LEEDS BECKETT UNIVERSITY

School of Computing, Creative Technology and Engineering

Student Name	Pooja Pantha
Student ID	77356785
Module Name	Embedded Intelligent Vision System
Level	L6
Assignment name	Coursework in Image Video Processing

Table of Contents

1. 1. Introduction 5
2. 2. Image and Video processing of the Coursework Files 5

2.1 Discuss and evaluate the step-by-step analysis and processing of Image1.jpg 5
2.2 Discuss and evaluate the step-by-step analysis and processing of Image2.jpg 7
2.3 Discuss and evaluate the step-by-step analysis and processing of Image3.bmp 9
2.4 Discuss and evaluate the step-by-step analysis and processing of Image4.jpg 12
2.5 Discuss and evaluate the step-by-step analysis and processing of Image5.jpg 15
2.6 Discuss and evaluate the step-by-step analysis and processing of Image6.jpg 18
2.7 Discuss and evaluate the step-by-step analysis and processing of Image7.jpg 23
2.8 Discuss and evaluate the step-by-step analysis and processing of Image8.jpg 28
2.9 Discuss and evaluate the step-by-step analysis and processing of image sequence 31
2.10 Discuss and evaluate the step-by-step analysis and processing of video1.mp4 36
2.11 Discuss and evaluate the step-by-step analysis and processing of video2.avi 38
2.12 Discuss and evaluate the step-by-step analysis and processing of video3.avi 42

3. 3. References 46

Table of Figures

Figure 1:Original Imag1.jpg 5
Figure 2: Matlab code for Image1 analysis and adjustment 6
Figure 3: Orginal image and Balanced image of Image1.jpg 7
Figure 4: Original Image2 7
Figure 5: Code snippet for image2 enhancement 8
Figure 6: Original Image and Sharpened Image of Image2 9
Figure 7: Original Image3 9
Figure 8: Code snippet for Image3 Enhancement 10
Figure 9: Comparison of different analysis in image3 11
Figure 10: Number plate detection of image3 11
Figure 11: Original Image4 12
Figure 12: Code snippet for image4 implementation and analysis 13
Figure 13: Comparison of original, standard qualized and CLAHE enhancement of Image4 14
Figure 14: Histogram comparison for Image4 14
Figure 15: Original Image5 15
Figure 16: Code snippet for Image5 enhancement 16
Figure 17: Motion Blur detection in Image5 17
Figure 18: Side by side comparison of the outcome of Image5 18
Figure 19: Original Image6 18
Figure 20: CLAHE analysis for Image6 20
Figure 21: Histogram Equalization for Image6 20
Figure 22: Gamma Correction for Image6 21
Figure 23: Super-Resolved Enhanced Image of Image6 21
Figure 24: Face Detection in Image6 22
Figure 25: Subplot of all the analysis done for Image6 23
Figure 26: Original Image7 23
Figure 27: Approach 1 code snippet for image7 24
Figure 28: Grayscale Image of image7 25
Figure 29: Applied median filter in image7 25
Figure 30: Histogram Equalized in image7 26
Figure 31: Histogram comparison for all 3 analysis in image7 26
Figure 32: Sandy Regions Detection of image7 27
Figure 33: Approach 2 code snippet for image7 27
Figure 34: Comparison between the analysis done for image7 enhancement 28
Figure 35: Original Image8 28
Figure 36: Code snippet for image8 enhancement 29
Figure 37: Magnitude Spectrum to click in a bright spot to remove bright spot in image8 30
Figure 38: Output after the enhancement in image8 30
Figure 39: vespa001.jpg 31
Figure 40: Code snippet Part 1 for Image Sequence 32
Figure 41: Code snippet Part 2 for Image Sequence 33
Figure 42: Code snippet Part 3 for Image Sequence 34
Figure 43: Marked area for the number plate detection 35
Figure 44: First frame of the video 35
Figure 45: Comparison after enhancement 36
Figure 46: First frame of Video-1 36
Figure 47: Code snippet part 1 for Video-1 37
Figure 48: Code snippet part 2 for video-1 37
Figure 49: Final Edge Detection of the video-1 38

Figure 50: First frame of video2 39
Figure 51: Code snippet part 1 for video2 39
Figure 52: Code snippet part 2 for video2 40
Figure 53: Code snippet part 3 for video2 40
Figure 54: Selected Number Plate Area of video2 41
Figure 55: Final output of the video2 42
Figure 56: First frame of video3 43
Figure 57: Code snippet part 1 for video3 43
Figure 58: Code snippet part 2 for video3 44
Figure 59: Frame count of video3 44
Figure 60: Comparison of video3 after enhancement 45

1. Introduction

This coursework involves the process and studies a set of photos and videos by applying various techniques for improvement and repair. Different files suffered from issues such as uneven colouring, lack of focus, low differences between bright and dark sections or lots of noise. Distinct ways are used to enhance the image in every file so it could be read and interpreted better (Shen and Ma, 2010).

Before processing, the coursework will highlight and check for any issues in every file, proceeding with a description of how each technique is put into action. Methods used on images involved adjusting colours, raising image contrast, sharpening images, filtering noise, identifying edges and adjusting histograms (Salvatore, 2025). The frame extraction, noise filtering, adjusted the contrast and corrected motion blur is used for all video files.

The completion of all the tasks is achieved using MATLAB R2024b. The software included various image and video analysis tools such as chromadapt, imsharpen, imgaussfilt, adapthisteq, fft2, deconvlucy and the Image Processing and Computer Vision toolboxes for video processing (TechnoLynx Ltd, 2025). Usual approaches and procedures in image and video processing are applied.

2. Image and Video processing of the Coursework Files

2.1 Discuss and evaluate the step-by-step analysis and processing of Image1.jpg

Figure 1:Original Imag1.jpg

Strong green colouring in Image1.jpg undermined the natural quality of the photo. Such balance disturbance prevents us from reading images accurately and reduces the usefulness of image analysis. For that reason, I employed a technique to adjust the colours and restore the image to better tones.

To solve the problem, the included chromadapt function from MATLAB was put to use. The purpose of this function is to correct colour differences in digital images due to various lighting situations. The 'grayworld' approach was decided upon in this analysis. It props on the idea that a typical natural scene's average colour is gray. Because of this, each of the RGB channels is modified so that their colours are balanced, making any colour dominance disappear (Pai, 2024).

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\image1.m *

image1.m *

```
1   % Read the image
2   img = imread('Image1.jpg');
3
4   % Convert image to double for processing
5   img_double = im2double(img);
6
7   % Compute average of each RGB channel
8   mean_R = mean(mean(img_double(:,:,1)));
9   mean_G = mean(mean(img_double(:,:,2)));
10  mean_B = mean(mean(img_double(:,:,3)));
11
12  % Gray world illuminant
13  gray_illuminant = [mean_R mean_G mean_B];
14
15  % Apply chromatic adaptation
16  img_balanced = chromadapt(img_double, gray_illuminant);
17
18  % Display the result
19  figure;
20  subplot(1,2,1); imshow(img); title('Original Image');
21  subplot(1,2,2); imshow(img_balanced); title('Color Balanced Image using chromadapt');
22
23  % Save the result
24  imwrite(img_balanced, 'Image1_balanced.jpg');
25
```

Figure 2: Matlab code for Image1 analysis and adjustment

Step-by-step Analysis of Image1:

I. Line 2 reads the original image from the file and keeps it in the ‘img’ variable.

II. Since chromadapt accepts input in the range [0,1], in line 5 the image is converted from uint8 to double precision.

III. Lines 8, 9, 10 calculates the average brightness for red, green and blue channels. The averages define the current lighting in the photograph.

IV. In line 13, average values of the channels are put together to create a vector. The vector describes the type of lighting used to take the photo.

V. In line 16, Chromatic adaptation is performed on the image based on the assumption of a gray world. The algorithm sets the level of each channel to ensure the average is equal, correcting the issue of colour bias.

VI. Lines 19, 20, 21 presents the image before and after colour correction next to each other for the comparison.

VII. Line 24 saves the image to another file to be used or reported later.

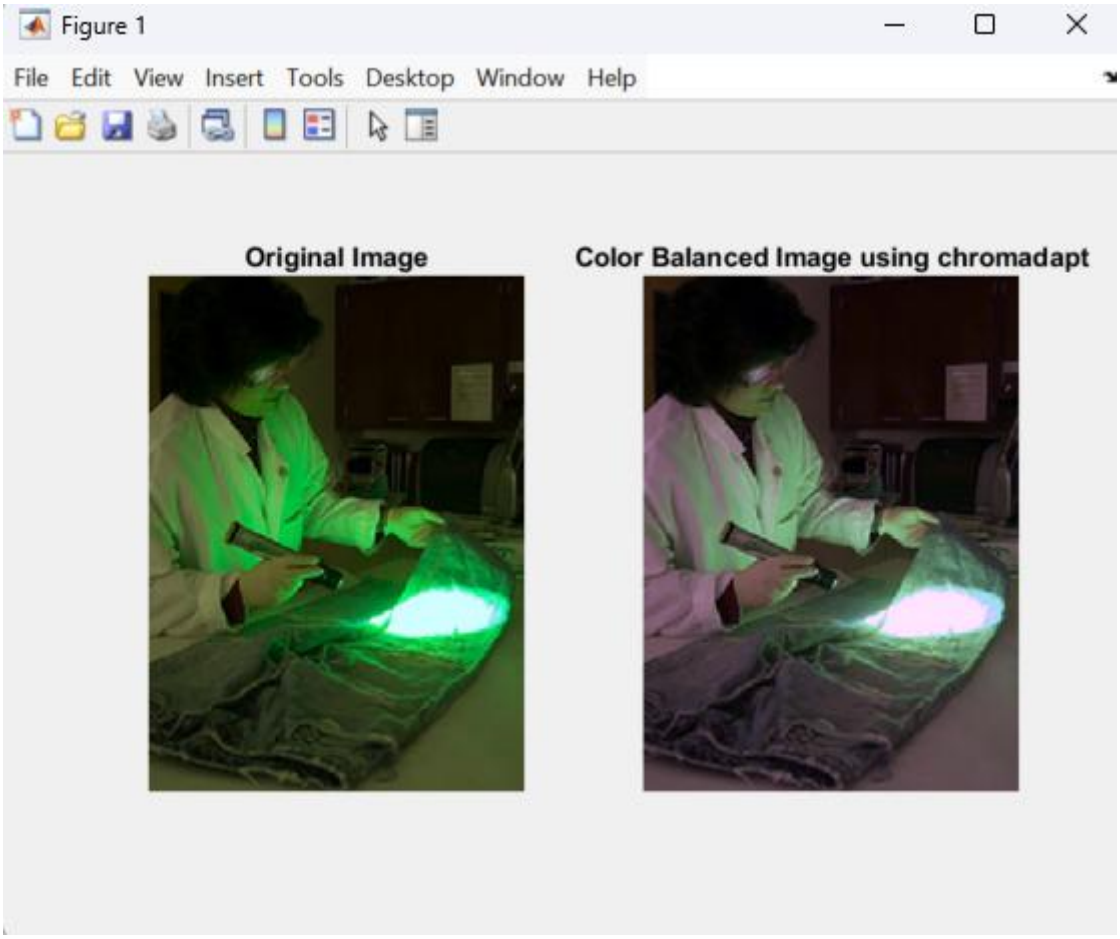

Figure 3: Orginal image and Balanced image of Image1.jpg

The result was put beside the original image using two plots side by side. The filtered image looked much more neutral and true to nature. Accurately displaying colour is very important in field such as forensic imaging and quality control.

2.2 Discuss and evaluate the step-by-step analysis and processing of Image2.jpg

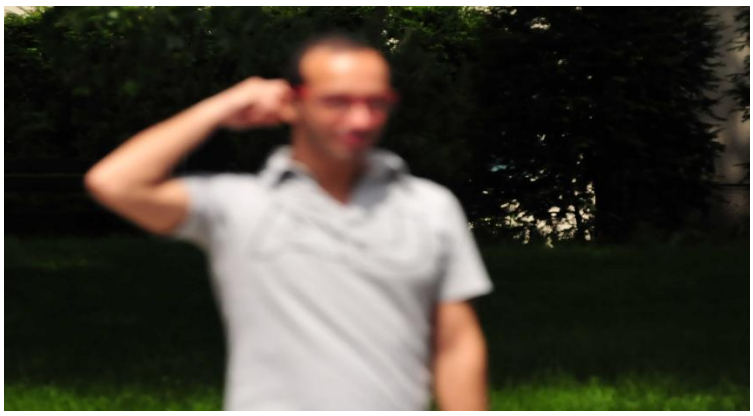

Figure 4: Original Image2

In this photo, the blur did not streak in any direction, indicating that it was most likely due to optical blur. The picture showed the subject looking soft and parts of the background was clearer and slightly sharper.

To make clearer picture, sharpened the image first and then smoothed it slightly. Using the sharpening tool, aggressive sharpening. To avoid harsh effects and halos that appear from too much sharpening, Gaussian blur was used for the next step (Cambridge in Colour, n.d.). With this technique, the picture appeared sharper without any unwanted objects.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\imag2.m

imag2.m

```
1   % Load and normalize the image
2   I = im2double(imread('Imag2.JPG'));
3
4   % Define sharpening config (as tested)
5   cfg = struct('Radius', 14, 'Amount', 11, 'Threshold', 0);
6
7   % Apply aggressive sharpening
8   sharpened = imsharpen(I, ...
9       'Radius', cfg.Radius, ...
10      'Amount', cfg.Amount, ...
11      'Threshold', cfg.Threshold);
12
13  % Apply smoothing (Gaussian blur) to reduce harsh edges
14  smoothed = imgaussfilt(sharpened, 1);  % sigma = 1
15
16  % Visualize all three images side-by-side
17  figure;
18
19  subplot(1, 3, 1);
20  imshow(I);
21  title('Original Image');
22
23  subplot(1, 3, 2);
24  imshow(sharpened);
25  title(sprintf('Sharpened (R=%d, A=%d)', cfg.Radius, cfg.Amount));
26
27  subplot(1, 3, 3);
28  imshow(smoothed);
29  title('Sharpened + Smoothed');
30
```

Figure 5: Code snippet for image2 enhancement

Step-by-step analysis for image2:

I. Line 2 converts the pixel values in the image to fall between [0, 1] to ensure the image is processed correctly.

II. Used the Radius, Amount and Threshold settings to enhance the edges that require a more pronounced effect.

III. Image is blurry at the edges, so used ‘imsharpen’ to sharpen and enhance the image.

IV. Run imgaussfilt to reduce the over-sharpening at the edges and look more natural.

V. Placed the original, sharpened and smoothed versions of the image next to each other.

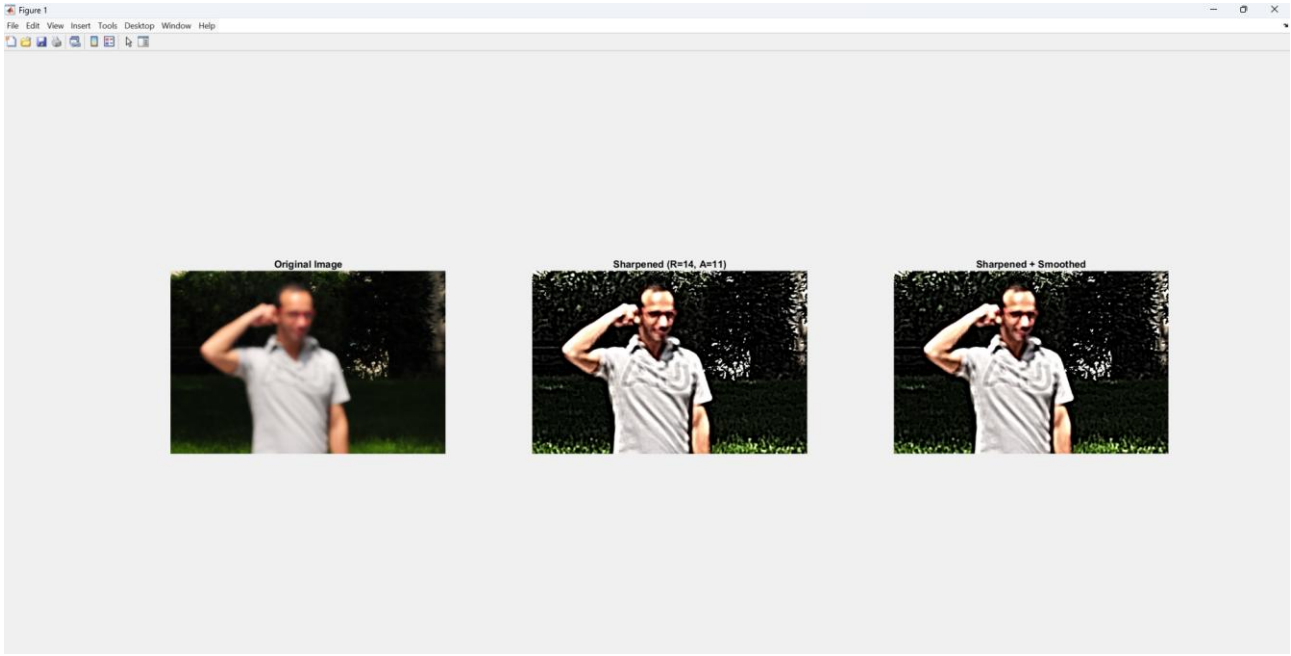

Figure 6: Original Image and Sharpened Image of Image2

This method helped to reduce the blurriness in the original image. Some features were lost during capture but the image sharpness and look were remarkably improved. After sharpening the image, applying Gaussian smoothing made the picture looks cleaner and more realistic. This workflow is suitable when it's not clear how the blur function works but the picture must be fixed quickly.

2.3 Discuss and evaluate the step-by-step analysis and processing of Image3.bmp

Figure 7: Original Image3

Because of low contrast and possible noise in the image, the number plate of the car seen in Image3.bmp is difficult to see. Because the image does not have colour, we need to enhance the number plate by cleaning it, adjusting the contrast and using edge recognition in MATLAB. The image does not provide data about the car's colour because it is grayscale.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\image3.m

image3.m

```
% 1. Load BMP grayscale image into workspace
img = imread('Image3.bmp');

% 2. Denoising
if exist('imnlmfilt', 'file')
    denoised = imnlmfilt(img);
else
    denoised = wiener2(img, [5 5]);
end

% 3. CLAHE - Adaptive histogram equalization to enhance contrast
enhanced = adapthisteq(denoised);

% 4. Edge Detection (Canny method)
edges = edge(enhanced, 'Canny');

% 5. Display all stages for comparison
figure;
subplot(2,2,1);
imshow(img);
title('Original Grayscale Image');

subplot(2,2,2);
imshow(denoised);
title('Denoised Image');

subplot(2,2,3);
imshow(enhanced);
title('Contrast Enhanced Image');

subplot(2,2,4);
imshow(edges);
title('Edge Detection (Canny)');

% 6. Optional: Overlay rectangle manually on number plate region
figure;
imshow(enhanced);
hold on;
rectangle('Position',[50 50 200 50],'EdgeColor','r','LineWidth',2);
title('Number Plate Area');
hold off;
```

Figure 8: Code snippet for Image3 Enhancement

Step-by-step analysis:

I. In line 2 the image is added to the workspace of MATLAB. No conversion to a different colour format is necessary as the image is already in grayscale.

II. In line 5-9, random noise is removed by denoising without affecting the sharp parts such as the edges. Because edge details are best kept using ‘imnlmfilt’, it is selected.

III. CLAHE is often used to increase the contrast level in darker parts of the image. So used CLAHE in line 12 too. Details such as letters on car number plates stand out more.

IV. In line 15, sharp transitions in brightness detected by canny edge ways allow you to highlight objects such as number plates and text.

V. In Line 18-33, two rows and two columns are created to hold the picture, denoised picture, picture with enhanced contrast and the edge result. With this approach, you can easily compare each stage of processing.

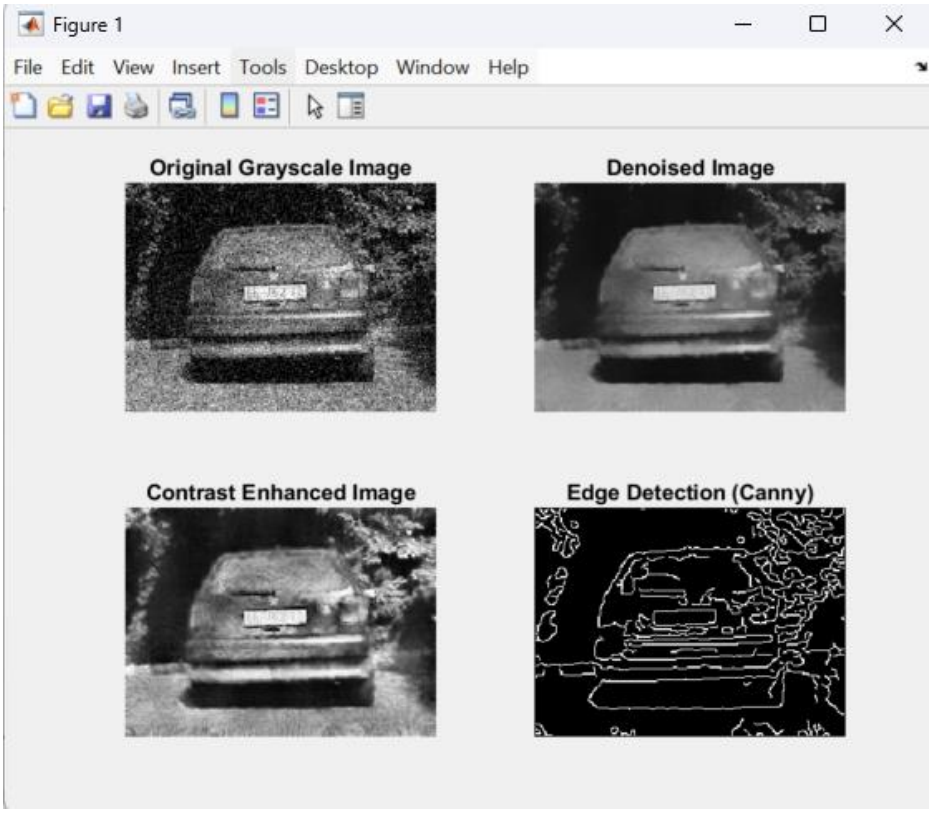

Figure 9: Comparison of different analysis in image3

VI. Line 36-41 then draws a rectangle on the screen over where the number plate should be.

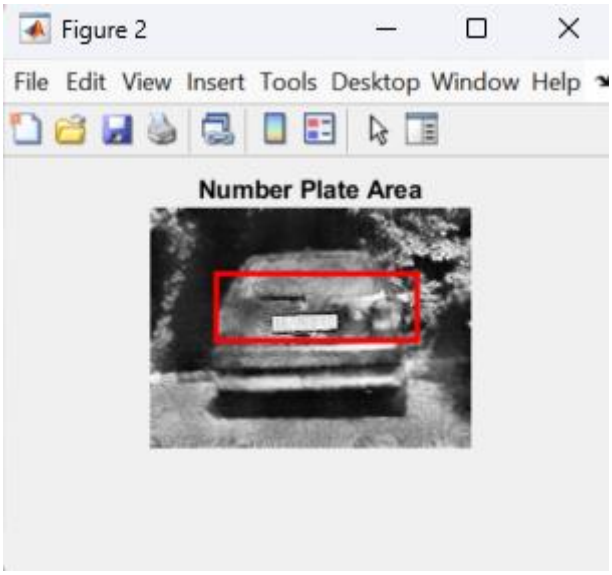

Figure 10: Number plate detection of image3

The given analysis involved modifying a grayscale picture to make the car's number plate easier to spot. To start, both the Non-Local Means Filter and Wiener Filter were used to reduce noise without affecting the borders of the image (Kim, Choi and Lee, 2020). After that, CLAHE was applied to increase the contrast which made the number plate more visible. Next, canny edge detection was used to bring out the edges and lines within the image.

2.4 Discuss and evaluate the step-by-step analysis and processing of Image4.jpg

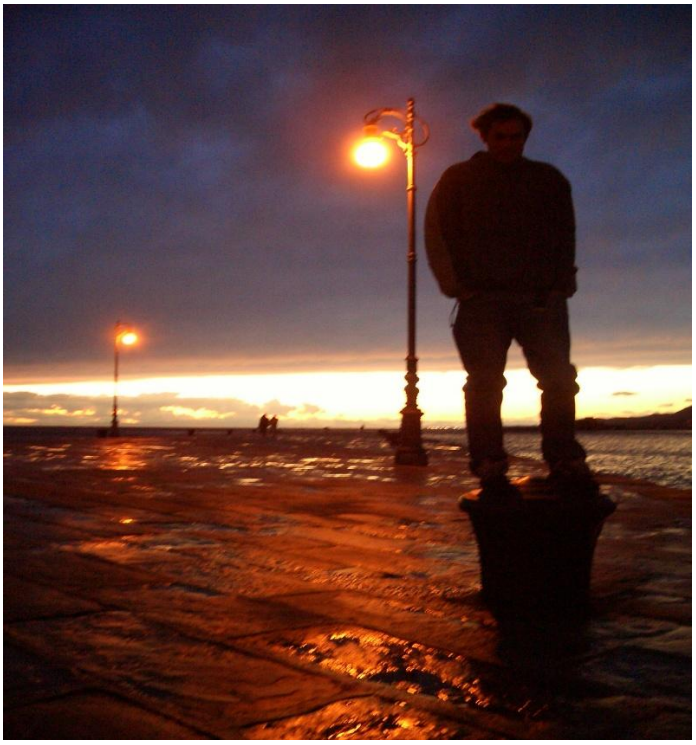

Figure 11: Original Image4

MATLAB code below was used to make the dark spots more noticeable in Image4.jpg, particularly for the figure which is standing by a bright background. This was done by equalizing both the Standard Histogram and the Adaptive Histogram (CLAHE) on each of the RGB channels (Rosebrock, 2021). The comparison of both images was done by putting them next to each other, along with their respective histograms before and after enhancing them.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\image4.m

image4.m

```
% Read the original image
img = imread('Image4.jpg');

% Split RGB channels
R = img(:,:,1);
G = img(:,:,2);
B = img(:,:,3);

% Apply Adaptive Histogram Equalization (CLAHE)
R_clahe = adapthisteq(R, 'ClipLimit', 0.02, 'NumTiles', [8 8]);
G_clahe = adapthisteq(G, 'ClipLimit', 0.02, 'NumTiles', [8 8]);
B_clahe = adapthisteq(B, 'ClipLimit', 0.02, 'NumTiles', [8 8]);
img_clahe = cat(3, R_clahe, G_clahe, B_clahe);

% Apply Standard Histogram Equalization (histeq)
R_histeq = histeq(R);
G_histeq = histeq(G);
B_histeq = histeq(B);
img_histeq = cat(3, R_histeq, G_histeq, B_histeq);

% Display images side by side
figure,
subplot(1,3,1), imshow(img), title('Original Image');
subplot(1,3,2), imshow(img_histeq), title('Standard Hist. Equalized');
subplot(1,3,3), imshow(img_clahe), title('CLAHE Enhanced');

% Display histograms for original and standard equalized channels
figure,
subplot(2,3,1), imhist(R), title('Original R');
subplot(2,3,2), imhist(G), title('Original G');
subplot(2,3,3), imhist(B), title('Original B');
subplot(2,3,4), imhist(R_histeq), title('Equalized R');
subplot(2,3,5), imhist(G_histeq), title('Equalized G');
subplot(2,3,6), imhist(B_histeq), title('Equalized B');

```

Figure 12: Code snippet for image4 implementation and analysis

Step-by-step analysis:

I. In line 2, brought Image4.jpg file into the MATLAB workspace.

II. In lines 5, 6, 7, partitioned each of the RGB channels of an image and took steps to process them individually.

III. In line 10-13, for every channel, used ‘adapthisteq’ at clip limit 0.02 and tile size 8×8 to enhance the local contrast. The processed channels were merged again to get a single image.

IV. In line 16-20, strengthened and decreased all colours of each channel uniformly with ‘histeq’. Combined the individual channels after they had been made equal.

V. In line 22-25, placed the original, standard equalized and CLAHE version of the image in one figure for easy comparison.

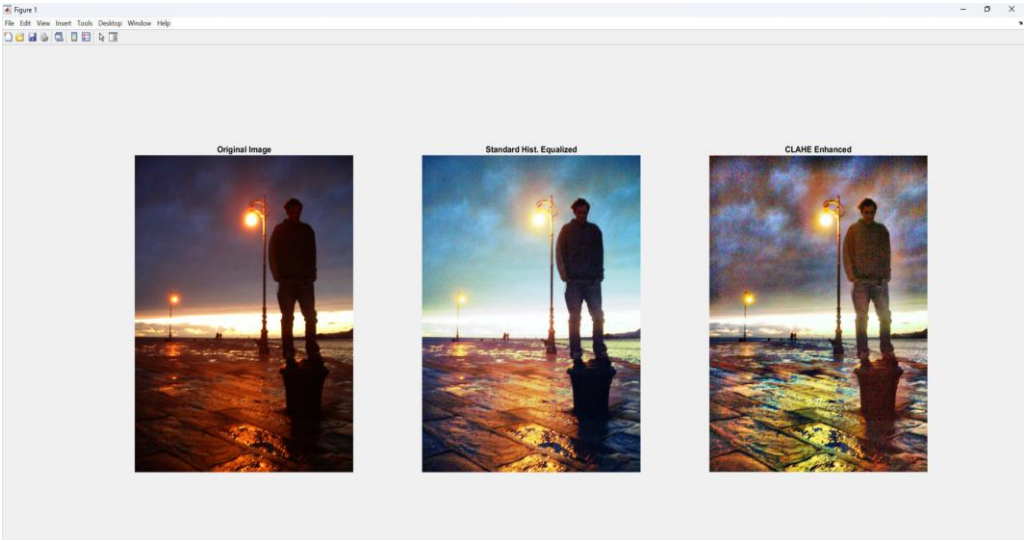

Figure 13: Comparison of original, standard qualized and CLAHE enhancement of Image4

VI. In line 28-34, plotted diagrams for each Red, Green and Blue channel before and after equalization to notice any changes in their intensity.

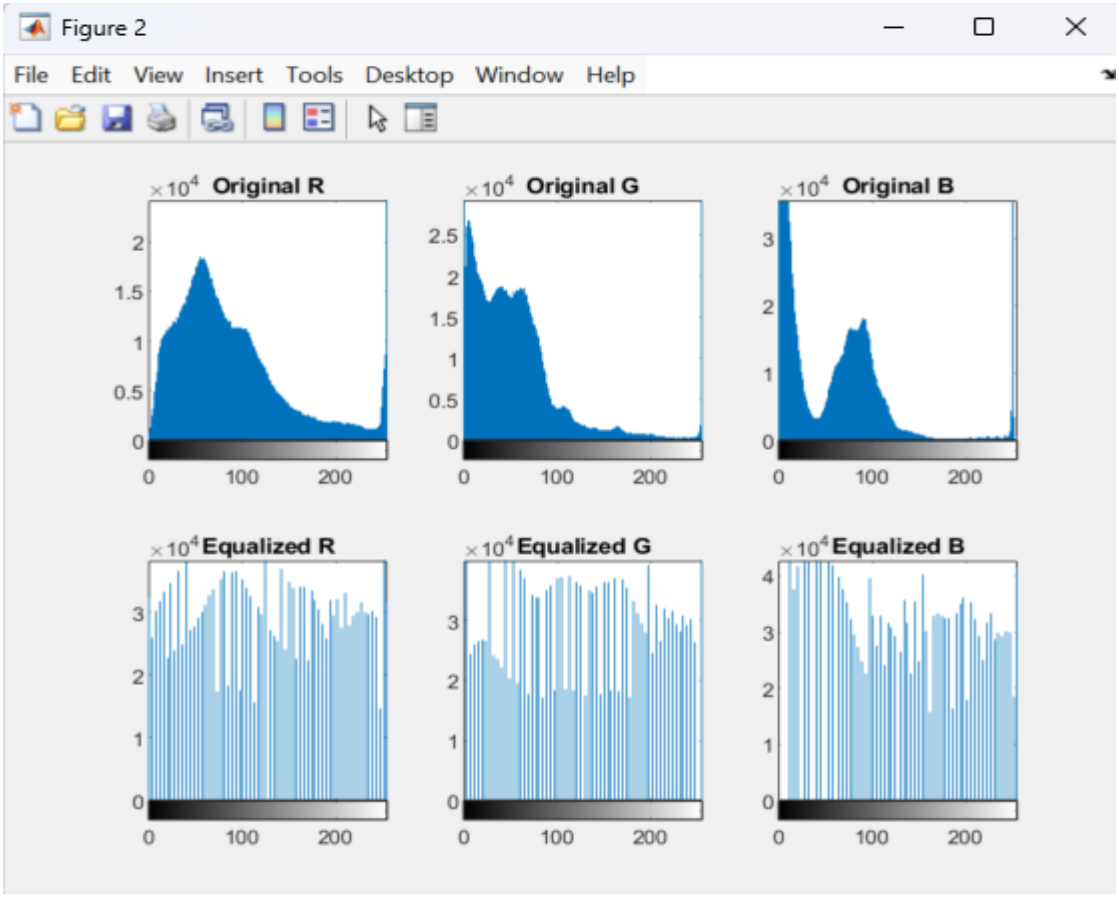

Figure 14: Histogram comparison for Image4

Here, two ways of boosting contrast were applied to Image4.jpg, making the image's darker parts visible yet still keeping the brightness of the background. The Histogram Equalization tool changed the overall brightness, while the CLAHE tool made adjustments to contrast in different areas. Individual processing of RGB made it possible to keep colour balance in place. This demonstrated that both techniques increase the visibility of photos in the image.

2.5 Discuss and evaluate the step-by-step analysis and processing of Image5.jpg

Figure 15: Original Image5

On first sight, the subject in Image5.jpg looked very blurry. Looking at the blurred picture closely, it was observed that the patterns leaned either straight across or at a slight angle from the top or bottom. Unlike optical blur, these lines generally show motion blur which is more directional (Abdullah-Al-Mamun, Tyagi and Zhao, 2021).

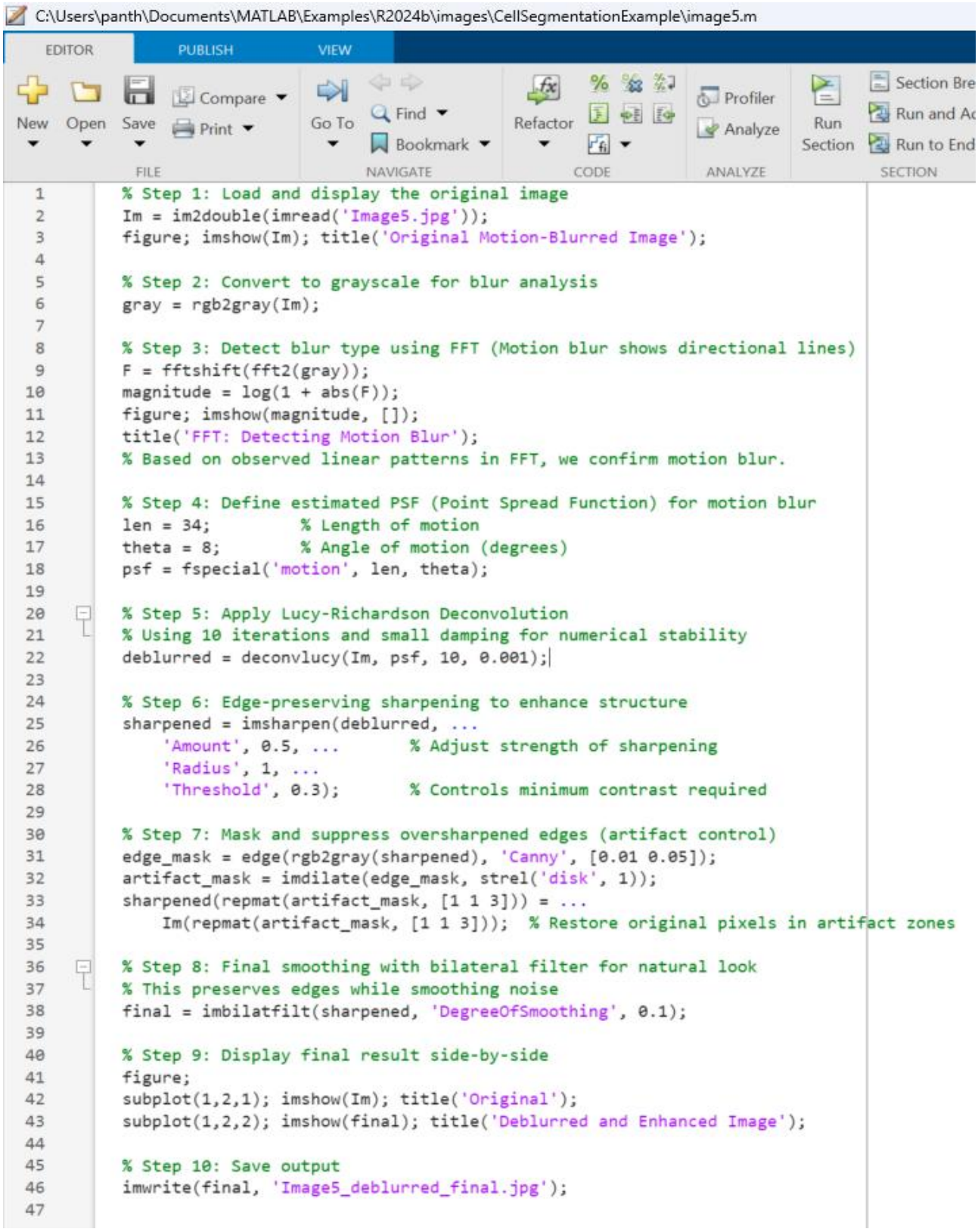

Figure 16: Code snippet for Image5 enhancement

Step-by-step analysis:

I. In line 2 and 3, after loading the image and upgrading its precision, it can be shown to check blur and other issues.

II. When analysing images with frequency-domain, grayscale conversion made the work much easier in line 6.

III. In line 9-12, FFT technology is applied to examine the frequency components. By observing linear patterns in the FFT magnitude spectrum, I could tell that motion blur was present, since motion also results in consistent patterns in frequencies.

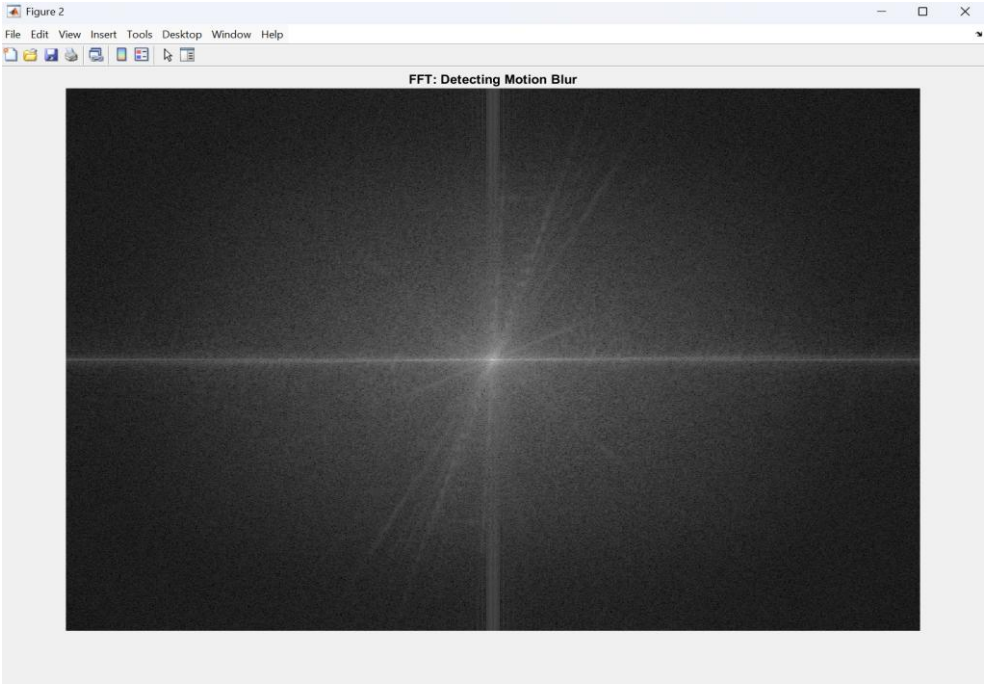

Figure 17: Motion Blur detection in Image5

IV. In line 16-18, a motion blur kernel (PSF) is set up by approximating what is seen. Adjusted both length and angle to better resemble the apparent motion blur in the shot.

V. The PSF was used with Lucy-Richardson deconvolution in line 22 and after 10 iterations, there was a decent balance maintained between the degree of sharpness and how much noise was increased.

VI. Line 25-28, 'imsharpen' improves the sharpness of lines that represent the man's face and body. It was important to avoid sharpening too much because this might have caused artifacts.

VII. In line 31-34, a Canny edge detector locates spots where the image is too blurry or full of noise. Through dilation and masking, areas with these artifacts are fixed by restoring the image's original pixels.

VIII. In line 38, the bilateral filter removes the rest of the noise to create a more natural scene. It retains the lines and angles of an image but clears away little noise found in the image.

IX. The outcomes are illustrated in line 41-43, next to each other so they can be compared.

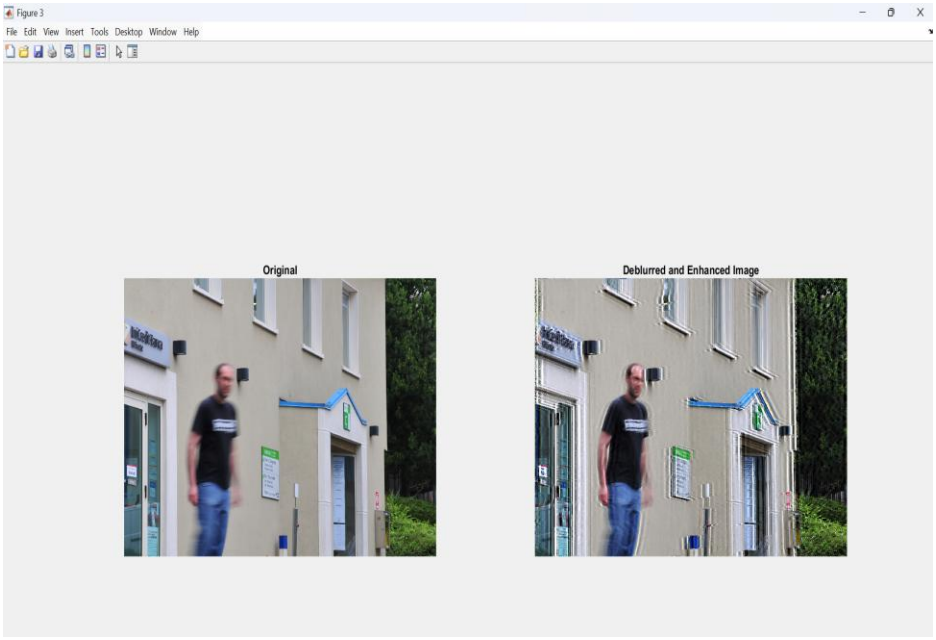

Figure 18: Side by side comparison of the outcome of Image5

First, Lucy-Richardson deconvolution was carried out using a motion PSF; after that, the images were sharpened using edge-aware methods, artifacts were suppressed, and bilateral filtering was applied. As a result, the picture became much clearer, helping to highlight the subject (the man) and keeping the noise and distortions to a minimum.

2.6 Discuss and evaluate the step-by-step analysis and processing of Image6.jpg

Figure 19: Original Image6

Here, the Image6.jpg image was improved in MATLAB using different techniques so it could be read more easily under difficult light. The original photo had low contrast, visible glare at the top and the face was poorly lit, so it was decided that advanced enhancement should be applied. The following describes how each step was applied.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\image6.m

image6.m

```
% Read the image
img = imread('Image6.jpg');
figure, imshow(img), title('Original Image');

% Convert to grayscale
grayImg = rgb2gray(img);

% Apply CLAHE (adaptive histogram equalization)
claheImg = adapthisteq(grayImg, 'NumTiles', [8 8], 'ClipLimit', 0.01);
figure, imshow(claheImg), title('After CLAHE');

% Histogram Equalization
histeqImg = histeq(claheImg);
figure, imshow(histeqImg), title('After Histogram Equalization');

% Gamma Correction to brighten mid-tones
gamma = 0.4; % < 1 brightens image
gammaCorrected = imadjust(histeqImg, [], [], gamma);
figure, imshow(gammaCorrected), title('After Gamma Correction');

% Super-Resolution using Bicubic Interpolation
superRes = imresize(gammaCorrected, 2, 'bicubic');
figure, imshow(superRes), title('Super-Resolved Enhanced Image');


% Detect Face using Viola-Jones
faceDetector = vision.CascadeObjectDetector();

% Detect face in the super-resolved enhanced image
bbox = step(faceDetector, superRes); % 'superRes' is the final enhanced image

% Draw bounding box if face is detected
if ~isempty(bbox)
    detectedImg = insertObjectAnnotation(superRes, 'rectangle', bbox, 'Face');
    figure, imshow(detectedImg), title('Detected Face');
else
    disp('No face detected.');
end

figure('Name','Image Enhancement and Face Detection','NumberTitle','off');
subplot(2,3,1), imshow(img), title('Original Image');
subplot(2,3,2), imshow(grayImg), title('Grayscale');
subplot(2,3,3), imshow(claheImg), title('CLAHE');
subplot(2,3,4), imshow(histeqImg), title('Histogram Equalized');
subplot(2,3,5), imshow(superRes), title('Super-Resolution');
subplot(2,3,6), imshow(detectedImg), title('Face Detection');

```

Step-by-step analysis:

I. In line 2, the image was displayed using ‘imread’ to detect any possible issues. I noticed that there was poor contrast due to sensor saturation and that the light at the top was very bright and washed out the rest of the image.

II. Because colours did not contribute to better image enhancement, rgb2gray was used to turn the image to grayscale. By doing this, the rest of the processing became easier and less complex in line 6.

III. In line 9 and 10, the process to apply CLAHE was done in MATLAB using the 'adapthisteq' function. Local histogram equalization improves the contrast in those places where the usual method does not work well. More details about the background came out and the intensity balance also improved as a result of the CLAHE operation.

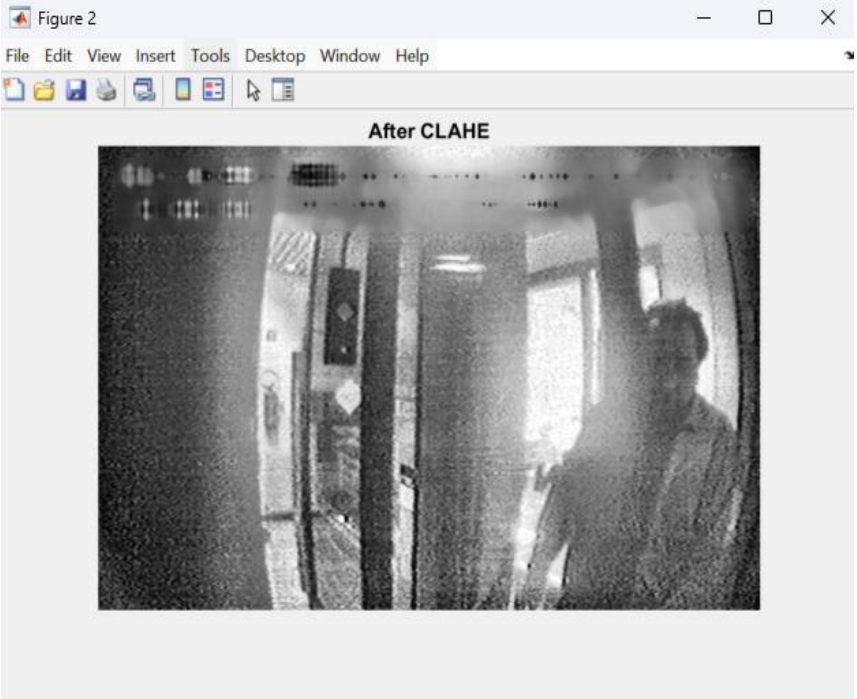

Figure 20: CLAHE analysis for Image6

IV. In line 13 and 14, for better global contrast, the image was processed through 'histeq'. The enhanced contrast made the regions of interest and the background in the image distinct throughout the picture.

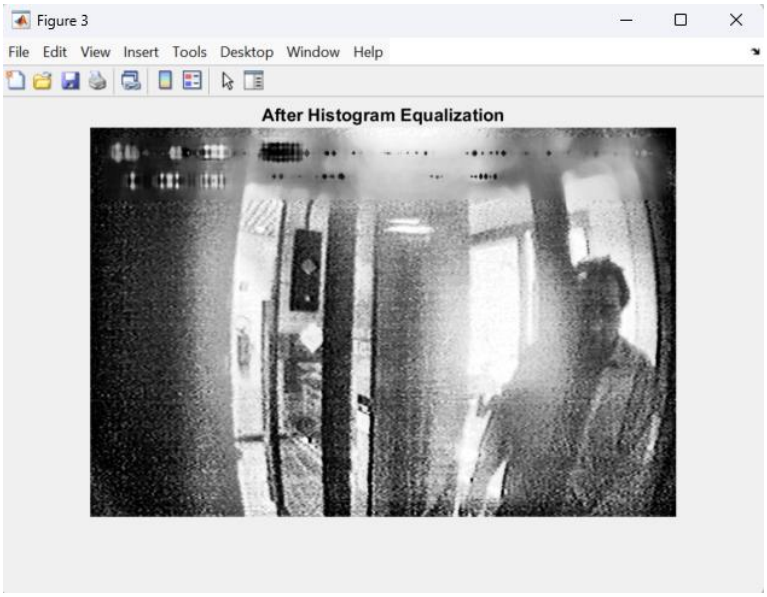

Figure 21: Histogram Equalization for Image6

V. In line 17-19, using gamma correction of 0.4 and 'imadjust', the mid-tone regions of the image were improved and made brighter. By carrying out this step, more of the face became visible in the dim areas without shining the brightest spots too much.

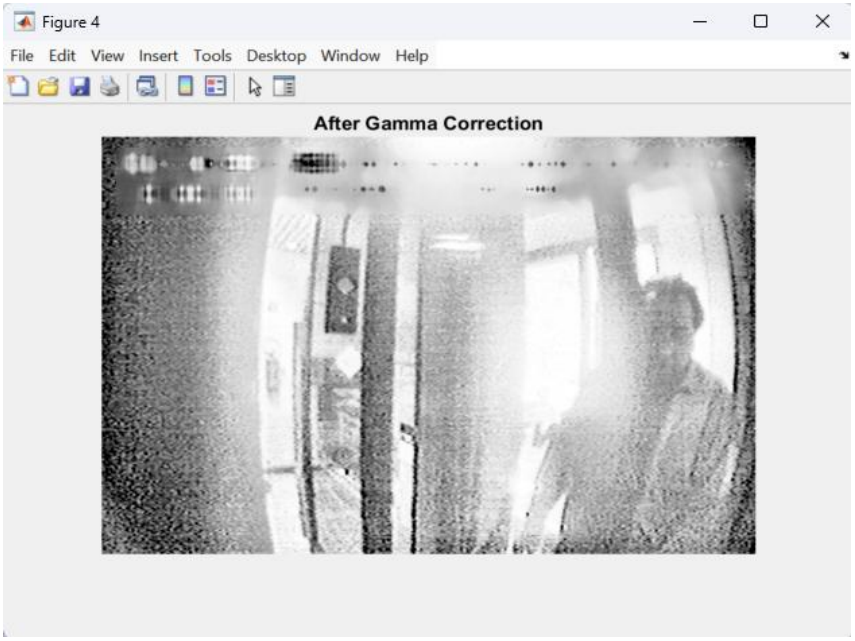

Figure 22: Gamma Correction for Image6

VI. To get clearer results, the main image was resized with the 'imresize' function using bicubic interpolation in line 22 and 23. Because of this, facial features are now sharper and this helps improve recognition in later computer processes.

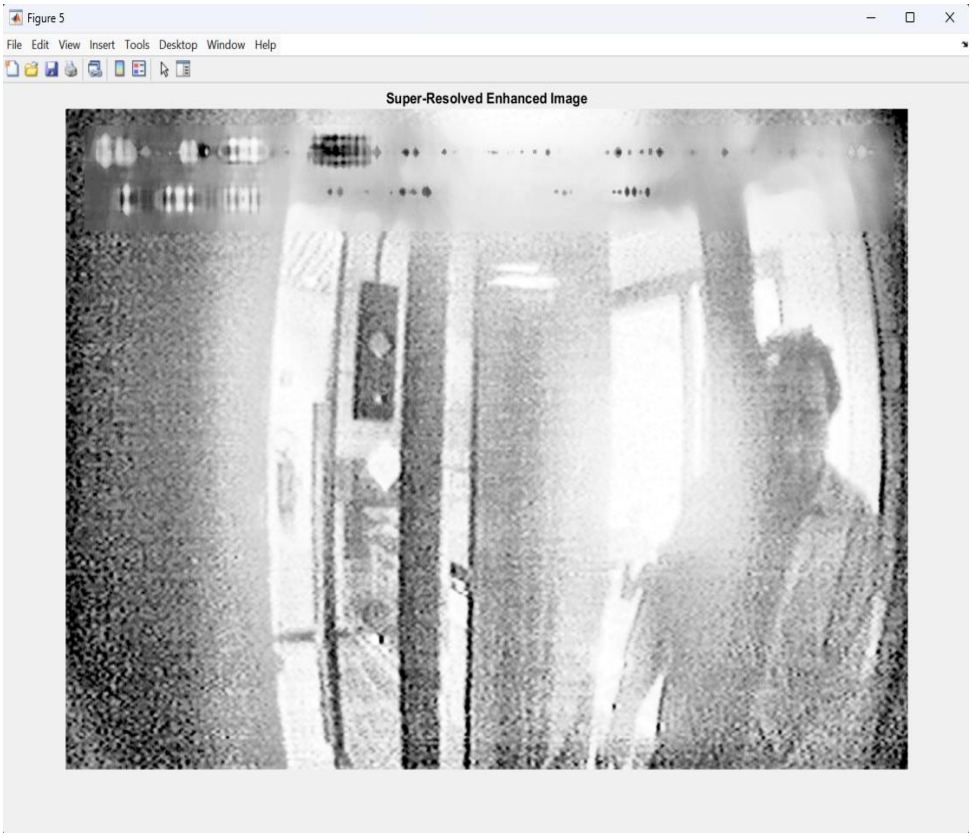

Figure 23: Super-Resolved Enhanced Image of Image6

VII. In line 27-38, face detection was enabled using 'vision.CascadeObjectDetector' which relies on the Viola-Jones algorithm. After enhancing the video, the detector was run and labelled any faces it found with 'insertObjectAnnotation'. As a result, facial improvements enabled the system to find and mark the face in the picture.

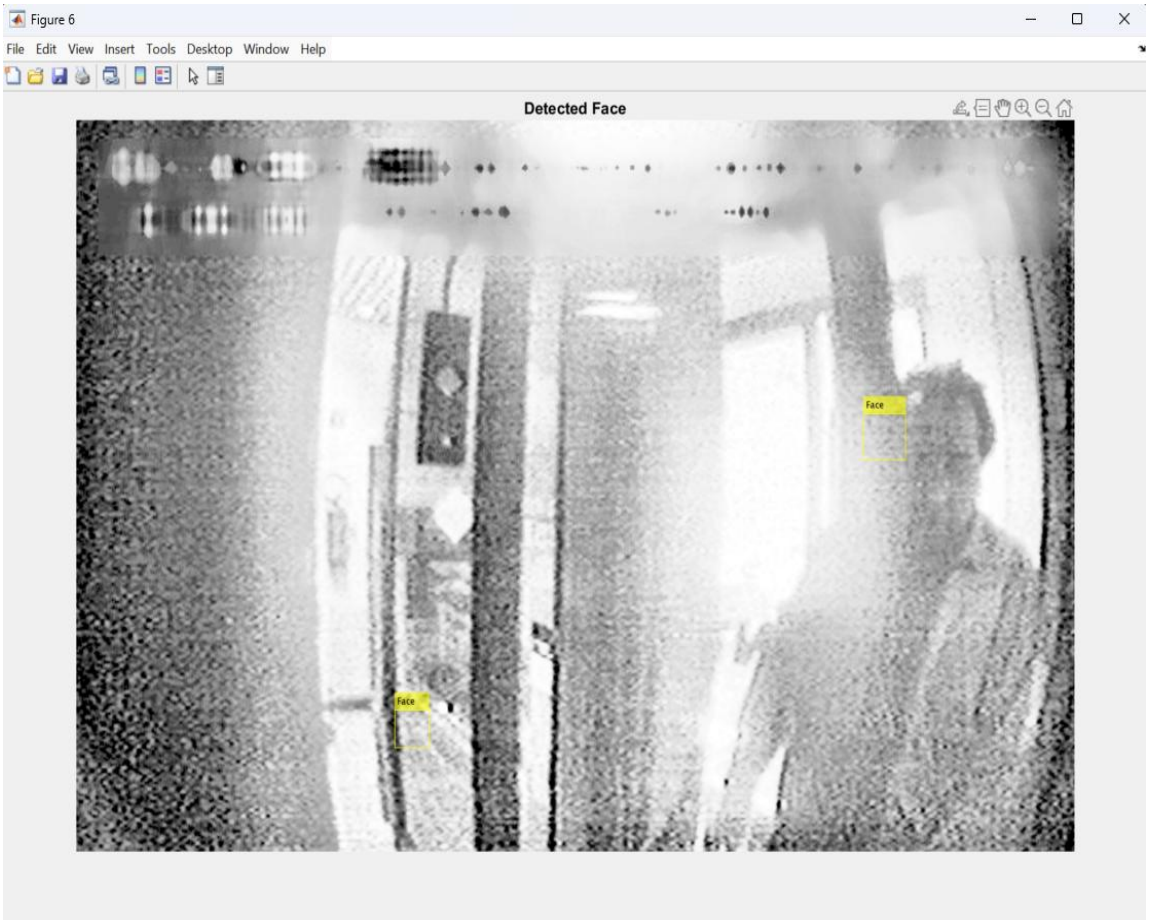

Figure 24: Face Detection in Image6

VIII. In line 40-46, the MATLAB subplot function was used to summarize all stages of the process, from the original image to face detection. With everything arranged in this manner, it was easy to tell how the steps contributed to improving the image and making the subject appear more clearly.

Figure 25: Subplot of all the analysis done for Image6

Due to local and global contrast enhancement, gamma correction and super-resolution, the quality of the image was enhanced (Singh et al., 2017). The face features became easier to see, and the automatic detection of faces became possible, even though it might have failed when working with the original image.

2.7 Discuss and evaluate the step-by-step analysis and processing of Image7.jpg

Figure 26: Original Image7

The picture, Image7.jpg, was taken in a sandy environment, so sand is visible as random spots of brightness in the image. They muddle the image and reduce the differences between dark and light areas. From an image processing point of view, the sand consists of impulsive or high-frequency noise, appearing as bright spots on the picture (Ansari, 2025). The aim is to limit these sandy artifacts and make the picture clearer and easier to view.

Method 1: Grayscale Preprocessing with Median Filtering and Histogram Equalization

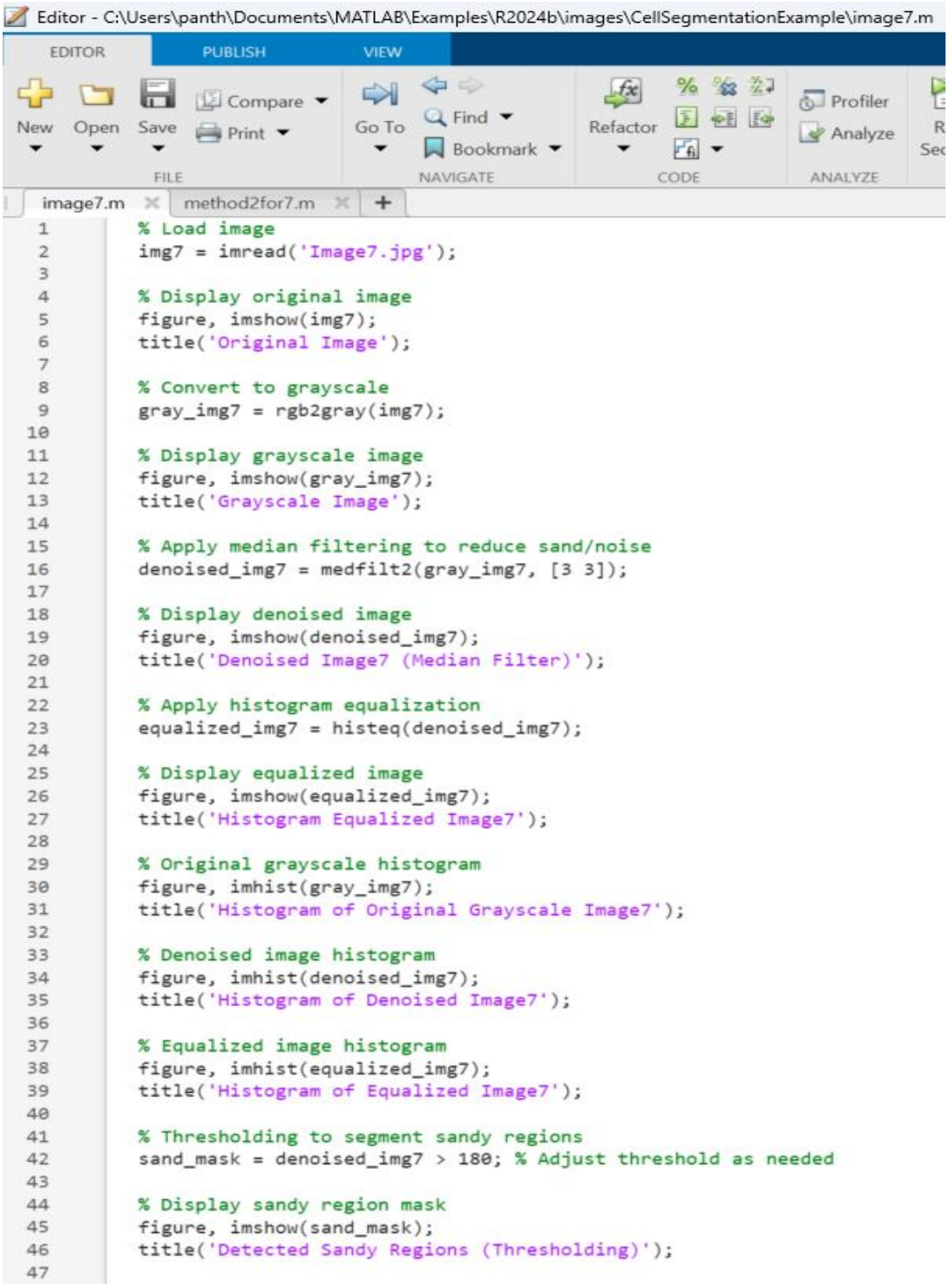

Figure 27: Approach 1 code snippet for image7

Step-by-step analysis:

I. Line 2-6 shows and reads the original RGB image.

II. Line 9-13 makes processing ways easier, as it removes colour input and then handles enhancement and smoothing based on brightness, rather than colours.

Figure 28: Grayscale Image of image7

III. In line 16-20, each pixel in the image is changed to the drawing of the neighbourhood's median, eliminating salt-and-pepper noise while retaining much of the detail around edges.

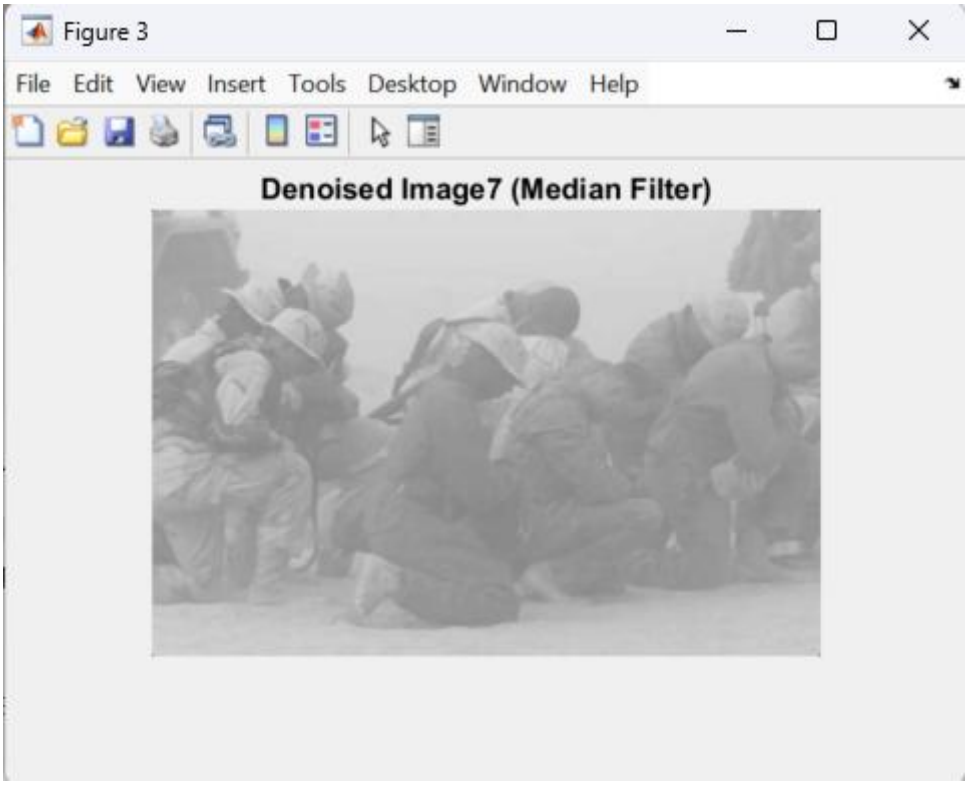

Figure 29: Applied median filter in image7

IV. Line 23-27 changes and intensities to enhance differences between the dark and light areas in the image.

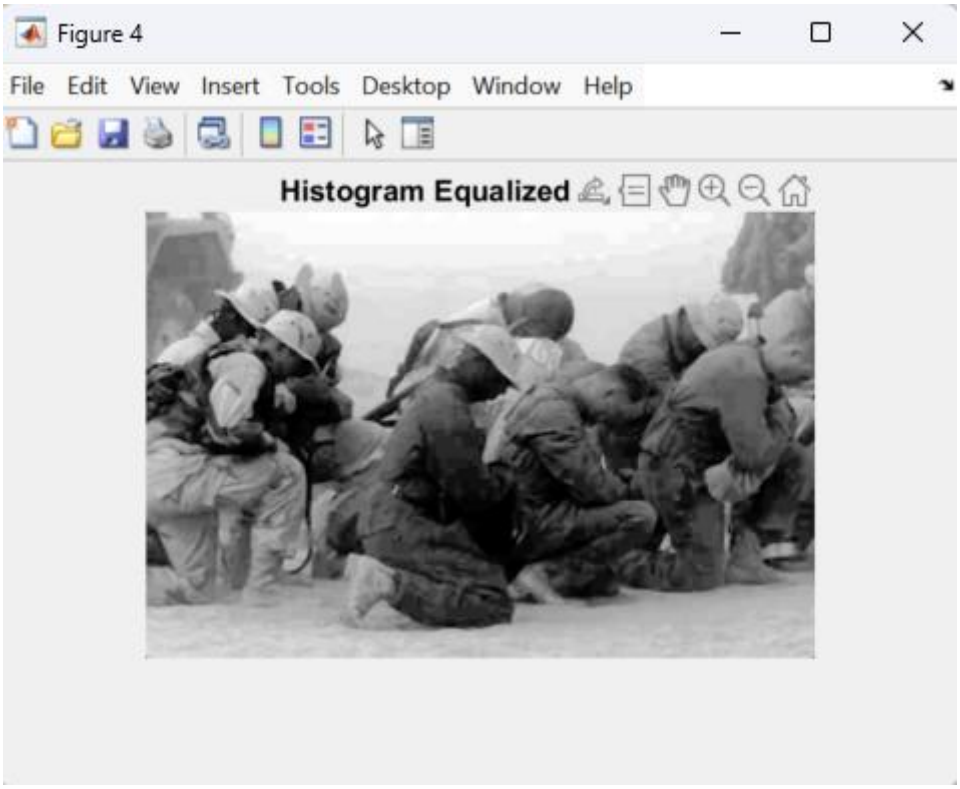

Figure 30: Histogram Equalized in image7

V. Line 30-39 illustrates how the noise level and contrast changes at each step of the process.

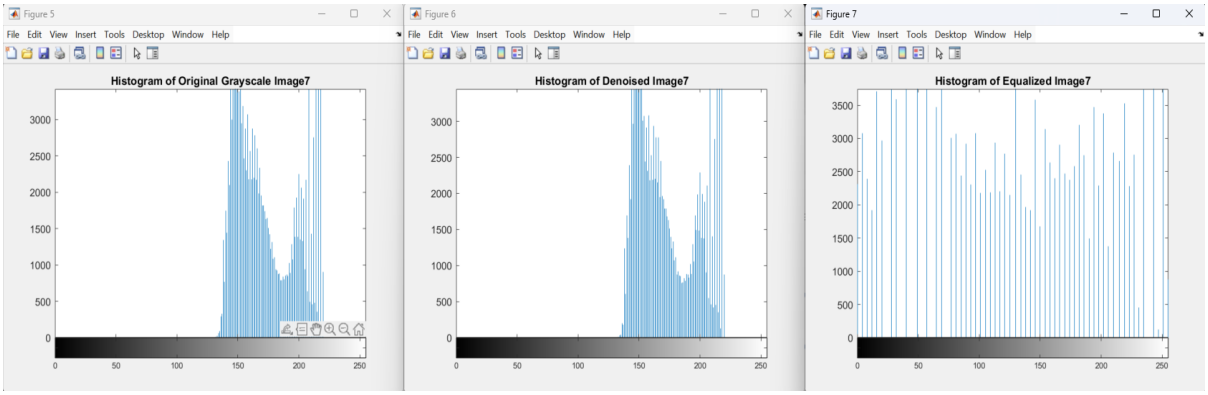

Figure 31: Histogram comparison for all 3 analysis in image7

VI. Line 42-46 highlights areas on the image that contain pixels that are brighter than 180, these are most likely sand. By doing this, you can focus further attention on the potential sources of noise.

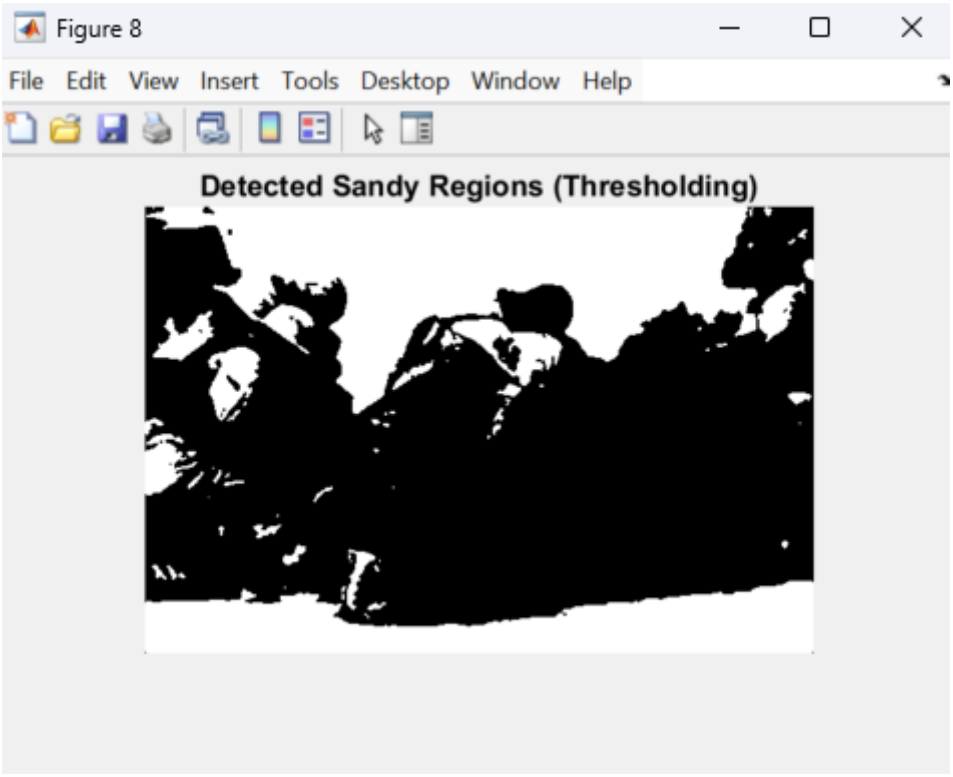

Figure 32: Sandy Regions Detection of image7

Method 2: Equalize the individual colour channels using Gaussian smoothing

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\method2for7.m

image7.m | method2for7.m

```
% Read the color image
img = imread('Image7.jpg');

% Initialize the equalized image
img_eq = img;

% Apply histeq to each RGB channel
for channel = 1:3
    img_eq(:,:,channel) = histeq(img(:,:,channel));
end

% Save histogram equalized image for display
img_histeq = img_eq;

% Apply a slight Gaussian blur to reduce pixelation artifacts
img_smooth = imgaussfilt(img_histeq, 1);  % Gentle smoothing

% Display original, histeq, and smoothed images
figure;
subplot(1, 3, 1), imshow(img), title('Original');
subplot(1, 3, 2), imshow(img_histeq), title('Histogram Equalized');
subplot(1, 3, 3), imshow(img_smooth), title('Smoothed');
```

Figure 33: Approach 2 code snippet for image7

Step-by-step analysis:

I. Line 2 and 5 loads the picture to work on and sets up a variable for the equalized image.

II. In line 8-10, separate contrast enhancement is applied for the red, green and blue channels, so the colours are not modified.

III. Line 16 uses a simple Gaussian blur that blurs all the pixels by 1 to smoothen the edges introduced by its equalization process.

IV. Through line 19-22, differences between the original, equalized and smoothed images can be seen.

Figure 34: Comparison between the analysis done for image7 enhancement

Above methods are successful in enhancing images where sand is a problem. When colour is unimportant and edges should be preserved, using method 1 is very useful for noise removal in grayscale images. With method 2, the benefit is sharper contrast and details while keeping colours, but smoothing artifacts is still necessary, so it is suitable for colour photos.

2.8 Discuss and evaluate the step-by-step analysis and processing of Image8.jpg

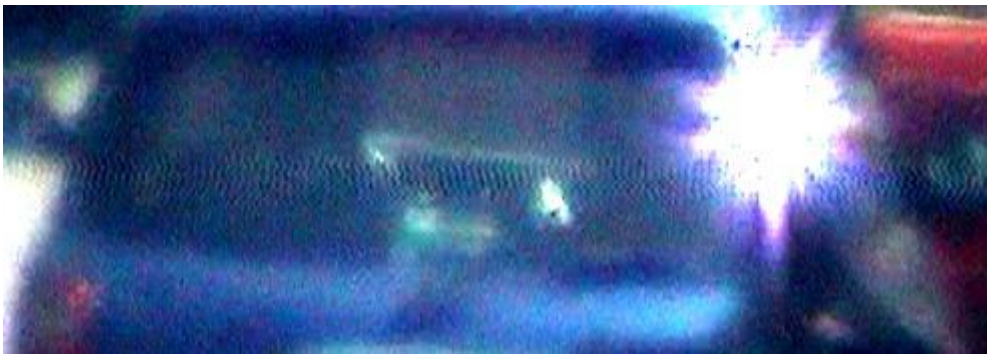

Figure 35: Original Image8

The Image8.jpg has a vehicle in it, but some details are difficult to see because of glare, a bright spot. Due to the super bright light, image details cannot be seen, and the actual scene is hard to decipher. The purpose of this task is to remove the bright patch using a Fourier Transform filter yet keep the original colour unchanged.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\image8.m

image8.m

```
% Read and split color channels
img = im2double(imread('Image8.jpg'));
img_hsv = rgb2hsv(img);  % Convert to HSV
V = img_hsv(:,:,3);  % Value (brightness) channel

% Apply FFT and filtering on V (like before)
F = fft2(V);
Fshift = fftshift(F);
magnitude = log(1 + abs(Fshift));
imshow(mat2gray(magnitude)), title('Magnitude Spectrum');
[x, y] = ginput(4);  % Click bright spots

% Create notch filter (same as before)
[M, N] = size(V);
[X, Y] = meshgrid(1:N, 1:M);
r = 10;
notchFilter = ones(M, N);

for i = 1:size(x,1)
    cx = round(x(i));
    cy = round(y(i));
    notchFilter(((X - cx).^2 + (Y - cy).^2) <= r^2) = 0;
end

% Apply filter and inverse transform
F_filtered = Fshift .* notchFilter;
F_ishift = ifftshift(F_filtered);
V_filtered = real(ifft2(F_ishift));

% Replace filtered V back into HSV and convert to RGB
img_hsv(:,:,3) = mat2gray(V_filtered);
img_result = hsv2rgb(img_hsv);

% Show result
imshow(img_result), title('Filtered Image with Preserved Color');
```

Figure 36: Code snippet for image8 enhancement

Step by step analysis:

I. In line 2-4, the image is reorganized from RGB to HSV to look only at brightness and colour (hue and saturation). The increase in contrast is only for brightness, inside which the glare is visible prominently.

II. In line 7-11, the FFT on two dimensions is used to transform the image into the frequency domain. In the magnitude spectrum, the bright interference patterns are shown as dots in the middle and at the four corners. To choose the unwanted parts, the user have to simply click four times the bright spots.

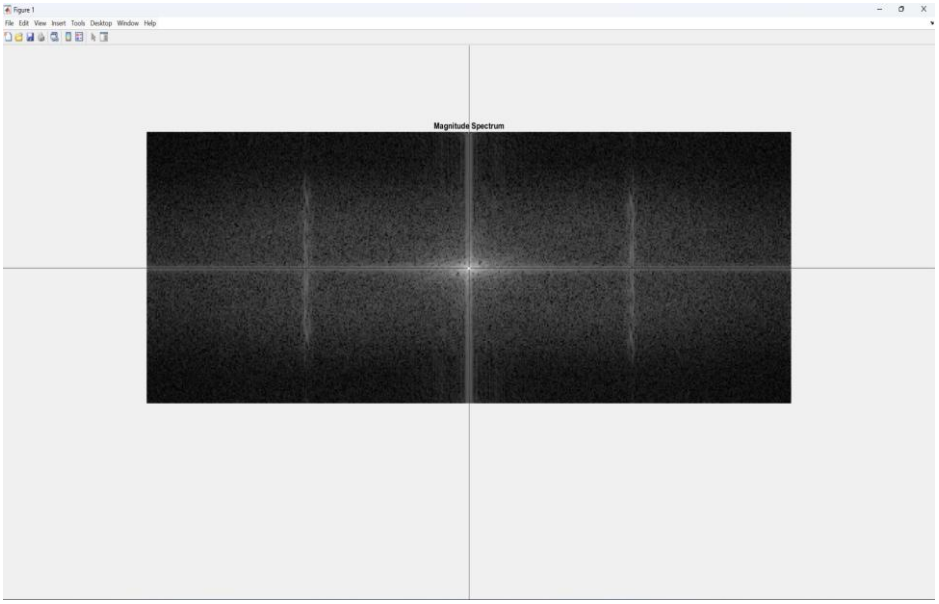

Figure 37: Magnitude Spectrum to click in a bright spot to remove bright spot in image8

III. In line 14-23, a notch filter is created to remove the light frequencies that cause the glare. The spot a user marks are reset within a circular area that has a radius of 10.

IV. In line 26-28, the filter works on the frequency spectrum. The inverse FFT is applied to shift back the spectrum. The result is a lightness channel that has been cleared of the intense glare.

V. After filtering the Value channel, it is normalized and combined with the HSV photo in line 31. After that, in line 32, the data is displayed using the RGB system. And in line 35, the image is free from glare, while all its original colours are still intact.

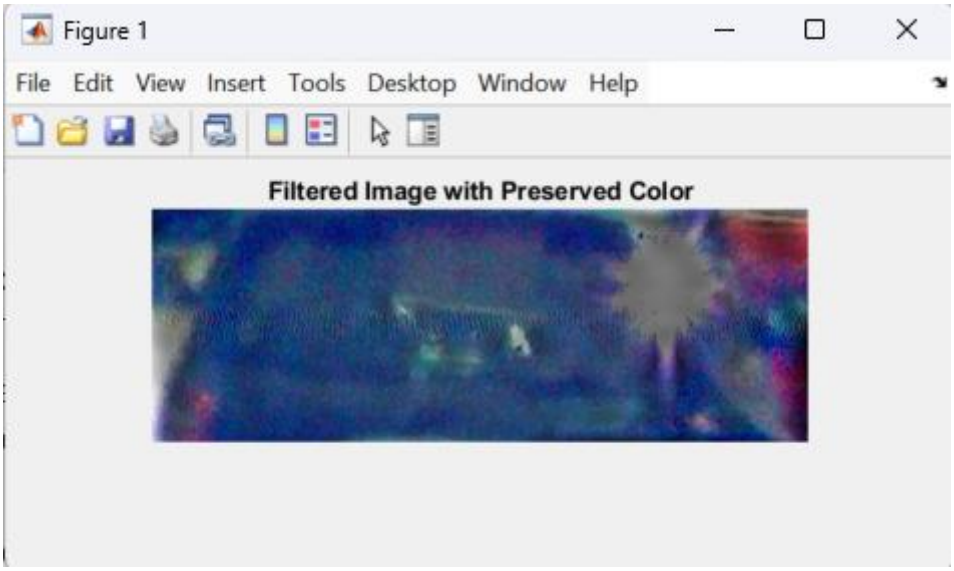

Figure 38: Output after the enhancement in image8

The photo was aligned in HSV space and the brightness channel was edited to remove unwanted high-frequency elements leading to the disturbance. Choosing the right spots in the frequency spectrum and filtering them out reduced the glare. After filtering the brightness, it was combined once again with the original colour data from the original image.

2.9 Discuss and evaluate the step-by-step analysis and processing of image sequence

Figure 39: vespa001.jpg

You have to process each of the images from vespa001.jpg to vespa240.jpg to find and recognize the number plate of the bike. By performing frame-by-frame stabilization, ROI selection, feature-based tracking, image enhancement and then making the video output with the text of the found number plate. The camera is not moving as the object (vehicle) moves.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Img_Sequence.m

Img_Sequence.m

```
% --- SETTINGS ---
imageFolder = fullfile(getenv('USERPROFILE'), 'Documents', 'MATLAB', 'image sequence');
outputVideoPath = 'stabilized_annotated_output.avi';

% --- READ IMAGE SEQUENCE ---
imageFiles = dir(fullfile(imageFolder, 'vespa*.jpg'));
[~, idx] = sort({imageFiles.name});
imageFiles = imageFiles(idx);
numFrames = length(imageFiles);
videoFrames = cell(1, numFrames);

for k = 1:numFrames
    videoFrames{k} = imread(fullfile(imageFolder, imageFiles(k).name));
end

% --- INITIALIZE STABILIZATION ---
firstGray = rgb2gray(videoFrames{1});
pointsPrev = detectSURFFeatures(firstGray);
[featuresPrev, pointsPrev] = extractFeatures(firstGray, pointsPrev);
tforms(numFrames) = affine2d(eye(3));

for k = 2:numFrames
    grayCurr = rgb2gray(videoFrames{k});
    pointsCurr = detectSURFFeatures(grayCurr);
    [featuresCurr, pointsCurr] = extractFeatures(grayCurr, pointsCurr);

    indexPairs = matchFeatures(featuresPrev, featuresCurr, 'Unique', true);
    matchedPrev = pointsPrev(indexPairs(:,1));
    matchedCurr = pointsCurr(indexPairs(:,2));

    if length(matchedPrev) >= 4
        tforms(k) = estimateGeometricTransform2D(matchedCurr, matchedPrev, 'similarity');
        tforms(k).T = tforms(k-1).T * tforms(k).T;
    else
        tforms(k) = tforms(k-1); % fallback
    end

    pointsPrev = pointsCurr;
    featuresPrev = featuresCurr;
end
```

Figure 40: Code snippet Part 1 for Image Sequence

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Img_Sequence.m

Img_Sequence.m

```
% --- STABILIZE VIDEO ---
outputView = imref2d(size(videoFrames{1}));
stabilizedFrames = cell(1, numFrames);
for k = 1:numFrames
    stabilizedFrames{k} = imwarp(videoFrames{k}, tforms(k), 'OutputView', outputView);
end

% --- SELECT NUMBER PLATE ON FIRST FRAME ---
firstFrame = stabilizedFrames{1};
figure; imshow(firstFrame);
title('Draw a box around the number plate');
plateROI = round(getPosition(imrect)); close;

% --- INITIALIZE POINT TRACKER FOR PLATE TRACKING ---
grayFrame = rgb2gray(firstFrame);
roiGray = imcrop(grayFrame, plateROI);
points = detectMinEigenFeatures(roiGray);
points.Location(:,1) = points.Location(:,1) + plateROI(1);
points.Location(:,2) = points.Location(:,2) + plateROI(2);

tracker = vision.PointTracker('MaxBidirectionalError', 2);
initialize(tracker, points.Location, grayFrame);
trackedBBoxes = zeros(numFrames, 4);
trackedBBoxes(1,:) = plateROI;

% --- TRACK PLATE ACROSS FRAMES ---
for k = 2:numFrames
    frameGray = rgb2gray(stabilizedFrames{k});
    [points, validity] = tracker(frameGray);
    validPoints = points(validity, :);

    if size(validPoints, 1) >= 2
        centerShift = mean(validPoints, 1) - mean(points(validity,:), 1);
plateCenter = [plateROI(1) + plateROI(3)/2, plateROI(2) + plateROI(4)/2];
newCenter = plateCenter + centerShift;
plateROI(1:2) = round(newCenter - [plateROI(3), plateROI(4)] / 2);
trackedBBoxes(k,:) = plateROI;
        setPoints(tracker, validPoints);
    else
        trackedBBoxes(k,:) = trackedBBoxes(k-1,:);
    end
end
```

Figure 41: Code snippet Part 2 for Image Sequence

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Img_Sequence.m

Img_Sequence.m

```
% --- PREPARE VIDEO WRITER ---
v = VideoWriter(outputVideoPath);
v.FrameRate = 15;
open(v);

% --- LOOP: ENHANCE, OCR, ANNOTATE, SAVE TO VIDEO ---
for k = 1:numFrames
    frame = stabilizedFrames{k};
    box = trackedBBoxes(k,:);
    cropped = imcrop(frame, box);

    % --- ENHANCE FOR OCR ---
    if size(cropped, 3) == 3
        hsv = rgb2hsv(cropped);
        hsv(:,:,3) = adapthisteq(hsv(:,:,3), 'ClipLimit', 0.03, 'NumTiles', [8 8]);
        enhanced = hsv2rgb(hsv);
    else
        enhanced = adapthisteq(cropped);
    end

    enhanced = imsharpen(enhanced, 'Amount', 2.5, 'Radius', 1.5);
    resized = imresize(enhanced, 3, 'bicubic');
    denoised = imnlmfilt(resized);
    finalPlate = imadjust(denoised, stretchlim(denoised), [], 1.2);

    % --- OCR ---
    try
        ocrResult = ocr(finalPlate, 'TextLayout', 'Line');
        textDetected = strtrim(ocrResult.Text);
    catch
        textDetected = '';
    end

    % --- ANNOTATE FRAME ---
    annotated = insertShape(frame, 'Rectangle', box, 'Color', 'yellow', 'LineWidth', 3);
    annotated = insertText(annotated, [box(1), box(2)-20], textDetected, ...
        'BoxColor', 'black', 'TextColor', 'white', 'FontSize', 16);

    % --- WRITE TO VIDEO ---
    writeVideo(v, annotated);
end
```

Figure 42: Code snippet Part 3 for Image Sequence

Step-by-step analysis:

I. Used dir() to gain access to all the images and store them in a cell array.

II. Features from SURF are compared and matched for each consecutive frame. A series of matrices are formed to ensure that images are aligned as intended.

III. Each frame is altered with the cumulative transform to reduce the jitter.

IV. An outline of a rectangle is sketched around the plate in the stabilized first frame.

V. Tracks various strong corner features in the chosen region of interest with the help of 'vision.PointTracker'.

VI. Moved frame by frame and changed the coordinates found in the bounding box accordingly.

VII. Conversion to HSV and CLAHE performed only on the value channel. Improve edge quality, adjust the size (×3) and despeckle with non-local means. Adjust contrast.

VIII. The ocr() function is used to interpret the characters in the enhanced plate.

IX. Marked rectangle on the image and had the text from the OCR appear in each frame. Then all the annotated frames are into a new .avi video.

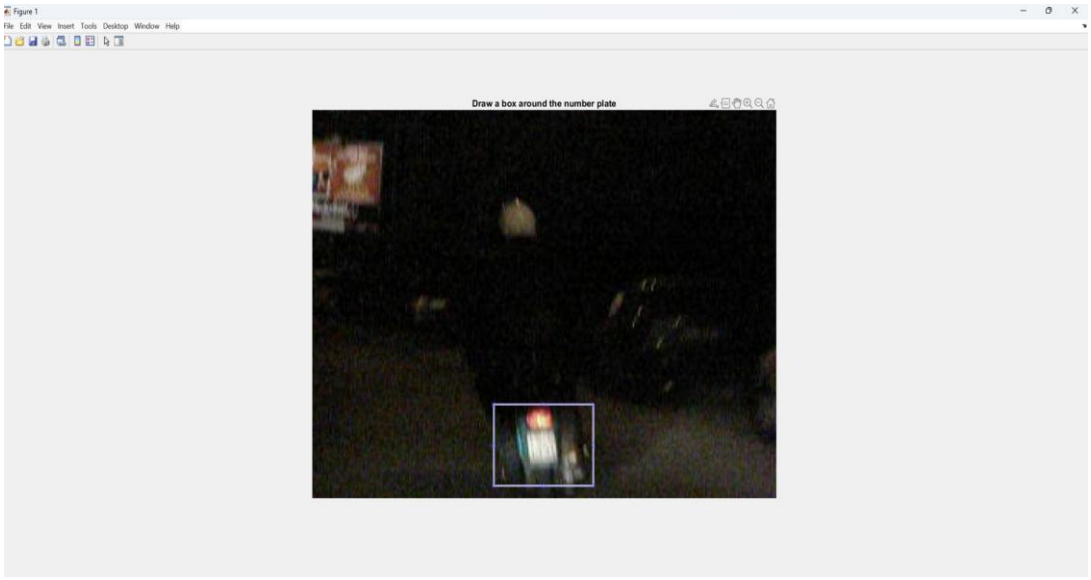

Figure 43: Marked area for the number plate detection

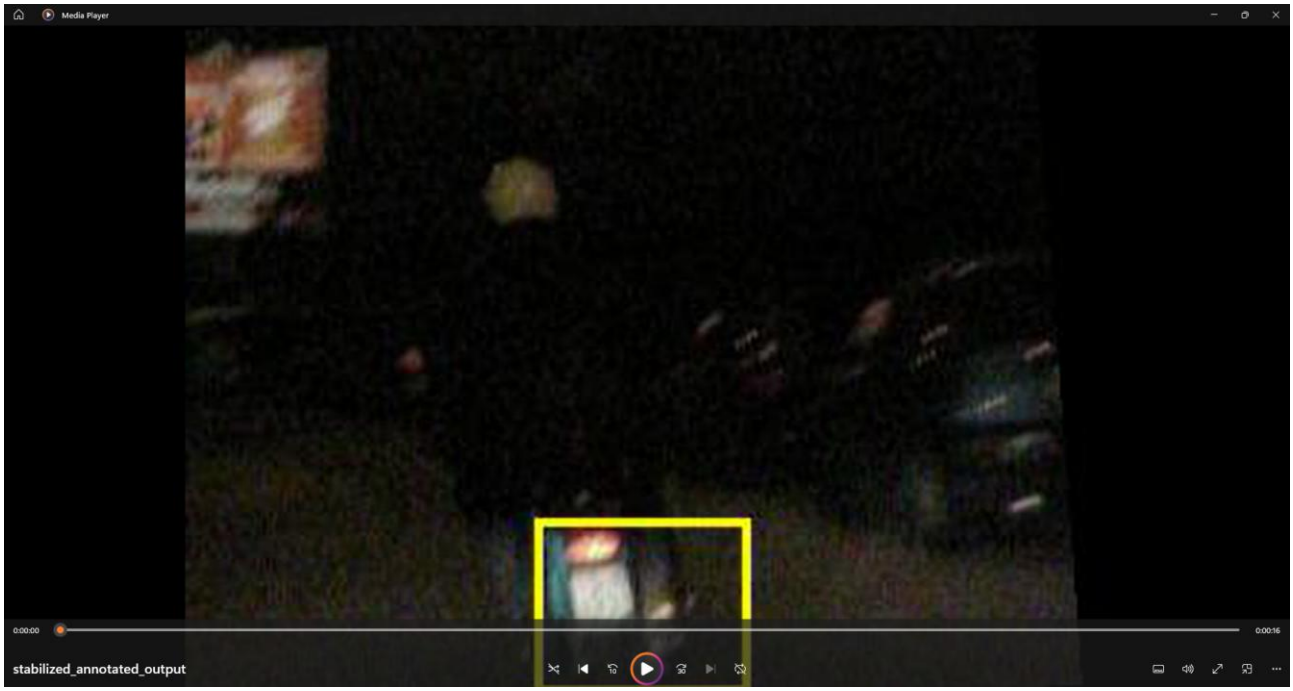

Figure 44: First frame of the video

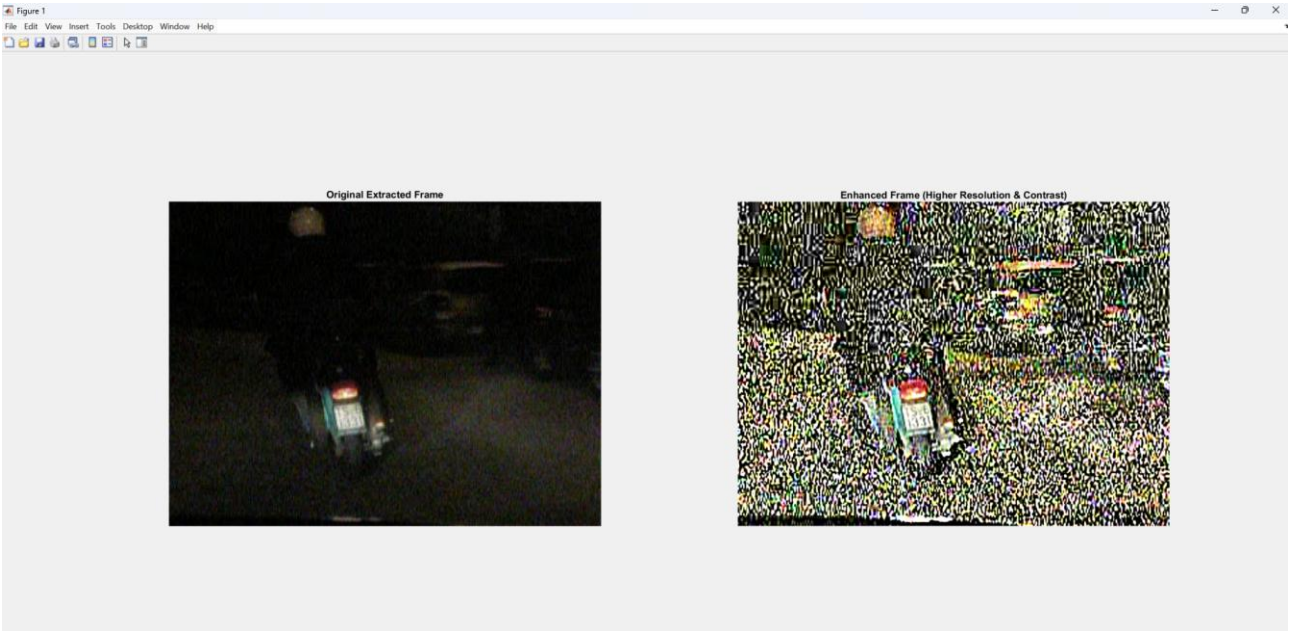

Figure 45: Comparison after enhancement

The process has smoothed out the sequence of images, followed the bike's license plate and applied various methods to decipher the plate using OCR. After that, the points are added to the original video to clarify them and make the report clearer. '1331' can be seen in number plate.

2.10 Discuss and evaluate the step-by-step analysis and processing of video1.mp4

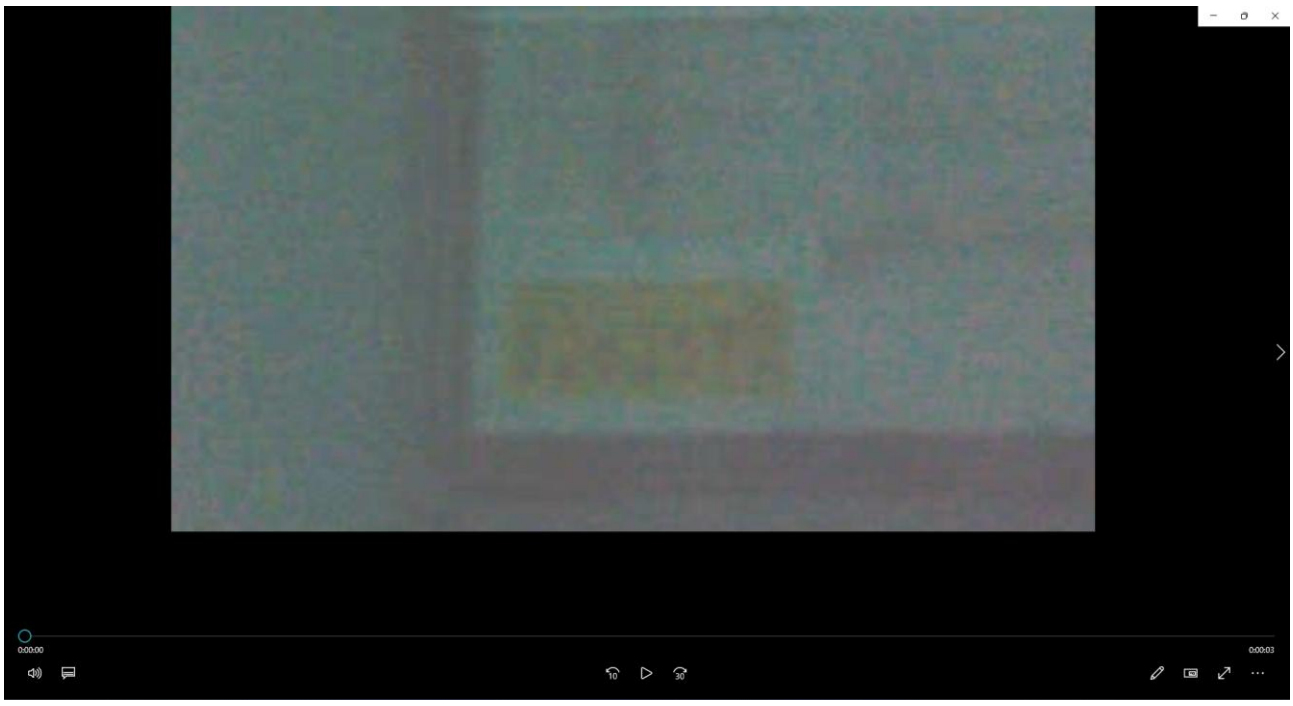

Figure 46: First frame of Video-1

This section carries out three tasks: video stabilization, reducing motion blur and making the contrast clearer in the shaky and motion-blurred video.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\video1.m *

video1.m * | Vid1.m | +

```
%% Load Video
videoFile = 'Video-1.mp4';
v = VideoReader(videoFile);
writer = VideoWriter('stabilized_output.avi');
open(writer);

%% 1. Video Stabilization using SURF + PointTracker
firstFrame = readFrame(v);
grayFirst = rgb2gray(firstFrame);
points = detectSURFFeatures(grayFirst);
[features, validPoints] = extractFeatures(grayFirst, points);

tracker = vision.PointTracker('MaxBidirectionalError', 1);
initialize(tracker, validPoints.Location, grayFirst);

frameCount = 1;
stabilizedFrames{frameCount} = im2double(firstFrame);
writeVideo(writer, firstFrame);

while hasFrame(v)
    frame = readFrame(v);
    gray = rgb2gray(frame);

    [newPoints, valid] = tracker(gray);

    if sum(valid) > 10
        % Compute transformation
        tform = estimateGeometricTransform2D(validPoints.Location(valid,:), newPoints(valid,:), 'similarity');
        stabilized = imwarp(frame, tform, 'OutputView', imref2d(size(gray)));
        validPoints = cornerPoints(newPoints(valid,:));
    else
        stabilized = frame; % fallback if tracking fails
    end

    frameCount = frameCount + 1;
    stabilizedFrames{frameCount} = im2double(stabilized);
    writeVideo(writer, stabilized);

    setPoints(tracker, newPoints(valid,:));
end
close(writer);
```

Figure 47: Code snippet part 1 for Video-1

```
%% 2. Frame Averaging (for Motion Blur Compensation)
numAvgFrames = min(5, frameCount);
avgFrame = mean(cat(4, stabilizedFrames{1:numAvgFrames}), 4);

%% 3. CLAHE Histogram Equalization (Global)
hsv = rgb2hsv(avgFrame);
v_channel = im2uint8(mat2gray(hsv(:,:,3)));
v_eq = adapthisteq(v_channel, 'ClipLimit', 0.03, 'NumTiles', [8 8]);
hsv(:,:,3) = im2double(v_eq);
enhanced = hsv2rgb(hsv);

%% 4. Display Enhanced Result
figure;
imshow(enhanced);
title('Stabilized and Enhanced Frame');
```

Figure 48: Code snippet part 2 for video-1

Step-by-step analysis:

I. Line 2-5 opens the video you want to process and configure the video writer to store the result.

II. Line 8-14 looks for specific points in the first frame by applying SURF.

III. Line 16-41 follows the movement between shots and distort the footage to even out the video.

IV. Line 44 and 45 cuts down on motion blur using temporal averaging.

V. Line 48-52 increases the difference between the light and dark parts of the image.

VI. Line 55-57 puts forward the final edge detection picture.

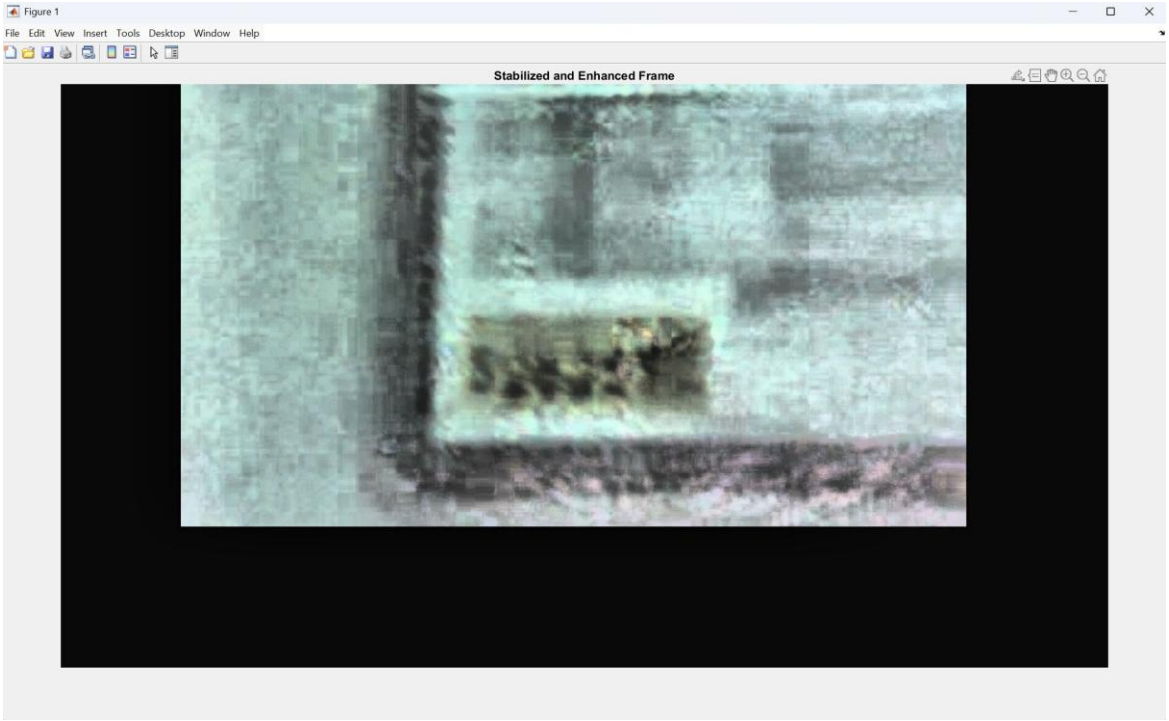

Figure 49: Final Edge Detection of the video-1

It starts by using SURF and a Point Tracker to stabilize movements between each frame of the video. After that, by taking the first few steady frames and averaging them, it makes the video clearer to avoid motion blur. Lastly, the tool applies CLAHE (Contrast Limited Adaptive Histogram Equalization) to increase both the brightness and contrast, especially when the light is not even. As a result, you can now use OCR or detect objects with more accuracy and reliability. The code stabilizes an unsteady video by relying on SURF features and geometric transformation, deals with motion blur by making the first five frames look more similar and improves the overall contrast in the video. The end product is stored as a stabilized video and shown in a single enhanced frame for viewing.

2.11 Discuss and evaluate the step-by-step analysis and processing of video2.avi

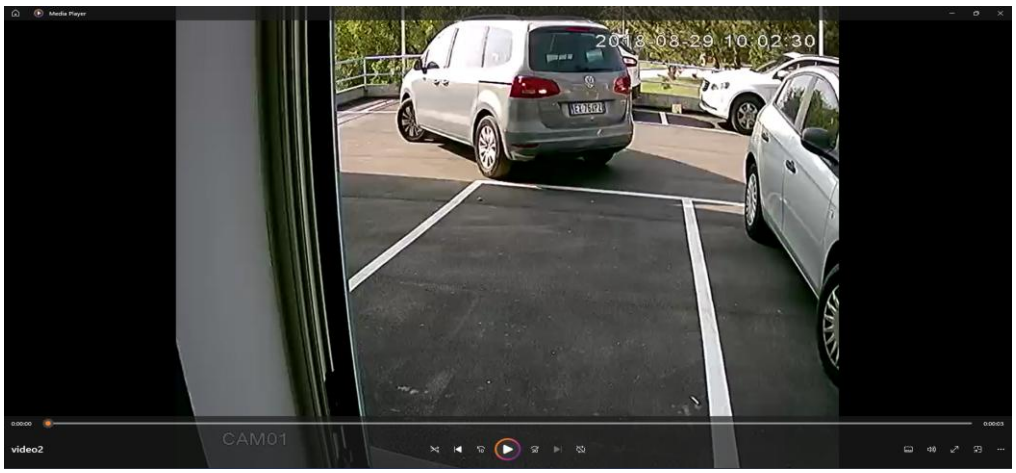

Figure 50: First frame of video2

This section takes one image from video2.avi to help determine the car's number plate and what colour it is. The user chooses the area they want to examine on the frame, after which image processing methods are used to sharpen it. The script also examines the colour of nearby cars to find out which colour is common by comparing the hue of the picture. It then reads the characters on the license plate with OCR and displays the information found.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Video2.m

Video2.m

```
% Load and display original frame
videoReader = VideoReader('video2.avi');
frameNum = 1;
videoReader.CurrentTime = (frameNum-1)/videoReader.FrameRate;
frame = readFrame(videoReader);
imshow(frame);
title('Original Frame - Select Plate Area');

% Interactive crop with error handling
try
    plateRegion = drawrectangle('Color','r');
    cropped = imcrop(frame, plateRegion.Position);
    if isempty(cropped)
        error('No region selected');
    end
catch
    error('Plate selection failed');
end

% --- Plate Enhancement ---

% Convert to grayscale
grayPlate = rgb2gray(cropped);

% Super-resolution approach (3x upscale)
upscaled = imresize(grayPlate, 3, 'bicubic');

% CLAHE
[h, w] = size(upscaled);
tilesX = max(2, min(8, floor(w / 32)));
tilesY = max(2, min(8, floor(h / 32)));
enhanced = adapthisteq(upscaled, 'ClipLimit', 0.01, 'NumTiles', [tilesY tilesX]);

% Denoise and Sharpen
denoised = medfilt2(enhanced, [3 3]);
sharpened = imsharpen(denoised, 'Amount', 2, 'Radius', 0.5);
final = locallapfilt(sharpened, 0.8, 0.5);

% --- OCR (Optional) ---
ocrResult = ocr(final);
plateText = strtrim(ocrResult.Text);
```

Figure 51: Code snippet part 1 for video2

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Video2.m

Video2.m

```
% --- Car Color Detection ---
% Expand the selection slightly to include surrounding car area
expandedPos = plateRegion.Position;
expandedPos(1) = max(1, expandedPos(1) - 30); % left
expandedPos(2) = max(1, expandedPos(2) - 30); % top
expandedPos(3) = min(size(frame,2) - expandedPos(1), expandedPos(3) + 60); % width
expandedPos(4) = min(size(frame,1) - expandedPos(2), expandedPos(4) + 60); % height

carRegion = imcrop(frame, expandedPos);

% Convert to HSV and get dominant hue
hsvImage = rgb2hsv(carRegion);
hue = hsvImage(:,:,1);
saturation = hsvImage(:,:,2);
value = hsvImage(:,:,3);

% Mask low saturation (gray/white/black areas)
colorMask = saturation > 0.2 & value > 0.2;
hueMasked = hue(colorMask);

% Dominant color estimation
if isempty(hueMasked)
    carColor = 'Unknown';
else
    avgHue = mean(hueMasked);

    % Convert average hue to color name
    if avgHue < 0.04 || avgHue > 0.95
        carColor = 'Red';
    elseif avgHue < 0.10
        carColor = 'Orange';
    elseif avgHue < 0.18
        carColor = 'Yellow';
    elseif avgHue < 0.45
        carColor = 'Green';
    elseif avgHue < 0.65
        carColor = 'Cyan/Blue';
    elseif avgHue < 0.85
        carColor = 'Blue/Purple';
    else
        carColor = 'Unknown';
```

Figure 52: Code snippet part 2 for video2

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\Video2.m

Video2.m

```
        carColor = 'Yellow';
    elseif avgHue < 0.45
        carColor = 'Green';
    elseif avgHue < 0.65
        carColor = 'Cyan/Blue';
    elseif avgHue < 0.85
        carColor = 'Blue/Purple';
    else
        carColor = 'Unknown';
    end
end

% --- Display Results ---
figure;
subplot(2,2,1); imshow(cropped); title('Original Plate Crop');
subplot(2,2,2); imshow(final); title('Enhanced Plate');
subplot(2,2,3); imshow(carRegion); title(['Car Region (Color: ', carColor, ')']);
subplot(2,2,4); imshow(final); title(['OCR Result: ', plateText]);
```

Figure 53: Code snippet part 3 for video2

Step-by-step analysis:

I. Loads video frames and can see the fields in the first video frame and can interact with them.

II. Outlined a defensible area on the plate; so that the software will only keep that part of the frame.

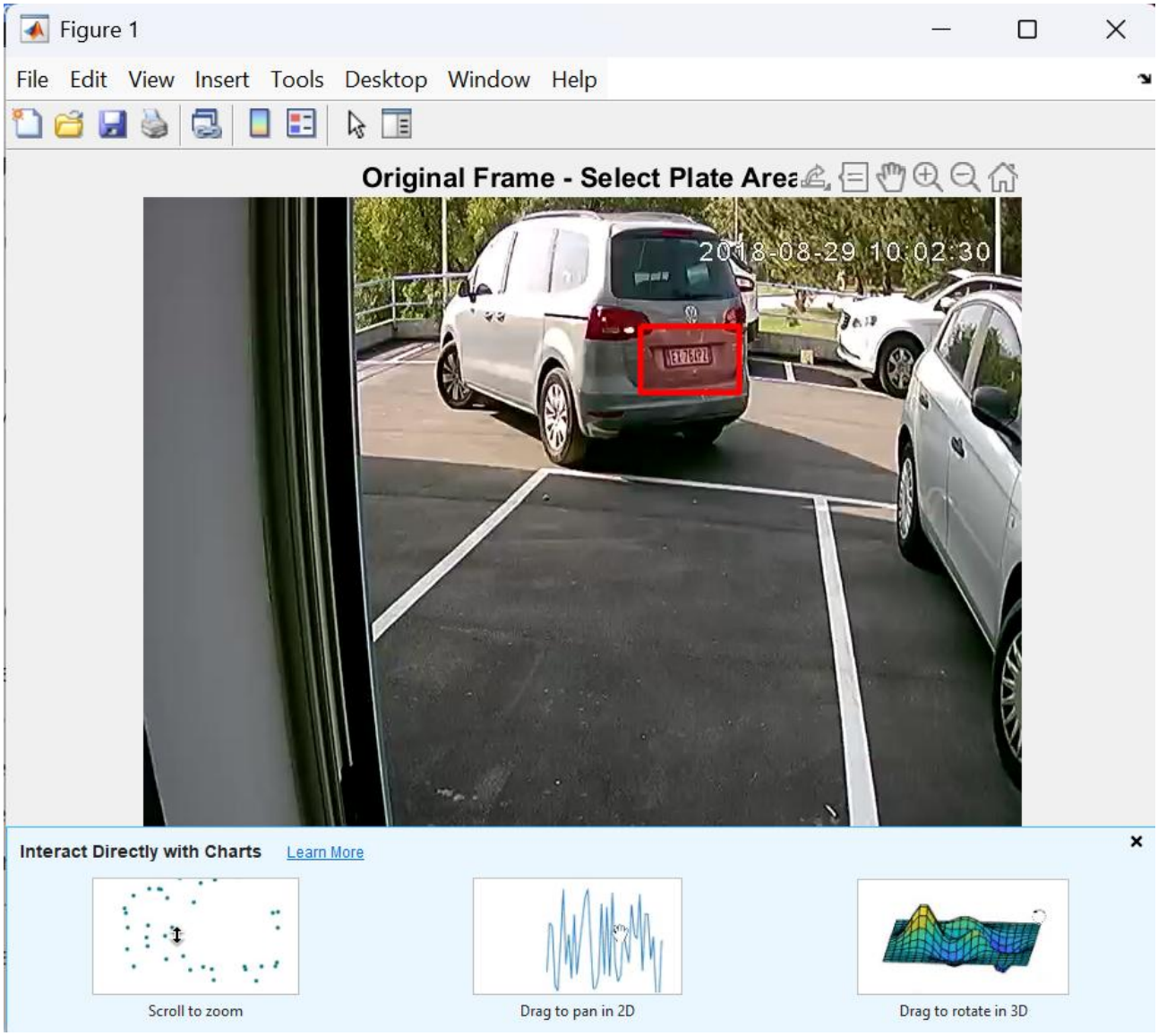

Figure 54: Selected Number Plate Area of video2

III. Made the cropped plate grayscale, increased its resolution 3 times using bicubic interpolation and uses CLAHE to boost the contrast. At the next step, it got rid of noise, sharpened the photo and refined the edges.

IV. Enhanced and analysed the plate to pull out any clear text and cut the string to only show the necessary data.

V. Enlarged the selected area on the plate region to take in car body areas, crops this new square, replaced it with HSV colours and got out pixels of the proper brightness and saturation. Then, it measured the average hue on every image to identify the most common car colour range.

VI. Featured the original cropped plate, the enhanced plate, the area with colour labels and the OCR output in a 2x2 grid.

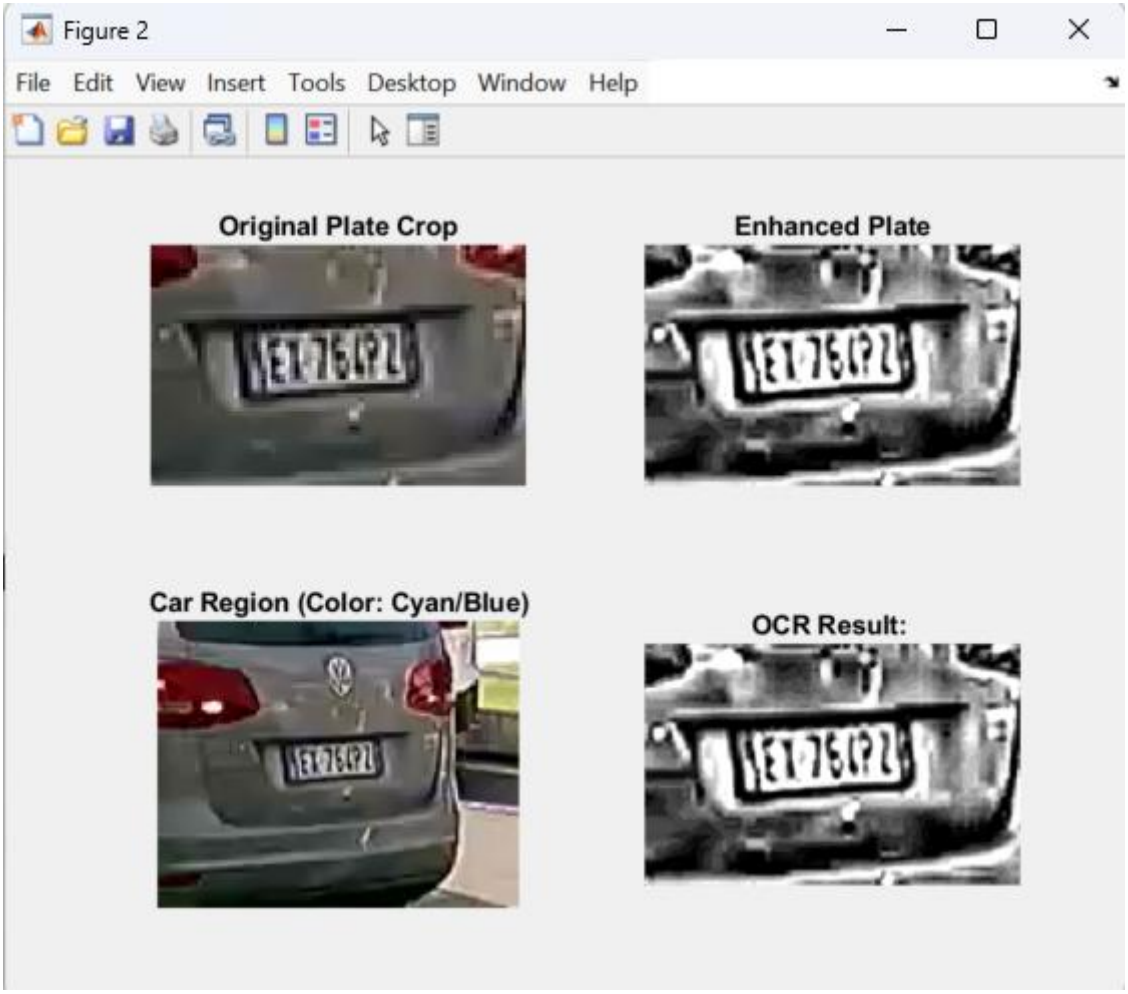

Figure 55: Final output of the video2

The code captures a video frame and allows the user to identify an area where the license plate has to be searched. It sharpens the photo of the license plate and runs OCR over it to obtain the plate number. Besides, it checks the neighbouring pixels around the plate to recognize the main colour of the car with HSV hue values. To verify the results: the improved plate, the detected colour and the OCR text can be seen.

2.12 Discuss and evaluate the step-by-step analysis and processing of video3.avi

Figure 56: First frame of video3

By using the code, the video3.avi file will look sharper and cleaner than it did before. Ensures the first 5 frames are aligned using normalized cross-correlation and processes them with median averaging. Once that is done, it uses upscaling and CLAHE to improve contrast, followed by sharpening to improve the quality of the image.

Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\video3.m

video3.m

```
%% Load video and initialize
video = VideoReader('video3.avi');
numFrames = floor(video.Duration * video.FrameRate);
fprintf('Total Frames: %d\n', numFrames);

% Read first frame as reference
refRGB = im2double(readFrame(video));
frameSize = size(refRGB);
refGray = im2gray(refRGB);

% Stack for aligned frames
stackRGB = zeros([frameSize, 5]);

%% Align next 5 frames to reference using cross-correlation
stackRGB(:,:,:,1) = refRGB;  % store reference

for k = 2:5
    frameRGB = im2double(readFrame(video));
    currentGray = im2gray(frameRGB);

    % Estimate shift via normalized cross-correlation
    c = normxcorr2(refGray, currentGray);
    [~, maxIdx] = max(c(:));
    [ypeak, xpeak] = ind2sub(size(c), maxIdx);
    corr_offset = [xpeak - size(refGray,2), ypeak - size(refGray,1)];

    % Align RGB and grayscale frames
    alignedRGB = imtranslate(frameRGB, -corr_offset, 'FillValues', 0);
    stackRGB(:,:,:,k) = alignedRGB;
end

%% Median averaging across aligned frames
avgRGB = median(stackRGB, 4);

%% Super-resolution (2x upscale)
superRes = imresize(avgRGB, 2, 'bicubic');
```

Figure 57: Code snippet part 1 for video3

```
Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\video3.m
video3.m
33  avgRGB = median(stackRGB, 4);
34
35  %% Super-resolution (2x upscale)
36  superRes = imresize(avgRGB, 2, 'bicubic');
37
38  %% CLAHE + Sharpening for each RGB channel
39  enhanced = zeros(size(superRes));
40  for c = 1:3
41      ch = superRes(:,:,c);
42      ch = adapthisteq(ch);  % CLAHE enhancement
43      enhanced(:,:,c) = imsharpen(ch, 'Radius', 1, 'Amount', 1.2);  % sharpened
44  end
45
46  %% Display output
47  figure;
48  subplot(1,3,1); imshow(refRGB); title('Original Frame');
49  subplot(1,3,2); imshow(avgRGB); title('Median Averaged');
50  subplot(1,3,3); imshow(enhanced); title('Super-Resolved & Enhanced');
```

Figure 58: Code snippet part 2 for video3

Step-by-step analysis:

I. *Line 2-4 p*lays the video. *, c*alculates the total number of frames.

```
Editor - C:\Users\panth\Documents\MATLAB\Examples\R2024b\images\CellSegmentationExample\vi
video3.m
1  %% Load video and initialize
2  video = VideoReader('video3.avi');
3  numFrames = floor(video.Duration * video.FrameRate);
4  fprintf('Total Frames: %d\n', numFrames);
```

```
Command Window
>> video3
Total Frames: 69
fx >>
```

Figure 59: Frame count of video3

II. Line 7-9 considers the first frame as the basis. Makes the image grayscale, as this helps with alignment.

III. Line 12-15 initializes an array in 4 dimensions to keep 5 RGB frames aligned. The initial plot of each film is saved just as it is.

IV. Line 17-30 feeds the vision transformation with the following 4 frames. It performs spatial shifting by using 'normxcorr2'. It uses translating to make sure frames are correctly lined up.

V. Line 33 3liminates noise from the aligned frames by averaging them over time using the median, since the median can handle outliers better than the mean.

VI. Line 36 upscales the image by double the resolution with the bicubic interpolation technique.

VII. Line 40-44 uses CLAHE to boost the contrast in every channel. Having sharper edges allows you to notice finer points.

VIII. Line 47-50 shows the original, median-averaged and enhanced improved picture side-by-side.

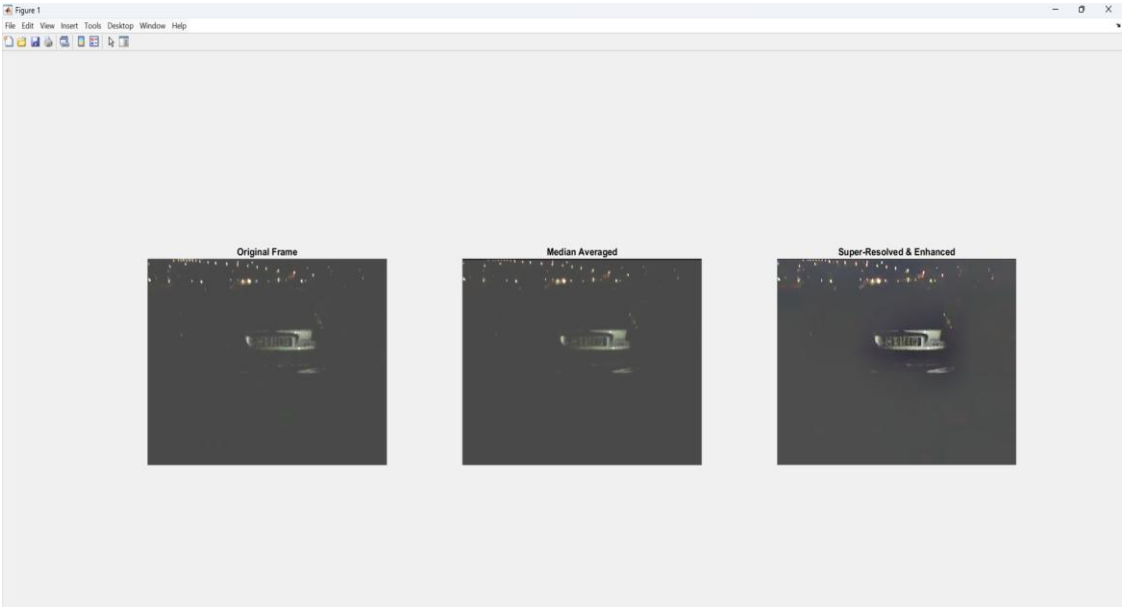

Figure 60: Comparison of video3 after enhancement

The five video frames are lined up, piled together and improved during the multi-frame super-resolution process with the help of code. It relies on cross-correlation, resizing using bicubic interpolation and extra user enhancements to improve both the detailed and sharp appearance of images. When you see the video after post-processing, the quality and details of the image have greatly improved.

3. References

Abdullah-Al-Mamun, M., Tyagi, V. and Zhao, H. (2021) A New Full-Reference Image Quality Metric for Motion Blur Profile Characterization. *IEEE Access*, 9, pp.156361–156371. Available from: <https://doi.org/10.1109/access.2021.3130177>.

Ansari, Z. (2025) *Understanding Image Noise: Exploring Different Kinds of Noises in Image Processing*. [online]. Medium. Available from: <https://medium.com/@zar_373/understanding-image-noise-exploring-different-kinds-of-noises-in-image-processing-dcd26c6c9e5a> [Accessed 18 May 2025].

Cambridge in Colour. (n.d.) *Guide to Image Sharpening*. [online]. Available from: <https://www.cambridgeincolour.com/tutorials/image-sharpening.htm#google_vignette> [Accessed 18 May 2025].

Kim, K., Choi, J. and Lee, Y. (2020) Effectiveness of Non-Local Means Algorithm with an Industrial 3 MeV LINAC High-Energy X-ray System for Non-Destructive Testing. *Sensors*, [online] 20(9), pp.2634–2634. Available from: <https://doi.org/10.3390/s20092634>.

Pai, N. (2024) *Gray-World Assumption in Computer Vision - Nick Pai - Medium*. [online]. Medium. Available from: <https://medium.com/@weichenpai/gray-world-assumption-in-computer-vision-0a6612c1420a> [Accessed 17 May 2025].

Rosebrock, A. (2021) *OpenCV Histogram Equalization and Adaptive Histogram Equalization (CLAHE)*. [online]. PyImageSearch. Available from: <https://pyimagesearch.com/2021/02/01/opencv-histogram-equalization-and-adaptive-histogram-equalization-clahe/> [Accessed 19 May 2025].

Salvatore (2025) *images_ug*. [online]. Scribd. Available from: <https://www.scribd.com/document/502532563/images-ug> [Accessed 17 May 2025].

Shen, Y. and Ma, L. (2010) Detecting and Removing the Motion Blurring from Video Clips. *International Journal of Modern Education and Computer Science*, 2(1), pp.17–23. Available from: <https://doi.org/10.5815/ijmecs.2010.01.03>.

Singh, K.B., Mahendra, T.V., Kurmvanshi, R.S. and Rama Rao, C.V. (2017) Image enhancement with the application of local and global enhancement methods for dark images. *2017 International Conference on Innovations in Electronics, Signal Processing and Communication (IESC)*, pp.199–202. Available from: <https://doi.org/10.1109/iespc.2017.8071892>.

TechnoLynx Ltd (2025) *Feature Extraction and Image Processing for Computer Vision*. [online]. TechnoLynx. Available from: <https://www.technolynx.com/post/feature-extraction-and-image-processing-for-computer-vision> [Accessed 18 May 2025].