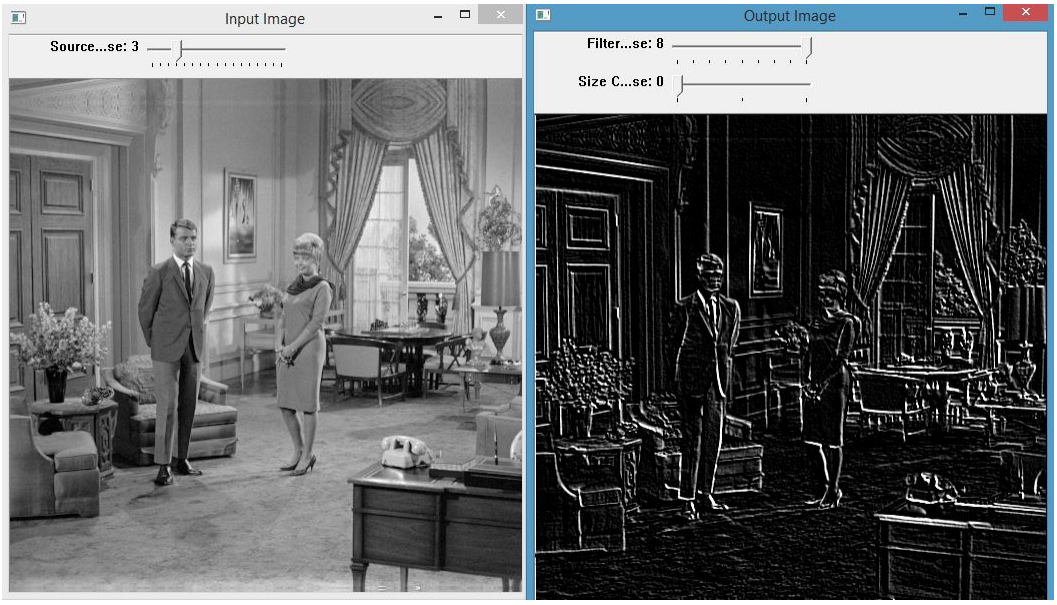
4 Analysis

- Averaging filter acts like a low-pass filter. It helps in reduction of irrelevant details in the image. It creates a smoothing effect but is also smoothes the edges.
- Median filter uses statistical properties of the image. It is very effective in removing impulse noise. It also preserves the edges in the image and thus preserves the details while reducing the noise.
- Gradient operator detects the edges in the image. It uses a first-order derivative and thus at the edges, it produces thick lines as the value goes from zero to negative to zero again at the edges. It has stronger response to gray level step than second-order derivative.
- Laplacian operator uses second-order derivative and thus provides stronger response to thin lines and isolated points. This way it also becomes susceptible to noise. Laplacian is isotropic in nature and thus is rotation-invarient.
- Sobel operator is a weighted first-order derivative. It differentiates in one direction and smoothes edges in other direction. The edge-detected output will be smoothed in this case.


Input Image
Source...se: 0
Output Image
Filter...se: 5
Size C...se: 0
U.S AIR FORCE


Input Image
Source...se: 3
Output Image
Filter...se: 6
Size C...se: 0





5 Sources

[1] AdeptSight User's Guide, Version 3.2.x
http://www1.adept.com/main/KE/DATA/ACE/AdeptSight_ User/ImageProcessing_

Operations.html

[2] Wikipedia page on Median Filter
https://en.wikipedia.org/wiki/Median_ filter

[3] Wikipedia page on Image Gradient
https://en.wikipedia.org/wiki/Image_ gradient

[4] NPTEL course on Image Processing, Module 5.8
http://nptel.ac.in/courses/117104069/chapter_ 8/8_ 26.html

[5] Wikipedia page on Sobel Operator
https://en.wikipedia.org/wiki/Sobel_ operatort