আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Lab Report

Department: Computer Science and Engineering.

Course Code: CSE-4876

Course Title: Pattern Recognition & Image Processing

Submitted By:

Name: Farida Nusrat

ID: C201242

Semester: 8th

Section: 8AF

Submitted To:

Mr. Mohammad Mahadi Hasaan

Associate Professor,

Dept. of CSE, IIUC.

Date of Submission:26 March,2024

Lab No: 06

Lab Title:

Smoothing filtering
- Low pass filtering
- Median Filtering
- Max filtering
- Min filtering
- Midpoint filtering

Introduction:

Smoothing filters are essential tools in image processing used to reduce noise while preserving important features of an image. In this lab, we explore five different smoothing filters: low pass filtering, median filtering, max filtering, min filtering, and midpoint filtering. Each filter has its unique characteristics and applications, which we investigate through experimental analysis.

- **Low Pass Filtering:**
  Averaging filter (3x3) mask was developed.
  The mask was convolved over the image to perform low pass filtering.
- **Median Filtering:**
  For every 3x3 area in the image, the median of the pixels was calculated.
  The center pixel was replaced by the median value.
- **Max Filtering:**
  For every 3x3 area in the image, the maximum pixel value was found.
  The center pixel was replaced by the maximum value.
- **Min Filtering:**
  For every 3x3 area in the image, the minimum pixel value was found.
  The center pixel was replaced by the minimum value.
- **Midpoint Filtering:**
  For every 3x3 area in the image, both the minimum and maximum pixel values were found.
  The midpoint between the minimum and maximum values was calculated, and the center pixel was replaced with this value.

Methodology:

The Python code for implementing each smoothing filter is provided below. Each filter is applied to a sample image, and the results are visually compared to assess their effectiveness.

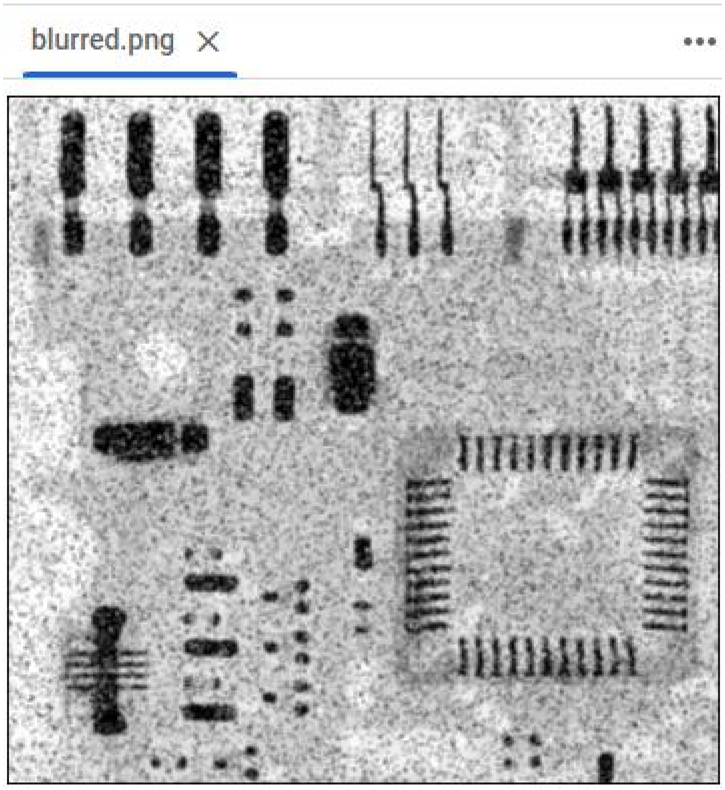
Smoothing Filters Implementation:

1. **Low pass filtering**

   - **Code:**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
img = cv2.imread('/content/orginal.png', 0)

# Obtain number of rows and columns
# of the image
m, n = img.shape

# Develop Averaging filter(3, 3) mask
mask = np.ones([3, 3], dtype = int)
mask = mask / 9

# Convolve the 3X3 mask over the image
img_new = np.zeros([m, n])

for i in range(1, m-1):
  for j in range(1, n-1):
    temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j + 1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mask[1, 1]+img[i, j + 1]*mask[1, 2]+img[i + 1, j-1]*mask[2, 0]+img[i + 1, j]*mask[2, 1]+img[i + 1, j + 1]*mask[2, 2]

    img_new[i, j]= temp

img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.png', img_new)
```

- **Result:**

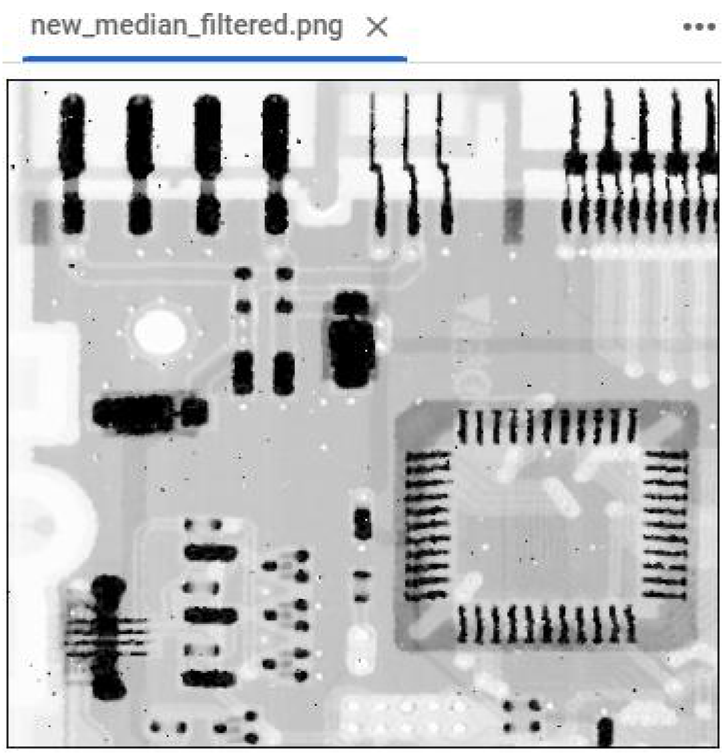


2. Median filtering

- **Code:**

```
# Read the image
img_noisy1 = cv2.imread('orginal.png', 0)

# Obtain the number of rows and columns
# of the image
m, n = img_noisy1.shape

# Traverse the image. For every 3X3 area,
# find the median of the pixels and
# replace the center pixel by the median
img_new1 = np.zeros([m, n])

for i in range(1, m-1):
    for j in range(1, n-1):
        temp = [img_noisy1[i-1, j-1],
               img_noisy1[i-1, j],
               img_noisy1[i-1, j + 1],
               img_noisy1[i, j-1],
               img_noisy1[i, j],
               img_noisy1[i, j + 1],
               img_noisy1[i + 1, j-1],
               img_noisy1[i + 1, j],
               img_noisy1[i + 1, j + 1]]

        temp = sorted(temp)
        img_new1[i, j]= temp[4]

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_median_filtered.png', img_new1)
```

- **Result:**

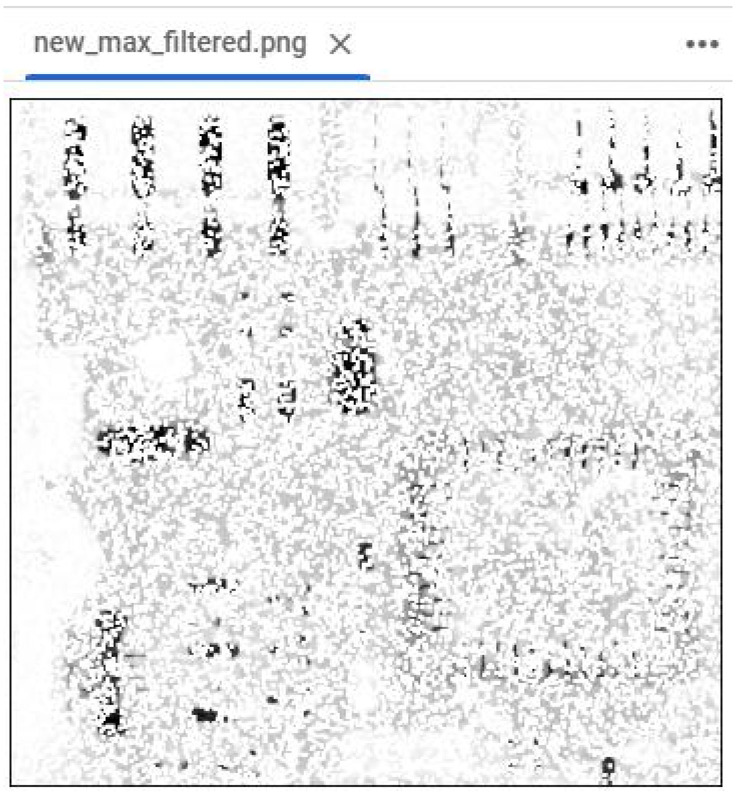


3. **Max filtering**

- **Code:**

```
for i in range(1, m-1):
    for j in range(1, n-1):
        temp = [img_noisy1[i-1, j-1],
               img_noisy1[i-1, j],
               img_noisy1[i-1, j + 1],
               img_noisy1[i, j-1],
               img_noisy1[i, j],
               img_noisy1[i, j + 1],
               img_noisy1[i + 1, j-1],
               img_noisy1[i + 1, j],
               img_noisy1[i + 1, j + 1]]

        # Find maximum value
        img_new1[i, j]= max(temp)

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_max_filtered.png', img_new1)
```

- **Result:**

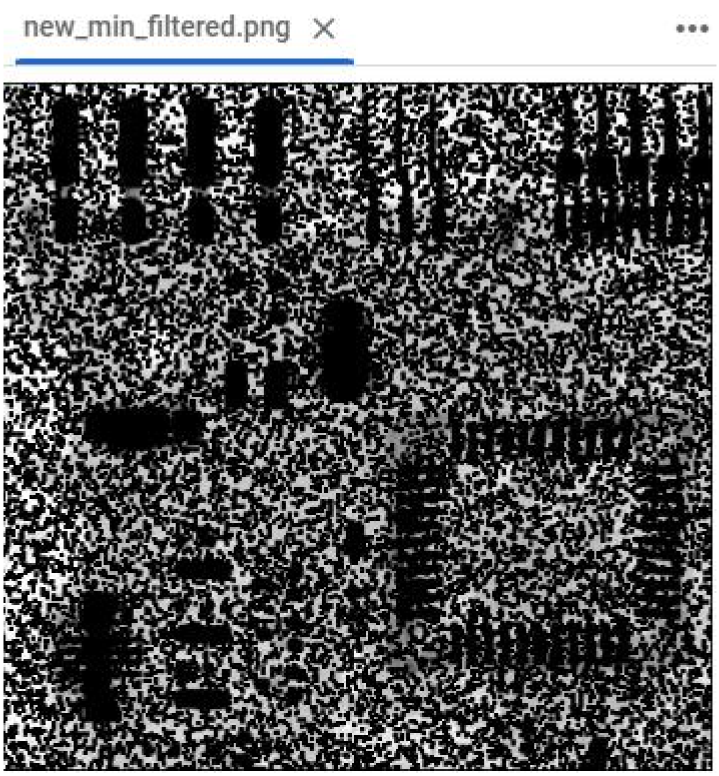


4. **Min filtering**

- **Code:**

```
for i in range(1, m-1):
    for j in range(1, n-1):
        temp = [img_noisy1[i-1, j-1],
               img_noisy1[i-1, j],
               img_noisy1[i-1, j + 1],
               img_noisy1[i, j-1],
               img_noisy1[i, j],
            img_noisy1[i, j + 1],
            img_noisy1[i + 1, j-1],
            img_noisy1[i + 1, j],
            img_noisy1[i + 1, j + 1]]

        # Find minimum value
        img_new1[i, j] = min(temp)

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_min_filtered.png', img_new1)
```

- **Result:**

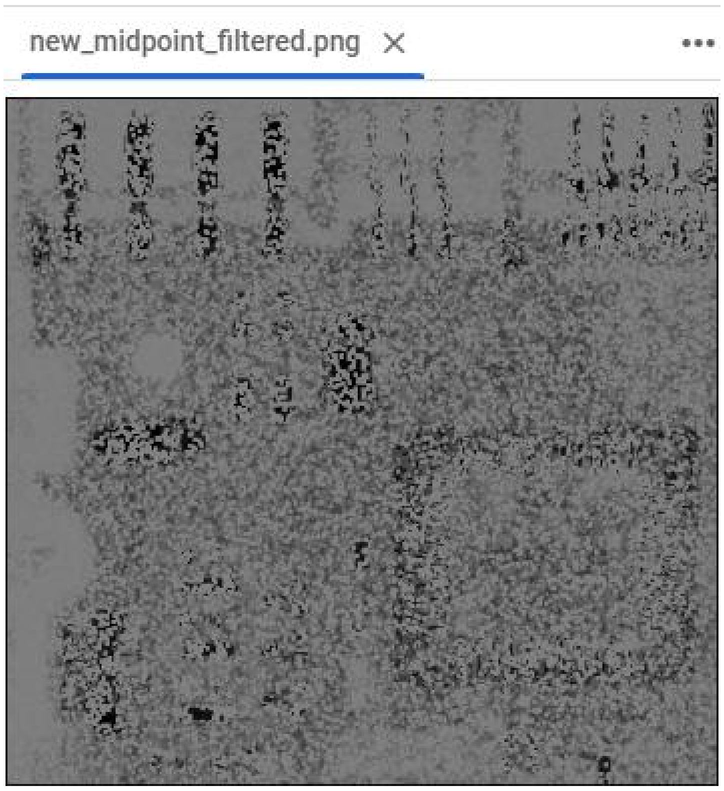


5. **Midpoint filtering**

- **Code:**

```
for i in range(1, m-1):
    for j in range(1, n-1):
        temp = [img_noisy1[i-1, j-1],
            img_noisy1[i-1, j],
            img_noisy1[i-1, j + 1],
            img_noisy1[i, j-1],
            img_noisy1[i, j],
            img_noisy1[i, j + 1],
            img_noisy1[i + 1, j-1],
            img_noisy1[i + 1, j],
            img_noisy1[i + 1, j + 1]]

        # Find minimum and maximum values
        min_val = min(temp)
        max_val = max(temp)
        # Calculate midpoint
        img_new1[i, j] = (min_val + max_val) // 2

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_midpoint_filtered.png', img_new1)
```

- **Result:**



Discussion:

The experimental evaluation provided insights into the performance of each smoothing filter:

Low Pass Filtering: Effectively reduced noise but resulted in blurring of fine details.

Median Filtering: Robust to salt-and-pepper noise, preserving edges well.

Max Filtering: Enhanced bright regions but could over smooth small features.

Min Filtering: Enhanced dark regions but showed similar characteristics to max filtering.

Midpoint Filtering: Provided a balance between noise reduction and detail preservation.

Conclusion:

In conclusion, each smoothing filter offers unique advantages and limitations depending on the specific requirements of the image processing task. Understanding the characteristics of different filters is essential for selecting the most appropriate technique for noise reduction while preserving image features. Further research could focus on hybrid approaches combining multiple filters to achieve optimal results in various scenarios.