DIGITAL IMAGE PROCESSING (BEC613C)

PROGRAMMING ASSIGNMENT REPORT ON

"Implementation of Order Statistic Filters and demonstrating their effect on added noise."

Team Members:

Abhinandan S	**1KG22EC003**
Bhuvana G.H	**1KG22EC015**
Boomika N	**1KG22EC017**
Darshan K	**1KG22EC027**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT BENGALURU-560109

2024-25

Aim:

To write a script/program to implement of Order Statistic Filters and demonstrating their effect on added noise.

Tools used:

MATLAB with Image Processing Toolbox / Python with Open CV Library.

Methodology:

This experiment focuses on applying various order statistic filters-Median, Min, Max, Midpoint, and Alpha-Trimmed Mean-to grayscale images corrupted with salt-and-pepper noise. The aim is to evaluate their effectiveness in noise reduction while preserving image details. The implementation is carried out using MATLAB and its built-in image processing functions.

Step 1: Image Acquisition

- The user is prompted to select an image file (*.jpg, *.png, *.bmp, etc.) using a file selection dialog (uigetfile).
- If the selected image is in RGB (colored) format, it is converted to grayscale using MATLAB's rgb2gray() function to simplify processing.
- Indexed images (like .gif) are converted to RGB using ind2rgb() before conversion to grayscale.
- The image is converted to uint8 format using im2uint8() for consistency in processing.

Step 2: Noise Addition

- Salt-and-pepper noise is artificially added to the grayscale image using MATLAB's imnoise() function.
- The noise density is set to 5% (0.05), simulating impulse noise commonly encountered in digital image transmission.

Step 3: Filtering Techniques

A 3×3 window is used for all filtering operations. The following filters are applied to the noisy image:

1) **Median Filter**:
   - Implemented using medfilt2().
   - Replaces each pixel with the median of its neighborhood.
   - Effectively removes both salt and pepper noise while preserving edges.

2) **Min Filter:**
   - Implemented using ordfilt2() with rank 1 (minimum value).
   - Replaces each pixel with the minimum value in the neighborhood.

- Reduces salt noise (bright pixels) but can darken the image.

3) Max Filter:

- Implemented using ordfilt2() with rank 9 (maximum value in a 3×3 window).
- Replaces each pixel with the maximum value in the neighborhood.
- Effective for reducing pepper noise (dark pixels) but may brighten the image.

4) Midpoint Filter:

- Combines the results of the Min and Max filters.
- Calculates the average of the minimum and maximum values in the window: Midpoint = (Min + Max) / 2.
- Offers a balanced approach, useful when both salt and pepper noise are present.

5) Alpha-Trimmed Mean Filter:

- A robust statistical filter that trims a percentage (α) of the lowest and highest values in the window.
- Implemented using nested loops:
  - Sort the values in the 3×3 window.
  - Trim the lowest and highest 25% values.
  - Compute the mean of the remaining values.
- Provides better performance than the standard mean filter when dealing with outliers.

Step 4: Visualization

- MATLAB's subplot() is used to display the following images side-by-side:
  1. Original (Grayscale) Image
  2. Noisy Image (with salt-and-pepper noise)
  3. Median Filtered Image
  4. Min Filtered Image
  5. Max Filtered Image
  6. Midpoint Filtered Image
- The Alpha-Trimmed Mean Filtered Image is displayed in a separate figure using imshow() due to subplot limits.

Step 5: Saving Filtered Outputs

- All processed images are saved in the same folder as the original image using MATLAB's imwrite() function.
- Filenames are modified to reflect the filter used, for example:
  - image_median_filtered.png
  - image_min_filtered.png
  - image_max_filtered.png
  - image_midpoint_filtered.png
  - image_alpha_trimmed_filtered.png

Code:

```
clc; clear; close all;

% File selection dialog

[filename, pathname] = uigetfile({'.jpg;.png;.bmp;.tif;*.gif', 'Supported Images (*.jpg, *.png, *.bmp, *.tif, *.gif)'}, 'Select an Image');

% Exit if no file selected

if isequal(filename, 0)

    disp('User canceled the operation');

    return;

end
% Read image

img_path = fullfile(pathname, filename);

[img, map] = imread(img_path);

% Convert indexed image (e.g., GIF) to RGB

if ~isempty(map)

    img = ind2rgb(img, map);

end

% Convert to grayscale if RGB

if size(img, 3) == 3

    img = rgb2gray(img);

end

img = im2uint8(img);  % Ensure image is in uint8 format

% Add Salt & Pepper noise

noisy_img = imnoise(img, 'salt & pepper', 0.05);

% Define window size and structuring element

w = 3;
```

```
se = true(w);
% Median Filter
median_filtered = medfilt2(noisy_img, [w w]);
% Min & Max Filters
min_filtered = ordfilt2(noisy_img, 1, se);
max_filtered = ordfilt2(noisy_img, w*w, se);
% Midpoint Filter
midpoint_filtered = uint8((double(min_filtered) + double(max_filtered)) / 2);
% Alpha-Trimmed Mean Filter
alpha = 0.25;
pad = floor(w/2);
padded = padarray(noisy_img, [pad pad], 'symmetric');
alpha_filtered = zeros(size(noisy_img));
for i = 1:size(noisy_img, 1)
    for j = 1:size(noisy_img, 2)
        window = double(padded(i:i+w-1, j:j+w-1));
        sorted_vals = sort(window(:));
        d = floor(alpha * numel(sorted_vals));  % number to trim
        trimmed = sorted_vals((d+1):(end-d));   % trim from both ends
        alpha_filtered(i,j) = mean(trimmed);
    end
end
alpha_filtered = uint8(alpha_filtered);
% Plot all results
figure('Name','Filter Comparison','NumberTitle','off');
subplot(2,3,1), imshow(img), title('Original Image');
```

```
subplot(2,3,2), imshow(noisy_img), title('Noisy Image');
subplot(2,3,3), imshow(median_filtered), title('Median Filtered');
subplot(2,3,4), imshow(min_filtered), title('Min Filtered');
subplot(2,3,5), imshow(max_filtered), title('Max Filtered');
subplot(2,3,6), imshow(midpoint_filtered), title('Midpoint Filtered');
figure, imshow(alpha_filtered), title('Alpha-Trimmed Mean Filter');
% Save output images
[~, name, ~] = fileparts(filename);
imwrite(median_filtered, fullfile(pathname, [name, '_median_filtered.png']));
imwrite(min_filtered, fullfile(pathname, [name, '_min_filtered.png']));
imwrite(max_filtered, fullfile(pathname, [name, '_max_filtered.png']));
imwrite(midpoint_filtered, fullfile(pathname, [name, '_midpoint_filtered.png']));
imwrite(alpha_filtered, fullfile(pathname, [name, '_alpha_trimmed_filtered.png']));
disp('? All filtered images saved successfully in the original folder.');
```

Results and Discussion:

Fig (a). Input image

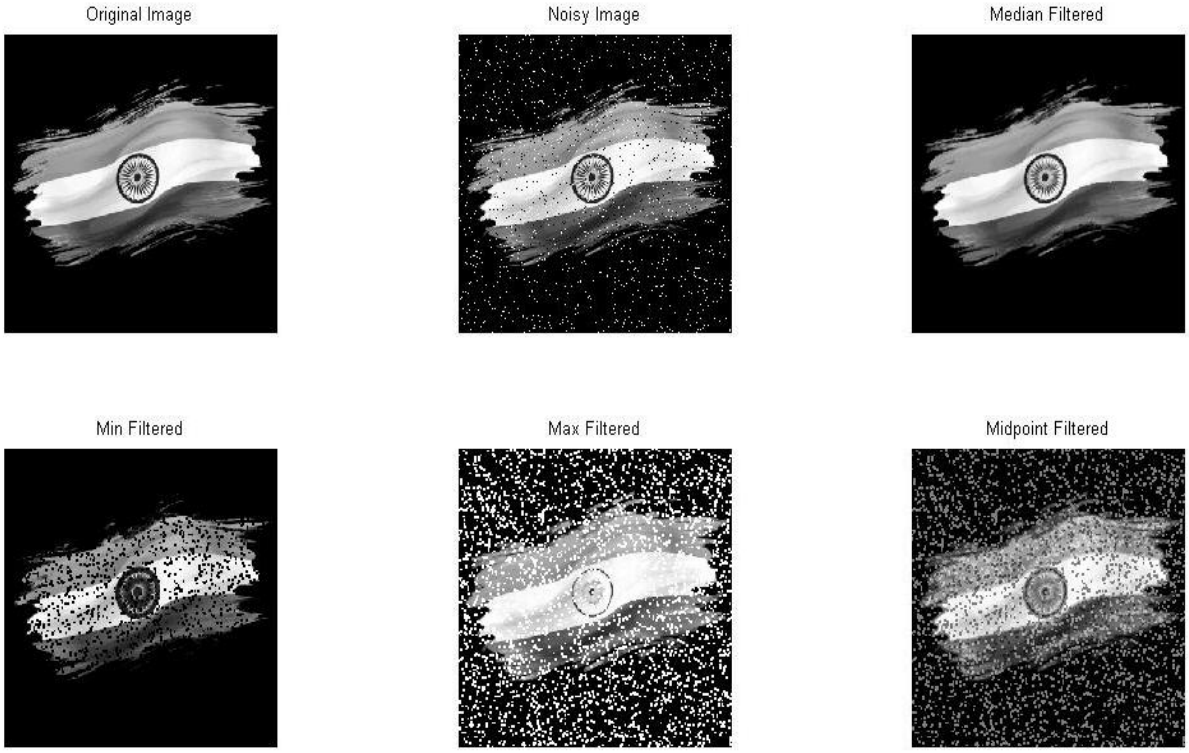

Fig (b). Order Static Filters

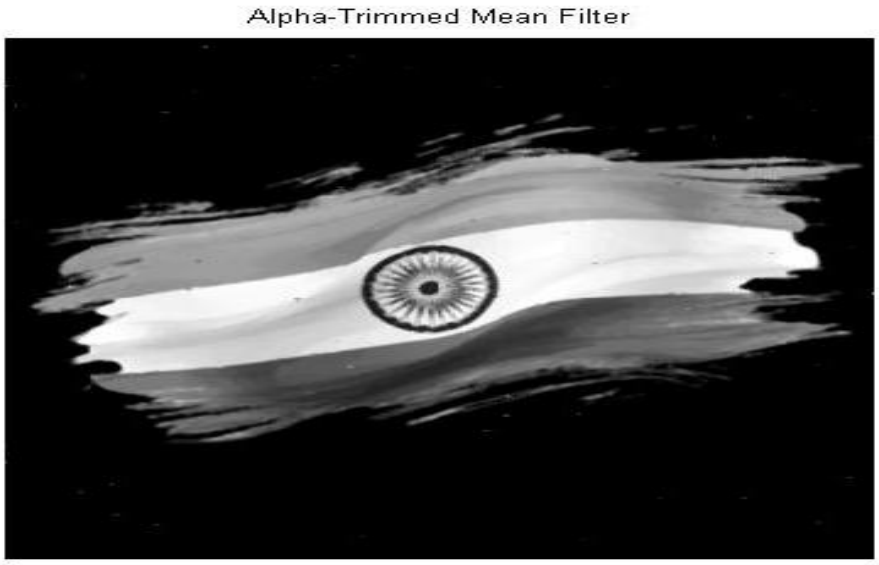

Fig (c). Alpha-Trimmed Mean Filter

Conclusion:

The experiment demonstrates that while Min and Max filters are simple and fast, they tend to either darken or brighten the image excessively and are less effective in preserving image details. The Median filter, though slightly more computationally intensive, offers significantly better performance in suppressing salt-and-pepper noise while preserving edges and fine details, making it ideal for most practical denoising applications.

The Midpoint filter, which averages the Min and Max responses, provides a balanced yet less accurate result in high-noise regions. In contrast, the Alpha-Trimmed Mean filter, though computationally heavier due to sorting and trimming operations, demonstrates robust performance by effectively rejecting outliers and maintaining image quality. Hence, the Median and Alpha-Trimmed Mean filters are generally more reliable for practical use, especially in cases where both noise suppression and detail preservation are critical.