Image Processing

Project 1

Report

Sumit Gupta and Prudhvini Putta

How to Compile and execute

Makefile takes care of compilation. The execution should be done as follows.

./p1 2.1 <input image> <output image>

./p1 2.2a <input image> <output image>

./p1 2.2b <input image> <output image>

./p1 3.1 <input image> <output image>

./p1 3.2 <input image> <output image> <sigma value>

./p1 3.3 <input image> <output image>

./p1 4.2a <input image> <output image> <filter size>

./p1 4.2b <input image> <output image> <sigma value>

./p1 4.3 <input image> <output image> <filter size>

./p1 5 <input image> <modified input image>

./p1 6.1a <input image> <output image> <filter size>

./p1 6.1b <input image> <output image>

./p1 6.2 <input image> <output image> <filter size>

./p1 6.3 <input image> <output image> <filter size>

Part 2: Basic Image Operations

1. Converting Color Image into a grayscale image

Input:

Output:

2) a) Converting grayscale to black and white image

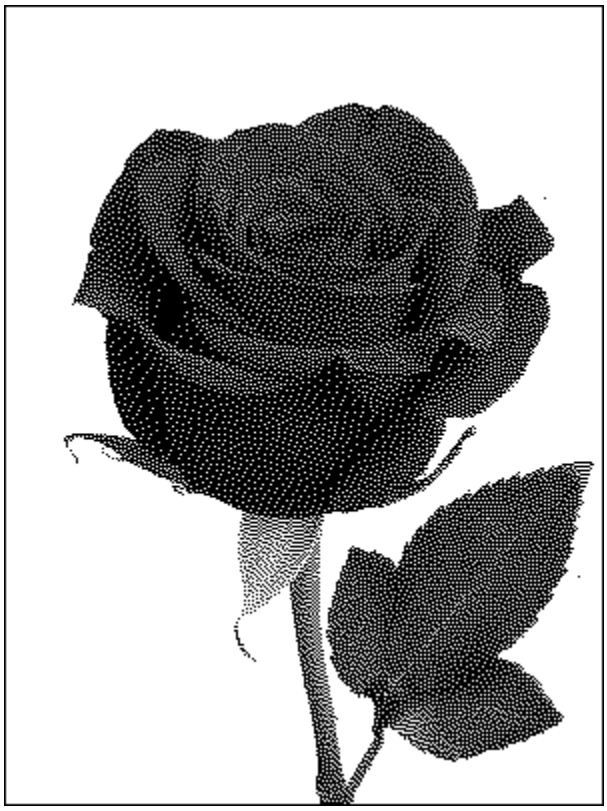
Input: Grayscale Image

Output: Black and white image

2.2 b) Advanced Black and white image

Input:

Output: Advanced black and white image



Part 3: Adding noise

3.1 Uniform Noise

The input image for all part 3 problems is the grayscale image used in the problem number 2.

Output:

3.2 Gaussian Noise

Output: Gaussian noise with sigma = 3

Note: To get good amount of Gaussian noise, we have taken some steps mentioned below:

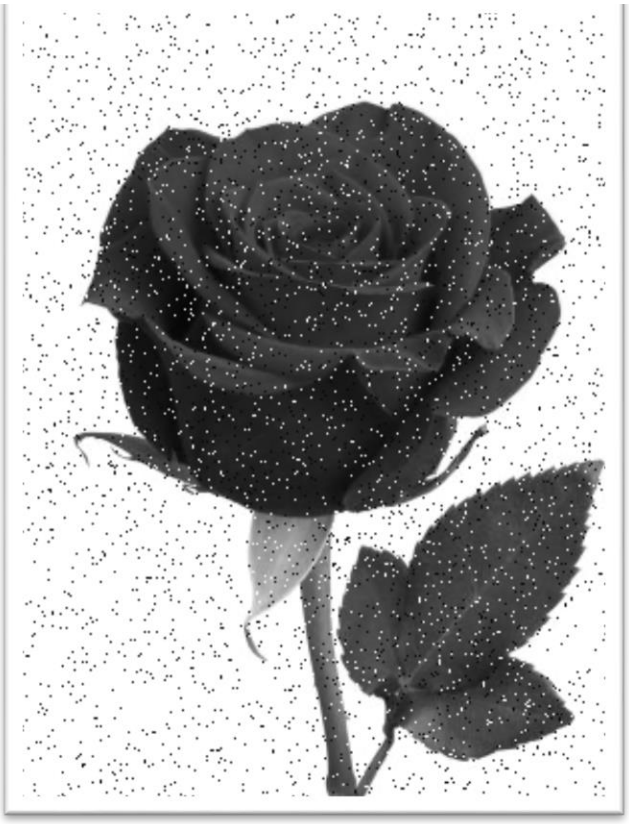
- Since 99.5% of the data in an Gaussian Distribution is within 3 standard deviations (SD), we have randomly sampled the values from a Gaussian Distribution from -3SD to +3SD

- Since the value of our Gaussian distribution is always positive, it will always add a positive noise. To remove this effect, we have taken a random value to choose whether to add or subtract the randomly generated noise value.

- The value of noise is very small to be noticed. We amplified it by multiplying the randomly generated noise value with a factor of 100.

3.3 Salt and pepper noise

Output**:**



Part 4: Filtering

4.1) We developed a generic convolution function `filter()` which will take input image and filter as input and convolves the image and returns the output image. Since the input is either mean filter or gaussian filter, flipping it (horizontally and vertically) is not required. However, we have implemented the code for flipping also. Also our code is generic. It will work for any filter size which is less than the image size. It will work even for non-square filter (i.e. rectangular).

4.2 a) **Box mean filter**

Based on the input size given by the user a filter is created and input image and filter are passed as input to convolution function. The convolution code will return the convolved image as output.

Output: Filter size 3

Output: Filter size 5

4.2 b) Gaussian Filter

The sigma value will be taken as input and Gaussian filter will be created. The Gaussian filter and input image are passed to convolution function which returns the convolved output image.

Output: Gaussian Filter using sigma value 1

Output: Gaussian Filter using sigma value 2

4.3) Median Filter

Since we need to find the median of the values within a square box of given size, there is no Kernel for this. We just need to assume a imaginary box of the given size and loop though the image to store the values in an array to sort them. Then we need to find out the median.

Creating the kernel for median filter is not possible. There is no one filter that on convolution will give the Median.

Output: Median filter convolution using neighborhood size 3

Note: note that it removed the salt-and-pepper noise.

Output: Median filter convolution using neighborhood size 5

We noticed that as the size of Median filter increases, it generates a water color painting type effect.

Part 5: Experiments

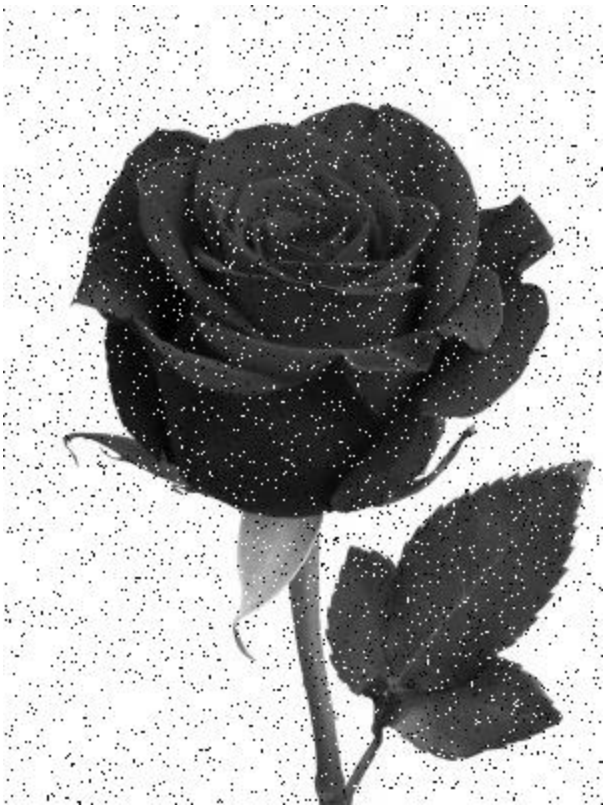
Qualitative Tests results

For qualitative tests following are the input images used

1) Input image with uniform noise

2) Input image with Gaussian noise

3) Input image with salt and pepper noise



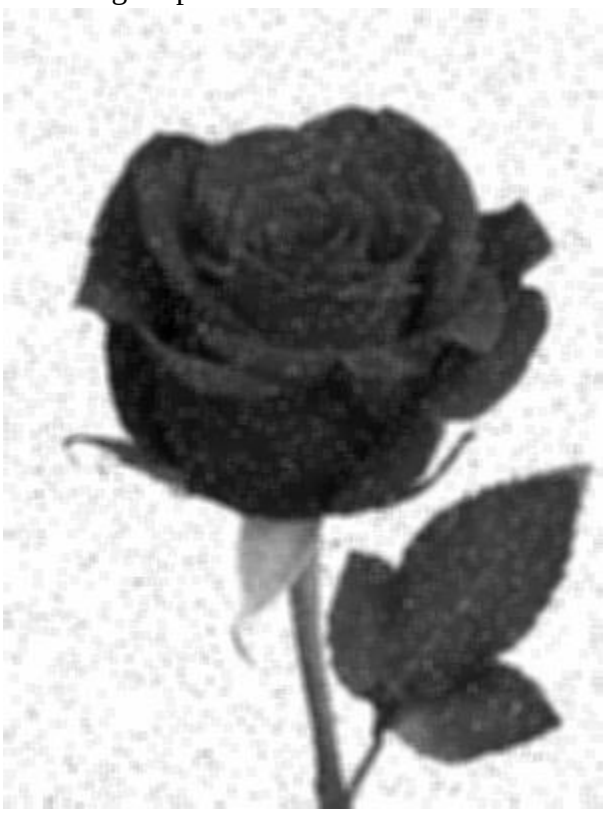
1) Applying mean filter with filter size 3 to remove uniform noise

Output: After applying mean filter with filter size 3 gave the following output

Output: After applying mean filter with filter size 5 gave the following output

2) Applying mean filter on image with gaussian noise generated with sigma value 2 gives

3) Applying mean filter with filter size 5 on image with salt and pepper noise gave the following output



4) Applying Gaussian filter on an input image with uniform noise gave the following output

5) Applying Gaussian filter on an image with Gaussian noise gave the following output

6) Applying Gaussian filter with sigma 2 on image with salt and pepper noise gave the following output

7) Applying median filter with neighborhood value 3 on image with uniform noise gives

8) Applying median filter with neighborhood value 3 on image with gaussian noise gave the following input

9) Applying median filter with neighborhood value 3 on image with salt and pepper noise gave the following input

Analysis:

We have considered mean filter with filter size 5, Gaussian filter with sigma value 3 and Median filter with neighborhood size 3

- After observing all the outputs we can see that median filter is working perfect for removing salt and pepper noise (well as far as our eyes can see!).
- Gaussian filter appears to work best with images with Gaussian noise.
- Gaussian filter appears to work best for images with uniform noise.

Quantitative Analysis

For every combination of filter and noise following are the mean squared error values found.

1) Mean filter with uniform noise
   Mean Square Error = 49900.7
2) Mean filter with gaussian noise
   Mean Square Error = 81936.1
3) Mean filter with salt and pepper noise
   Mean Square Error = 108950
4) Gaussian filter with uniform noise
   Mean Square Error = 114832
5) Gaussian filter with gaussian noise

Mean Square Error = 110519

6) Gaussian filter with salt and pepper noise
   Mean Square Error = 118740
7) Median filter with uniform noise
   Mean Square Error = 39956.9
8) Median filter with guassian noise
   Mean Square Error = 45616
9) Median filter with salt and pepper noise
   Mean Square Error = 48897.7

From qualitative analysis we can see that median filter is best for removing any kind of noise.

But the above results are taken with filters of different sizes. Qualitative and Quantitative results might match if filter sizes in all the three filters is same but as we have taken different filter sizes we observed a slight difference between outputs from both qualitative and quantitative analysis.

We observed that according to quantitative results median filter works best for any kind of noise but we took neighborhood size of median filter as 3. Therefore according to our analysis Median filter works best for any kind of noise.

Part 6: Speed up

6.1a) Separable Mean Filter

Depending the input filter size, we are creating 2 filters one horizontal filter and one vertical filter. In the code we are passing input image and vertical filter as input to a function which convolves the input image and returns the result. The output of first step along with horizontal filter is sent as an input to the same function again. The output of second step gives us the desired output.

Output from separable convolution mean filter with filter size 11 is

Separable mean filter convolution process took **98 ms**.

Output from normal mean filter with filter size 11 is

Normal mean filter convolution process took **230 ms.**

Speed boost = 60.86%

Therefore we can see that separable convolution mean is more than twice fast compared to normal mean filter (in this particular case).

6.1 b. Separable Gaussian Filter

We are creating two filters one horizontal filter and vertical filter which is a transpose of horizontal filter. We will first convolve the image by passing vertical filter and input image. We will pass the output of previous step and horizontal filter as input to the same function we used before which gives the required output. In this step the filter size is constant and is equal to 9 and sigma is equal to 3.

Output from separable Gaussian filter with filter size 9 is

Time taken = **40 ms**

Output from normal Gaussian filter with filter size 9 is

Time taken = **150 ms**

We can see that separable Gaussian filter took less time than normal Gaussian filter.

Speed boost = 73.3%

6.2) Dynamic separable box mean filter

We are making use of dynamic programming by adding rightmost pixel to and subtracting leftmost pixel from the accumulated sum at a particular pixel.

By using dynamic programming and filter size 11 dynamic separable mean filter output is as follows

Time taken = **10 ms**

Output from normal mean filter with filter size 11 is

Normal mean filter convolution process took **230 ms.**

We can see that mean filter takes 23 times more time compared to dynamic separable mean convolution.

Speed boost = Dynamic Separable Mean filter is 95% faster compared to mean filter.

6.3) Cascaded Gaussian Filter

In this problem we are convolving an input image with dynamic separable box filter that we created in 6.2. We are repeating the convolution for 3 times and following is the output image with mean filter size 9

Time taken = **50 ms**

Gaussian filter with sigma value 3 gives the following output

Time taken = **150 ms**

Speed boost = Cascaded Gaussian filter is 66.6 % fast.

Mean square error between the image convolved with Gaussian filter with standard deviation 3 and cascaded Gaussian filter with filter size 11 is 868230. The huge mean squared error value might be because of big filter size.