UNIVERSITY OF WESTMINSTER

Road Sign or Text Detection

Objective: Detect text or road signs from natural scenes

Mahima Dharmasena

20520720

09/01/2025

Abstract

This report presents the development of an interactive image-processing application implemented in MATLAB. The application features a graphical user interface (GUI) designed to streamline the analysis and preprocessing of images, particularly for detecting text or road signs in natural scenes. Its core functionalities include grayscale conversion with customisable RGB weights, RMS contrast calculation, histogram visualisation, binarisation using adjustable thresholds, and morphological filtering with selectable operations (dilation or erosion). The user-friendly GUI offers a responsive and visually appealing interface, ensuring efficient and intuitive interactions. This project highlights the integration of advanced image processing algorithms into a cohesive and interactive tool, demonstrating its utility for real-world applications such as autonomous driving systems, traffic monitoring, and accessibility tools for visually impaired individuals.

Table of Contents

Abstract *1*

Table of Figures *3*

1. Introduction *4*

1.1 Objective **4**

1.2 Relevance **5**

1.2.1 Enhancing Road Safety 5

1.2.2 Improving Autonomous Driving Systems 5

1.2.3 Supporting Accessibility Tools 5

1.2.4 Alignment with Learning Outcomes 6

2. Methodology *7*

2.1 Application Design **7**

2.2 Features and Implementation **8**

2.2.1 Image Upload 8

2.2.2 Grayscale Conversion 9

2.2.3 Contrast Calculation 10

2.2.4 Test All Weights 11

2.2.5 Histogram Visualization 12

2.2.6 Binarization 13

2.2.7 Morphological Filtering 14

3. Results *15*

3.1 GUI Usability **15**

3.2 Image Processing Performance **16**

4. Ethical and Societal Impact *17*

4.1 Environmental Impact **17**

4.2 Societal Benefits **17**

4.2.1 Improving Road Safety 17

4.2.2 Supporting Accessibility Tools 18

4.3 Ethical Considerations **18**

4.3.1 Privacy 18

4.3.2 Bias 19

5. Conclusion *20*

6. References *21*

7. Appendices *22*

Appendix A: GUI Screenshots **22**

Appendix B: Full MATLAB Code **25**

Table of Figures

Figure 1: Original Image Uploaded to the GUI 8
Figure 2: Grayscale Image Displayed in the GUI 9
Figure 3: RMS Contrast Calculation Result 10
Figure 4: Best RGB Weights for Maximum Contrast 11
Figure 5: Histogram of the Grayscale Image 12
Figure 6: Binarized Image After Applying Threshold 13
Figure 7: Morphologically Filtered Image 14

1. Introduction

1.1 Objective

This project aims to design and implement a comprehensive MATLAB-based graphical user interface (GUI) that enables users to efficiently process and analyse images, focusing on detecting road signs or text in natural scenes. The system integrates various image processing techniques to provide a seamless image acquisition, analysis, and feature extraction workflow.

This project aims to:

1. **Facilitate Road Sign and Text Detection**: The application is designed to preprocess and enhance image features, enabling the accurate detection of road signs or textual information in diverse environments. These capabilities are critical for applications such as autonomous vehicles, traffic monitoring systems, and accessibility tools.

2. **Simplify Image Processing Tasks**: By providing an intuitive, user-friendly GUI, the application allows users to perform complex image processing tasks like grayscale conversion, contrast calculation, histogram visualisation, binarisation, and morphological filtering without requiring extensive technical expertise.

3. **Integrate Key Image Processing Techniques**:

    • **Grayscale Conversion**: Allows customisable RGB weight combinations for optimal feature extraction.

    • **Contrast Analysis**: Uses RMS contrast to evaluate the visibility of image features.

    • **Binarization and Morphological Filtering**: Segments and enhances text or object features, reducing noise and improving clarity.

    • **Histogram Analysis**: Visualizes intensity distribution to aid in decision-making.

4. **Enhance Usability and Accessibility**: The application provides real-time, inline visualisations of original and processed images, ensuring users can analyse results effectively and adjust parameters dynamically.

5. **Support Practical Applications**: By automating tedious tasks such as preprocessing and feature extraction, the application is tailored for real-world applications, such as improving road safety through better sign detection for autonomous vehicles or enhancing text detection for visually impaired users.

In summary, this project seeks to bridge the gap between theoretical image processing techniques and their practical applications by developing a versatile, accessible, and impactful tool that can be used in various scenarios.

1.2 Relevance

This application is significant in addressing real-world challenges and contributing to various domains such as road safety, autonomous driving, and accessibility. Its practical use cases demonstrate the impact of integrating advanced image processing techniques into intelligent systems. The application's design and implementation align with the learning outcomes of this project, making it an essential tool for both academic and practical contexts.

1.2.1 Enhancing Road Safety

One of this application's most critical use cases is improving road safety. Accurately detecting and analysing road signs and text in natural scenes is vital for advanced driver assistance systems (ADAS) and autonomous vehicles. Road sign recognition is a key component of autonomous driving, as it helps vehicles make decisions like adjusting speed, stopping at intersections, or yielding to pedestrians. By preprocessing images to enhance features and reduce noise, this application ensures the clarity and accuracy of road sign detection, even in challenging conditions such as poor lighting or adverse weather. This contributes directly to reducing traffic accidents and enhancing the safety of road users.

1.2.2 Improving Autonomous Driving Systems

Autonomous vehicles rely heavily on real-time data from various sensors, including cameras, to navigate safely and efficiently. This application supports the development of intelligent sensor systems by providing robust preprocessing tools that improve the accuracy of object and text detection. For example, the application prepares images for further processing in machine learning or computer vision models using grayscale conversion with customisable RGB weights, contrast analysis, and binarisation. This enhances autonomous systems' overall performance and reliability, making them more effective in diverse environments.

1.2.3 Supporting Accessibility Tools

The application is also highly relevant in developing accessibility tools for visually impaired individuals. Accurately detecting and preprocessing textual information can be integrated into text-to-speech systems or wearable assistive devices. These tools enable visually impaired users to interpret their surroundings more effectively, such as reading road signs, menus, or public notices. This enhances their independence and quality of life, promoting inclusivity and accessibility.

1.2.4 Alignment with Learning Outcomes

The application aligns directly with the learning outcomes (LOs) specified for this project:

1. **Using Image Acquisition Tools and Image Processing Algorithms**

   The application demonstrates the integration of image acquisition and processing techniques into a cohesive system. By leveraging MATLAB's Image Processing Toolbox, it showcases the practical use of algorithms for grayscale conversion, contrast calculation, histogram visualisation, binarisation, and morphological filtering.

2. **Understanding the Societal and Ethical Implications of Intelligent Sensor Solutions**

   The application contributes to societal needs, such as road safety and accessibility, while addressing ethical concerns like data privacy and bias. The inclusion of privacy-compliant image processing methods and the emphasis on diverse datasets highlight its adherence to ethical practices.

3. **Presenting Technical Designs and Results Effectively**

   The GUI-based design simplifies technical tasks for users, allowing real-time visualisation of processed images and results. The application also provides structured outputs, such as contrast values and histograms, essential for presenting technical results clearly and concisely.

In conclusion, this application is highly relevant to addressing contemporary challenges in road safety, autonomous driving, and accessibility. Its alignment with the project learning outcomes demonstrates its importance as both an academic exercise and a practical solution. By providing robust preprocessing capabilities, it paves the way for intelligent systems that are safer, more inclusive, and ethically responsible.

2. Methodology

2.1 Application Design

The GUI is designed to provide a transparent, responsive, and intuitive layout with the following features:

1. **Image Upload and Replace**: A button allows users to upload or replace the currently processed image.

2. **Dynamic Processing Options**: Dropdowns and sliders for user-selected parameters like thresholding and morphological operations.

3. **Inline Visualization**: Original and processed images are displayed side by side for easy comparison.

4. **Custom RGB Weights**: Allows testing custom RGB weights for grayscale conversion.

5. **Advanced Image Processing**: Includes operations like contrast calculation, histogram visualisation, binarisation, and morphological filtering.

2.2 Features and Implementation

2.2.1 Image Upload

• **Description**: Users can upload or replace an image using the “Upload/Change Image” button.

• **How It Works**: The selected image is displayed in the left panel of the GUI (original image area).

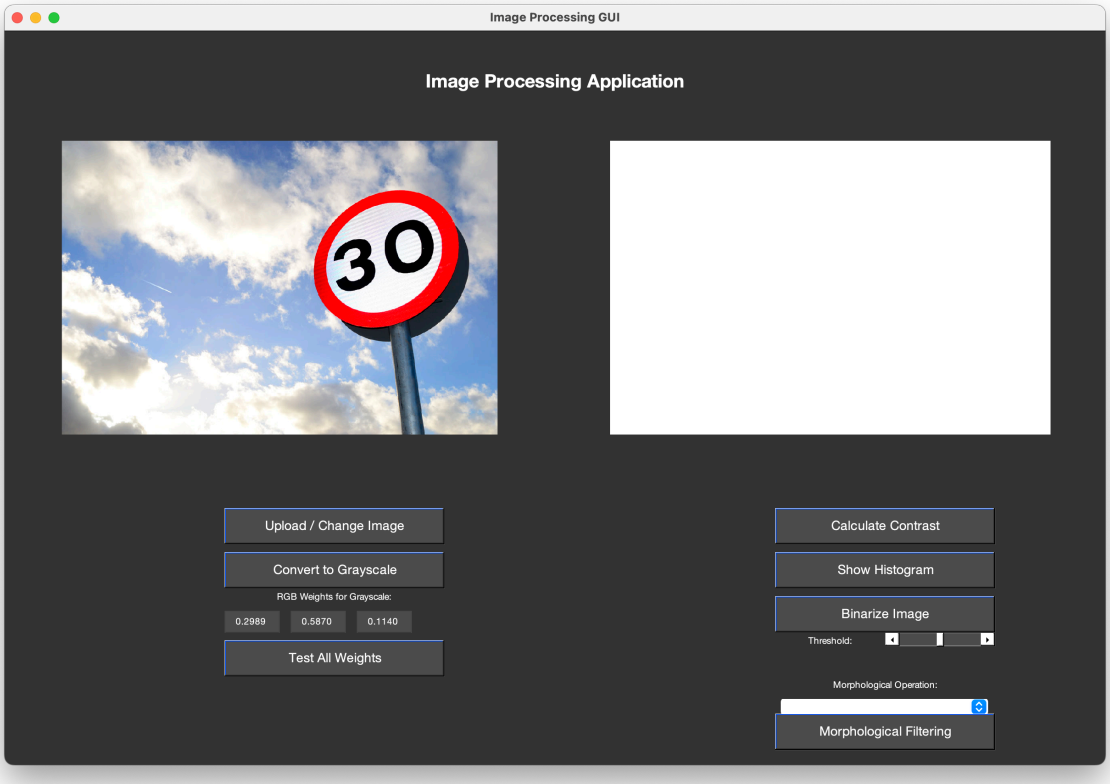


Figure 1: Original Image Uploaded to the GUI

2.2.2 Grayscale Conversion

• **Description**: Converts the uploaded image to grayscale based on user-defined RGB weights.

• **Implementation**: The RGB weights can be manually entered, and the grayscale image is displayed in the processed image panel.

• **Formula**: *\text{Grayscale Image} = R \cdot w_r + G \cdot w_g + B \cdot w_b*

Where *w_r, w_g, w_b* are user-defined weights that sum to 1.

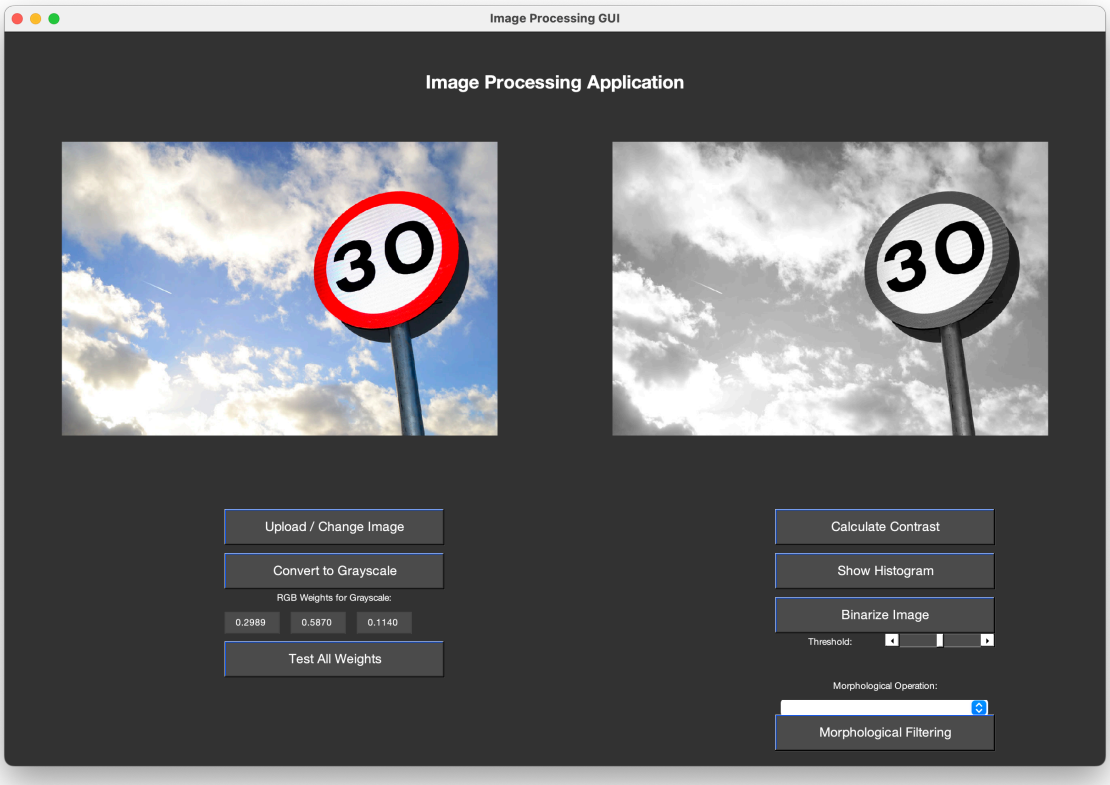


Figure 2: Grayscale Image Displayed in the GUI

2.2.3 Contrast Calculation

• **Description**: Computes the RMS contrast of the grayscale image and displays the result in a dialogue box.

• **Formula**: $\text{RMS Contrast} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (I_i - \bar{I})^2}$

Where I_i is the intensity of pixel i and $\bar{I}$ is the mean intensity.

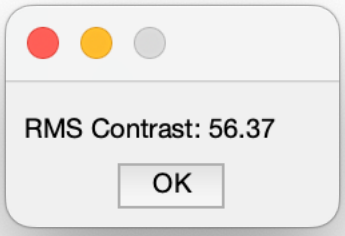


Figure 3: RMS Contrast Calculation Result

2.2.4 Test All Weights

• **Description**: Tests all 66 possible combinations of RGB weights and identifies the one that produces the highest contrast.

• **Output**: The best weight combination is displayed in a dialog box.

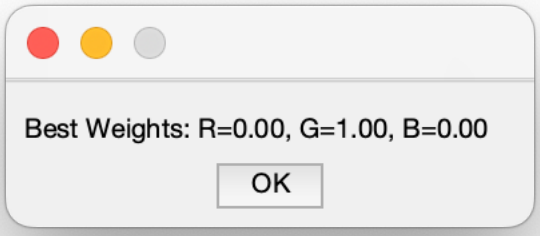


Figure 4: Best RGB Weights for Maximum Contrast

2.2.5 Histogram Visualization

• **Description**: Displays the intensity distribution of the grayscale image as a histogram in the processed image panel.

• **How It Works**: The histogram is generated dynamically based on the current grayscale image.

Figure 5: Histogram of the Grayscale Image

2.2.6 Binarization

• **Description**: Converts the grayscale image into a binary image using a threshold value adjustable via a slider.

• **Implementation**: Users can manually adjust the threshold slider to fine-tune the binarisation.

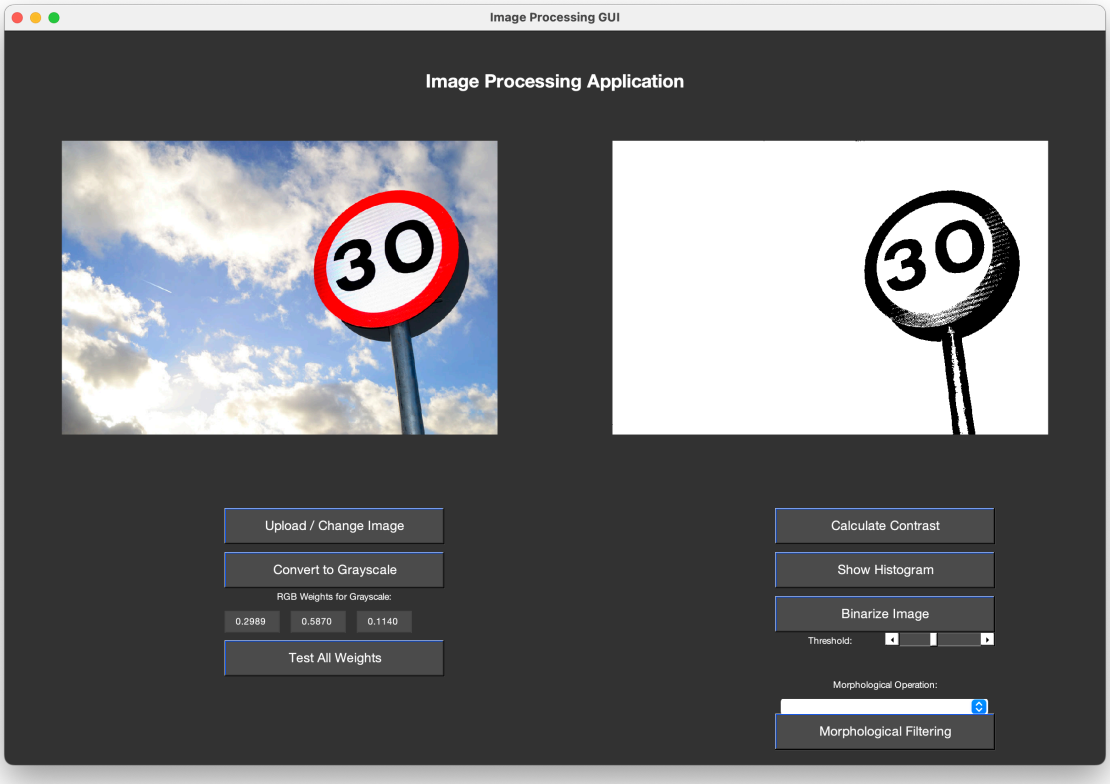


Figure 6: Binarized Image After Applying Threshold

2.2.7 Morphological Filtering

• **Description**: Applies morphological operations (dilation or erosion) to the binarised image.

• **Implementation**: The user selects the desired operation (dilation or erosion) from a dropdown menu. A disk-shaped structuring element is used.

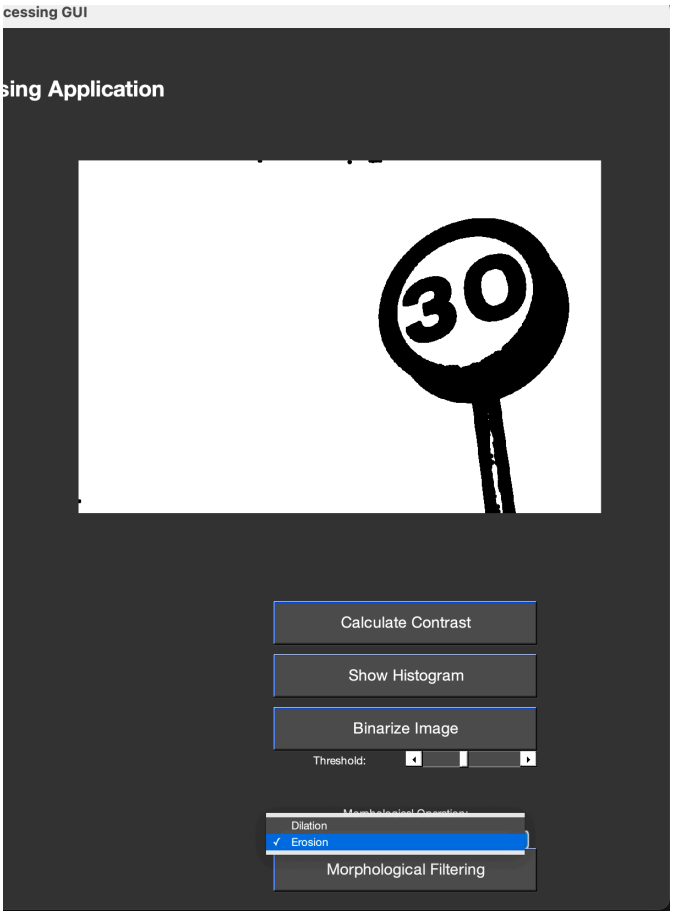


Figure 7: Morphologically Filtered Image

3. Results

3.1 GUI Usability

The graphical user interface (GUI) is designed to ensure a seamless and intuitive user experience. The layout includes clearly labelled buttons, sliders, and dropdown menus, which are systematically organised for easy navigation. Each operation, such as uploading an image, converting to grayscale, or applying a morphological filter, is executed with minimal effort. The GUI provides **real-time inline visual feedback**, allowing users to view the results of their actions directly in the **Processed Image** panel. This approach eliminates the need for external windows or pop-ups and ensures the user remains focused on the task. The application's colour scheme, responsive design, and consistent aesthetics enhance usability and make it accessible for users of varying technical expertise.

3.2 Image Processing Performance

The application successfully implements key image processing functionalities with robust performance across all operations. Detailed results for each feature are outlined below:

- **Grayscale Conversion**:

  The application allows users to convert colourful images into grayscale using customisable RGB weight combinations. The grayscale images are clear and suitable for further analysis. The flexibility to manually adjust weights or test all combinations ensures high adaptability for diverse image datasets.

- **Contrast Calculation**:

  The tool accurately computes grayscale images' **Root Mean Square (RMS) contrast**. This functionality provides insight into the image's visibility and feature clarity, displaying results in an easy-to-read dialogue box.

- **Histogram Visualization**:

  The application generates precise histograms depicting grayscale images' intensity distribution. These histograms are displayed inline within the GUI, providing a clear understanding of the image's brightness and contrast characteristics.

- **Binarization**:

  The **threshold slider** lets users adjust the binarisation level dynamically, ensuring precise grayscale image segmentation. This feature is handy for separating objects (e.g., text or road signs) from the background, even in challenging conditions like low contrast or uneven lighting.

- **Morphological Operations**:

  The application supports **dilation** and **erosion**, allowing users to enhance or refine the binarised image's features. The operations effectively clean noise and improve feature quality using a disk-shaped structuring element. For example, dilation can enhance text regions, while erosion helps reduce minor noise artefacts.

The results demonstrate that the application performs all key image-processing tasks effectively and efficiently. Its intuitive GUI and robust processing capabilities make it a versatile tool for analysing and enhancing images, particularly for road sign detection and text recognition applications. The combination of flexibility, accuracy, and usability ensures that the application meets academic and practical requirements.

4. Ethical and Societal Impact

4.1 Environmental Impact

The application significantly reduces manual labour and energy consumption by automating the process of analysing road signs and text in images. Traditionally, such tasks would require manual inspection, which is time-consuming, labour-intensive, and prone to human error. For instance, in traffic monitoring or infrastructure maintenance scenarios, workers would need to physically inspect roads and analyse signs, often using vehicles and equipment that consume fuel and produce emissions. By automating this analysis, the application minimises the need for such resources, leading to lower energy usage and reduced carbon emissions. Furthermore, the tool's ability to preprocess and enhance image data for machine learning or computer vision systems allows for seamless integration into larger-scale autonomous systems, such as intelligent traffic management and autonomous vehicles. These systems further enhance sustainability by optimising resource usage and reducing inefficiencies. In summary, the application is critical in transitioning to eco-friendly and sustainable solutions for image analysis and decision-making processes.

4.2 Societal Benefits

4.2.1 Improving Road Safety

The application enhances road safety by enabling accurate and efficient detection of road signs, which is critical for the development and functionality of autonomous vehicles. Autonomous vehicles rely on real-time identification and interpretation of road signs to make informed decisions, such as adjusting speed limits, stopping at intersections, or yielding to pedestrians. The preprocessing capabilities of this application, such as grayscale conversion, contrast enhancement, and morphological filtering, ensure that the features of road signs are extracted clearly and accurately, even under challenging conditions like poor lighting, motion blur, or obstructions. By improving the reliability of road sign detection systems, the application contributes to the safe operation of autonomous vehicles, reducing the likelihood of accidents caused by missed or misinterpreted signs. This ultimately creates safer roads for both drivers and pedestrians.

4.2.2 Supporting Accessibility Tools

The application also supports the development of accessibility tools, particularly for visually impaired individuals. Text-to-speech systems, which convert visual text into audible speech, rely on accurate text detection and preprocessing to function effectively. The application's ability to preprocess images by binarising text regions, enhancing features, and reducing noise ensures that text can be accurately extracted. These accessibility tools can be integrated into wearable devices, mobile applications, or assistive technologies, enabling visually impaired users to read signs, menus, and other text in their surroundings. By empowering individuals with disabilities to navigate and interact with their environment more independently, the application promotes inclusivity and improves quality of life.

In summary, the application addresses critical societal needs by improving the safety and efficiency of autonomous vehicles and empowering visually impaired individuals through advanced text recognition capabilities.

4.3 Ethical Considerations

The application raises important ethical considerations that must be addressed to ensure responsible and fair use of the technology. Two key areas are privacy and bias, which have significant implications for societal trust and the system's effectiveness.

4.3.1 Privacy

When processing images, especially those captured in public spaces, there is a potential risk of handling personal or sensitive information, such as faces, license plates, or private property details. Without proper safeguards, this could lead to individual privacy rights breaches or personal data misuse. To address this, strict data protection policies must be implemented. For instance, anonymisation techniques, such as blurring or masking identifiable features, can be applied to ensure that individuals cannot be identified in processed images. Additionally, all data the application uses should comply with regulations like the **General Data Protection Regulation (GDPR)** or other relevant local privacy laws. Furthermore, secure storage and handling of images should be prioritised to prevent unauthorised access, and users must be informed of how their data is processed and used. The risk of ethical violations can be minimised by embedding privacy protections into the application's design and usage policies.

4.3.2 Bias

Bias in the detection process is another critical ethical issue. If the dataset used to train or test the application is not diverse, the system may perform poorly on images that differ from the conditions represented in the dataset. For example, suppose the dataset predominantly contains road signs from urban environments in developed countries. In that case, the application may struggle to accurately detect signs in rural areas or developing regions, where road signs vary in design, language, or material. This could lead to unequal performance, limiting the effectiveness of the application in specific contexts and perpetuating systemic inequalities. To prevent this, diverse datasets that include variations in geography, lighting, weather conditions, and cultural contexts must be used. Periodic evaluations of the application's performance across different scenarios should be conducted to identify and address any biases. Additionally, transparency about the dataset and its limitations is crucial to ensure accountability.

In conclusion, ethical considerations such as privacy and bias are essential to the responsible development and deployment of the application. By safeguarding personal data and ensuring fairness in the detection process, the application can achieve its full potential while maintaining public trust and promoting ethical practices in technology.

5. Conclusion

The interactive image processing tool successfully achieves the objectives set out for this project by integrating essential image processing techniques into a cohesive and accessible application. Through its intuitive and user-friendly GUI, the application enables users to preprocess and analyse images effectively, offering functionalities such as grayscale conversion, contrast calculation, histogram visualisation, binarisation, and morphological filtering. These features support practical applications such as road sign detection for autonomous vehicles and text recognition for accessibility tools. The project also emphasises the ethical considerations of data privacy and bias, ensuring responsible development and deployment. The tool effectively bridges the gap between theoretical image processing concepts and practical, impactful implementations. As a versatile and efficient solution, this application sets a foundation for future advancements in intelligent sensor systems and image-based decision-making processes.

6. References

• Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson Education.

• MathWorks. (n.d.). *Image Processing Toolbox User's Guide*. MATLAB Documentation. Retrieved from https://www.mathworks.com/help/images/

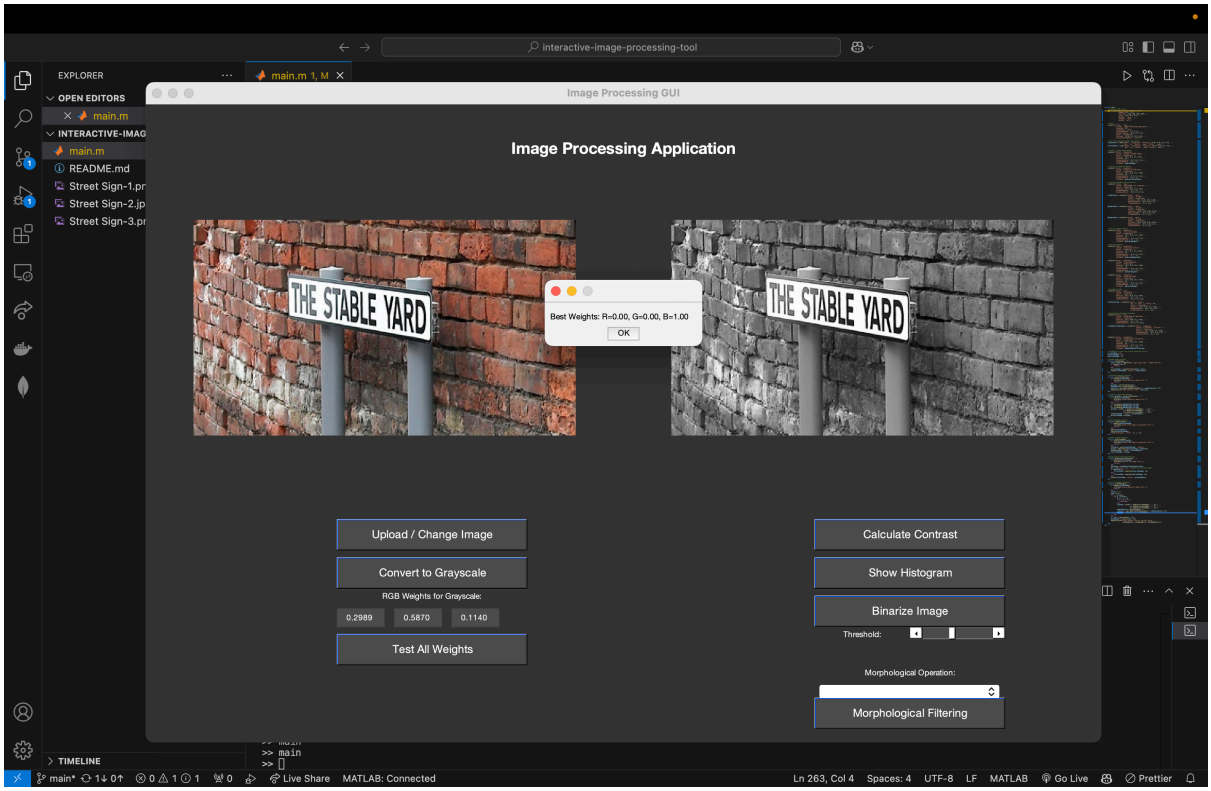
• Kaggle. (n.d.). *Road Sign Detection Dataset*. Retrieved from https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

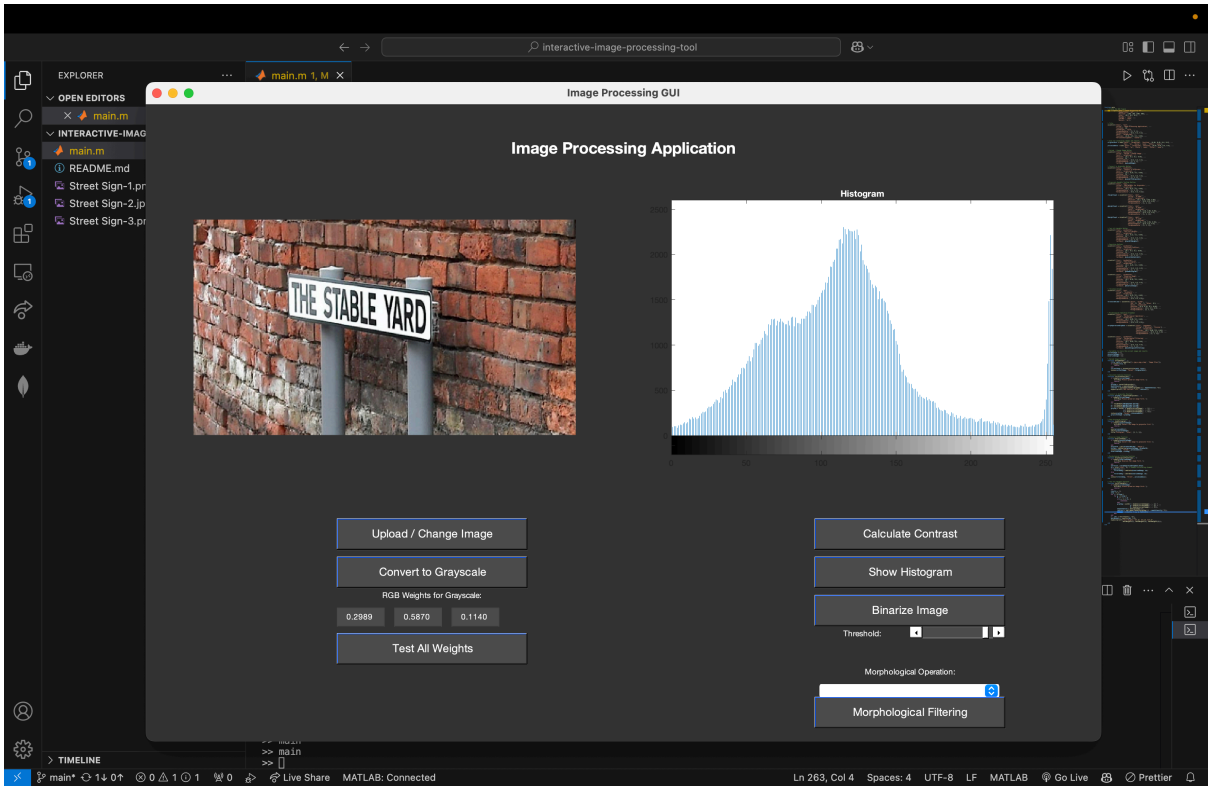
• Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

• European Union. (2018). *General Data Protection Regulation (GDPR)*. Retrieved from https://gdpr-info.eu

7. Appendices

Appendix A: GUI Screenshots





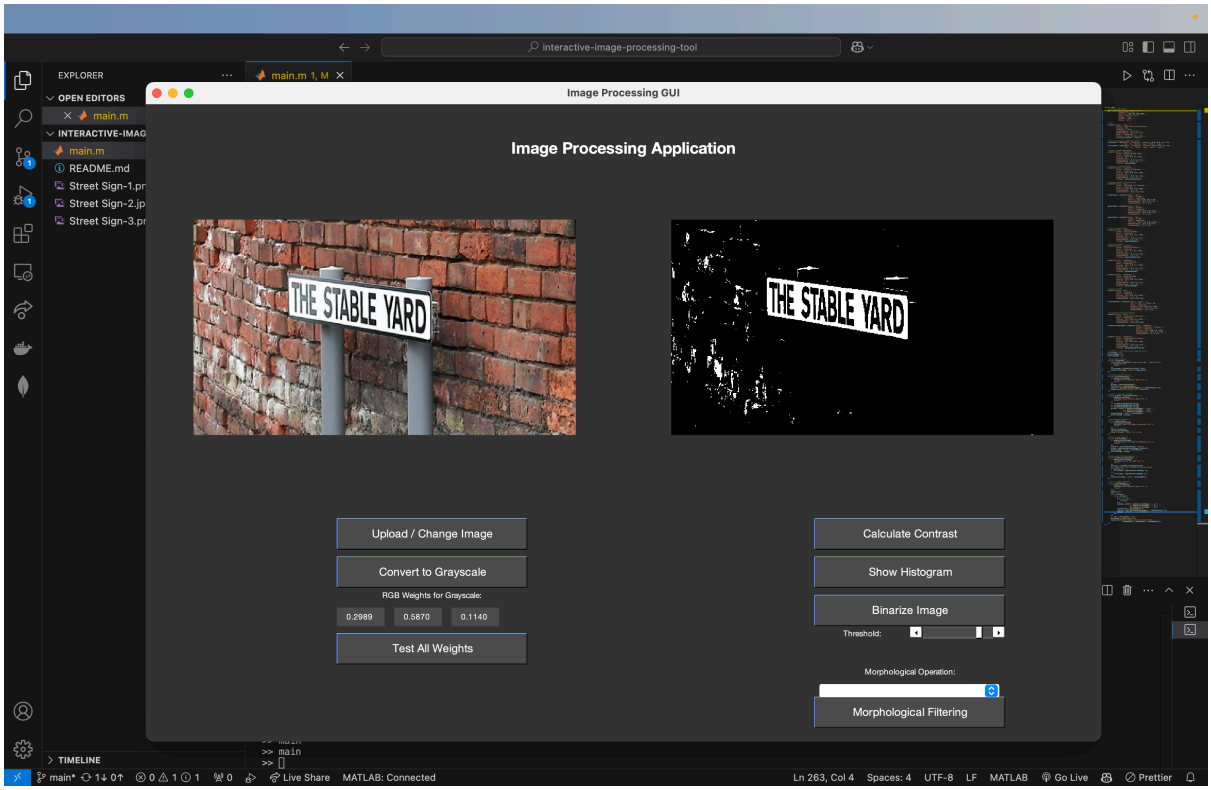


Image Processing GUI
Image Processing Application
THE STABLE YARD
THE STABLE YARD
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

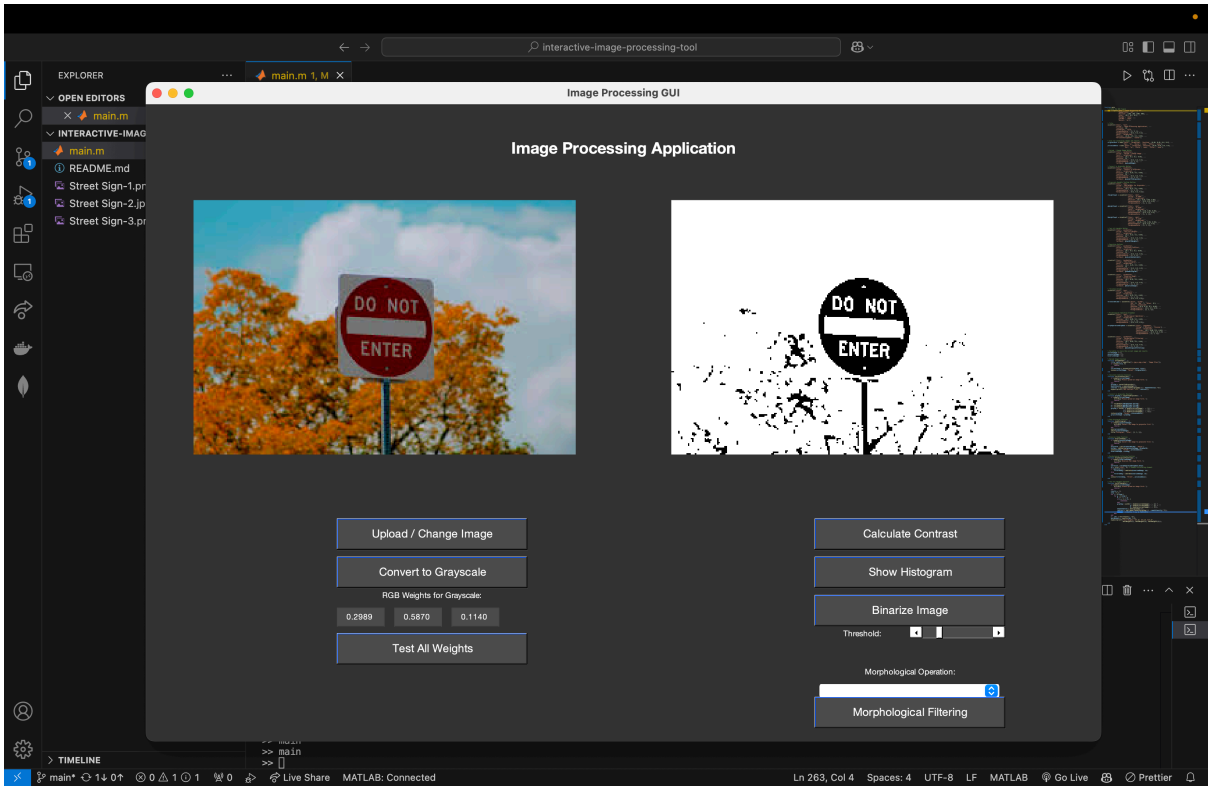


Image Processing GUI
Image Processing Application
DO NOT
ENTER
DO NOT
ENTER
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

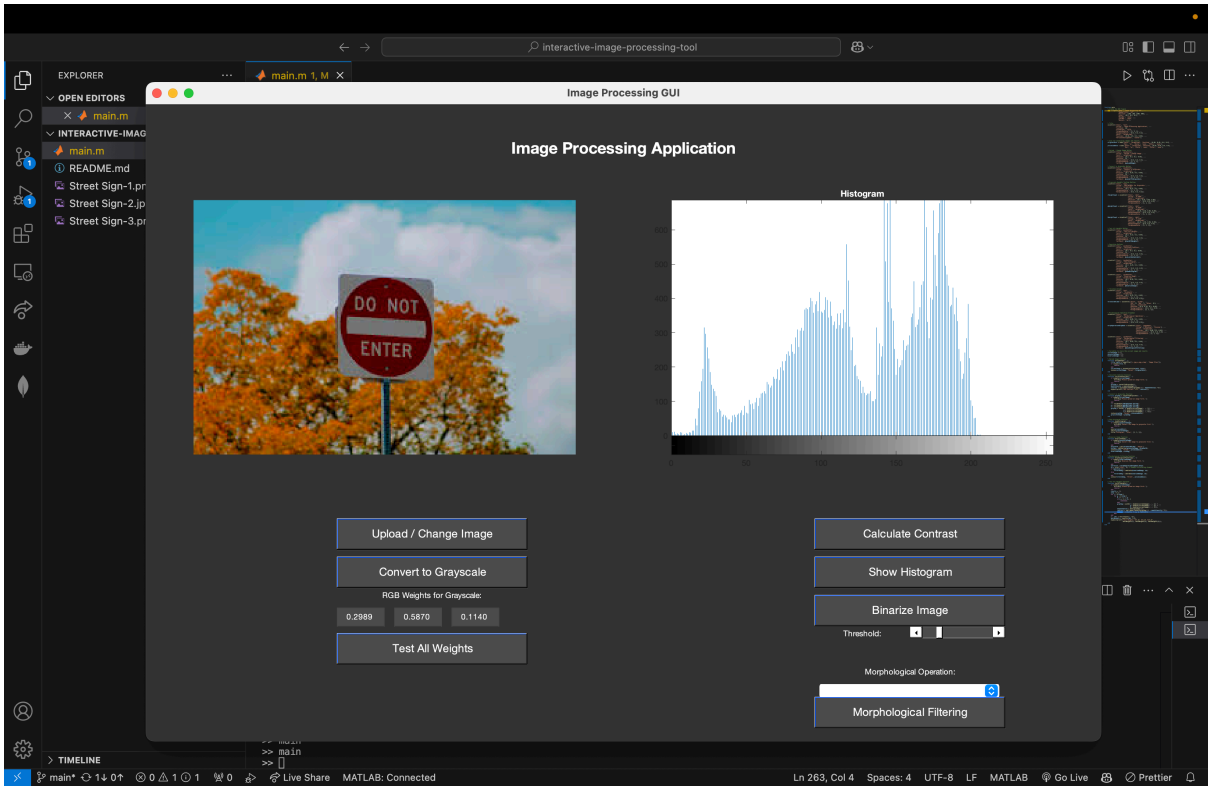


Image Processing GUI
Image Processing Application
Histogram
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

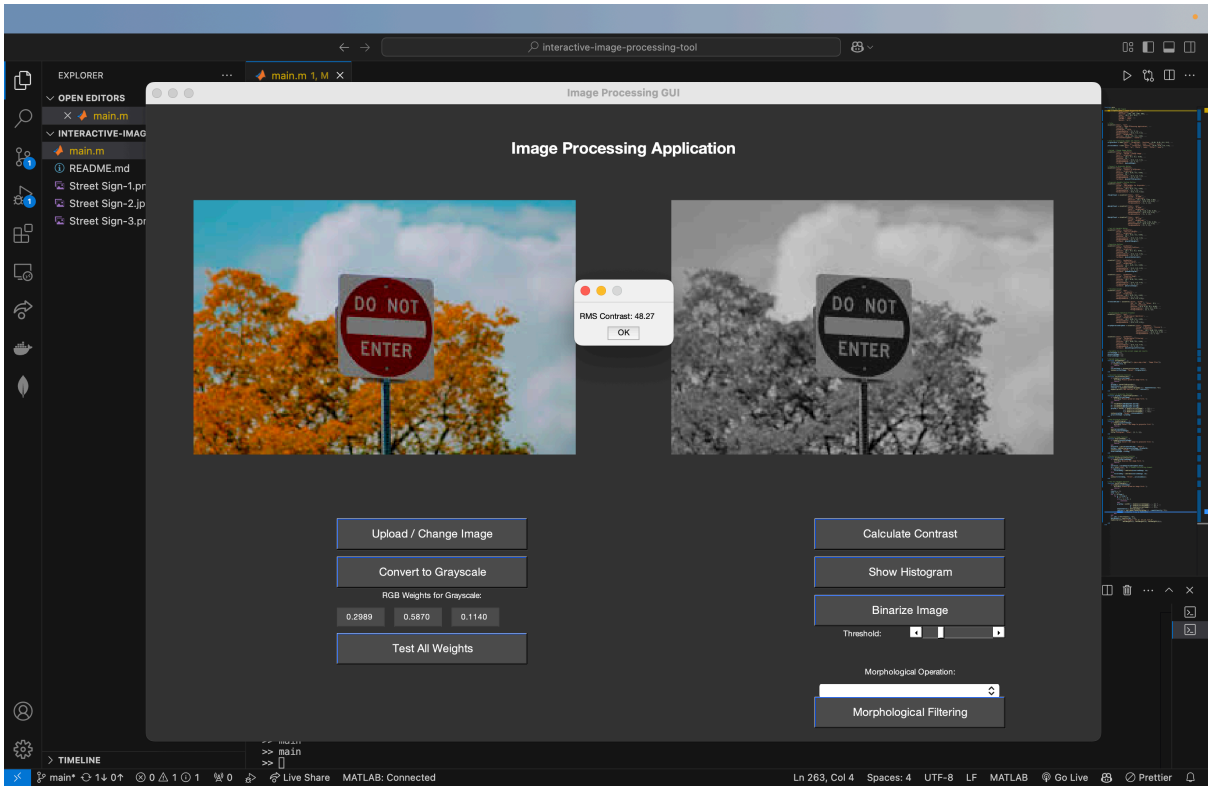


Image Processing GUI
Image Processing Application
RMS Contrast: 48.27
OK
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

Appendix B: Full MATLAB Code

```
function main
    % Create the GUI figure
    fig = figure('Name', 'Image Processing GUI', ...
                 'NumberTitle', 'off', ...
                 'Position', [100, 100, 1200, 800], ...
                 'Color', [0.2, 0.2, 0.2], ...
                 'MenuBar', 'none', ...
                 'ToolBar', 'none', ...
                 'Resize', 'on');

    % Title
    uicontrol('Style', 'text', ...
              'String', 'Image Processing Application', ...
              'FontSize', 20, ...
              'FontWeight', 'bold', ...
              'ForegroundColor', [1, 1, 1], ...
              'BackgroundColor', [0.2, 0.2, 0.2], ...
              'Units', 'normalized', ...
              'Position', [0.35, 0.9, 0.3, 0.05], ...
              'HorizontalAlignment', 'center');

    % Axes for displaying images and results
    originalAxis = axes('Units', 'normalized', 'Position', [0.05, 0.45, 0.4, 0.4],
...
                        'Box', 'on', 'XColor', 'none', 'YColor', 'none');
    processedAxis = axes('Units', 'normalized', 'Position', [0.55, 0.45, 0.4, 0.4],
...
                         'Box', 'on', 'XColor', 'none', 'YColor', 'none');

    % Upload / Change Image Button
    uicontrol('Style', 'pushbutton', ...
              'String', 'Upload / Change Image', ...
              'Units', 'normalized', ...
              'Position', [0.2, 0.3, 0.2, 0.05], ...
              'FontSize', 14, ...
              'BackgroundColor', [0.3, 0.3, 0.3], ...
              'ForegroundColor', [1, 1, 1], ...
              'Callback', @uploadImage);

    % Convert to Grayscale Button
    uicontrol('Style', 'pushbutton', ...
              'String', 'Convert to Grayscale', ...
              'Units', 'normalized', ...
              'Position', [0.2, 0.24, 0.2, 0.05], ...
              'FontSize', 14, ...
              'BackgroundColor', [0.3, 0.3, 0.3], ...
              'ForegroundColor', [1, 1, 1], ...
              'Callback', @convertToGrayscale);
```

```
    % Grayscale Weights Testing Section
    uicontrol('Style', 'text', ...
              'String', 'RGB Weights for Grayscale:', ...
              'Units', 'normalized', ...
              'Position', [0.2, 0.21, 0.2, 0.03], ...
              'ForegroundColor', [1, 1, 1], ...
              'BackgroundColor', [0.2, 0.2, 0.2]);

    rWeightInput = uicontrol('Style', 'edit', ...
                             'String', '0.2989', ...
                             'Units', 'normalized', ...
                             'Position', [0.2, 0.18, 0.05, 0.03], ...
                             'BackgroundColor', [0.3, 0.3, 0.3], ...
                             'ForegroundColor', [1, 1, 1]);

    gWeightInput = uicontrol('Style', 'edit', ...
                             'String', '0.5870', ...
                             'Units', 'normalized', ...
                             'Position', [0.26, 0.18, 0.05, 0.03], ...
                             'BackgroundColor', [0.3, 0.3, 0.3], ...
                             'ForegroundColor', [1, 1, 1]);

    bWeightInput = uicontrol('Style', 'edit', ...
                             'String', '0.1140', ...
                             'Units', 'normalized', ...
                             'Position', [0.32, 0.18, 0.05, 0.03], ...
                             'BackgroundColor', [0.3, 0.3, 0.3], ...
                             'ForegroundColor', [1, 1, 1]);

    % Test All Weights Button
    uicontrol('Style', 'pushbutton', ...
              'String', 'Test All Weights', ...
              'Units', 'normalized', ...
              'Position', [0.2, 0.12, 0.2, 0.05], ...
              'FontSize', 14, ...
              'BackgroundColor', [0.3, 0.3, 0.3], ...
              'ForegroundColor', [1, 1, 1], ...
              'Callback', @testAllWeights);

    % Operation Buttons
    uicontrol('Style', 'pushbutton', ...
              'String', 'Calculate Contrast', ...
              'Units', 'normalized', ...
              'Position', [0.7, 0.3, 0.2, 0.05], ...
              'FontSize', 14, ...
              'BackgroundColor', [0.3, 0.3, 0.3], ...
              'ForegroundColor', [1, 1, 1], ...
              'Callback', @calculateContrast);

    uicontrol('Style', 'pushbutton', ...
              'String', 'Show Histogram', ...
```

```
            'Units', 'normalized', ...
            'Position', [0.7, 0.24, 0.2, 0.05], ...
            'FontSize', 14, ...
            'BackgroundColor', [0.3, 0.3, 0.3], ...
            'ForegroundColor', [1, 1, 1], ...
            'Callback', @showHistogram);

    uicontrol('Style', 'pushbutton', ...
            'String', 'Binarize Image', ...
            'Units', 'normalized', ...
            'Position', [0.7, 0.18, 0.2, 0.05], ...
            'FontSize', 14, ...
            'BackgroundColor', [0.3, 0.3, 0.3], ...
            'ForegroundColor', [1, 1, 1], ...
            'Callback', @binarizeImage);

    % Threshold Slider
    uicontrol('Style', 'text', ...
            'String', 'Threshold:', ...
            'Units', 'normalized', ...
            'Position', [0.7, 0.15, 0.1, 0.03], ...
            'ForegroundColor', [1, 1, 1], ...
            'BackgroundColor', [0.2, 0.2, 0.2]);

    thresholdSlider = uicontrol('Style', 'slider', ...
                                'Min', 0, 'Max', 1, 'Value', 0.5, ...
                                'Units', 'normalized', ...
                                'Position', [0.8, 0.15, 0.1, 0.03], ...
                                'BackgroundColor', [0.3, 0.3, 0.3], ...
                                'ForegroundColor', [1, 1, 1]);

    % Morphological Operation Dropdown
    uicontrol('Style', 'text', ...
            'String', 'Morphological Operation:', ...
            'Units', 'normalized', ...
            'Position', [0.7, 0.09, 0.2, 0.03], ...
            'ForegroundColor', [1, 1, 1], ...
            'BackgroundColor', [0.2, 0.2, 0.2]);

    morphOperationDropdown = uicontrol('Style', 'popupmenu', ...
                                       'String', {'Dilation', 'Erosion'}, ...
                                       'Units', 'normalized', ...
                                       'Position', [0.7, 0.06, 0.2, 0.03], ...
                                       'BackgroundColor', [0.3, 0.3, 0.3], ...
                                       'ForegroundColor', [1, 1, 1]);

    uicontrol('Style', 'pushbutton', ...
            'String', 'Morphological Filtering', ...
            'Units', 'normalized', ...
            'Position', [0.7, 0.02, 0.2, 0.05], ...
            'FontSize', 14, ...
```

```
            'BackgroundColor', [0.3, 0.3, 0.3], ...
            'ForegroundColor', [1, 1, 1], ...
            'Callback', @morphologicalFiltering);

    % Variables to store the current image and results
    currentImage = [];
    grayscaleImage = [];
    binarizedImage = [];

    % Upload Image Function
    function uploadImage(~, ~)
        [file, path] = uigetfile({'*.jpg;*.png;*.bmp', 'Image Files'});
        if isequal(file, 0)
            return;
        end
        currentImage = imread(fullfile(path, file));
        imshow(currentImage, 'Parent', originalAxis);
    end

    % Calculate Contrast Function
    function calculateContrast(~, ~)
        if isempty(currentImage)
            errordlg('Please upload an image first.');
            return;
        end
        grayImg = convertToGrayscale();
        meanIntensity = mean(grayImg(:));
        contrast = sqrt(mean((double(grayImg(:)) - meanIntensity).^2));
        msgbox(sprintf('RMS Contrast: %.2f', contrast));
    end

    % Convert to Grayscale Function
    function grayImg = convertToGrayscale(~, ~)
        if isempty(currentImage)
            errordlg('Please upload an image first.');
            return;
        end
        r = str2double(rWeightInput.String);
        g = str2double(gWeightInput.String);
        b = str2double(bWeightInput.String);
        grayImg = uint8(r * double(currentImage(:, :, 1)) + ...
                        g * double(currentImage(:, :, 2)) + ...
                        b * double(currentImage(:, :, 3)));
        imshow(grayImg, 'Parent', processedAxis);
        grayscaleImage = grayImg;
    end

    % Show Histogram Function
    function showHistogram(~, ~)
        if isempty(grayscaleImage)
            errordlg('Convert the image to grayscale first.');
```

```
            return;
        end
        axes(processedAxis);
        imhist(grayscaleImage);
        title('Histogram', 'Color', [1, 1, 1]);
    end

    % Binarize Image Function
    function binarizeImage(~, ~)
        if isempty(grayscaleImage)
            errordlg('Convert the image to grayscale first.');
            return;
        end
        threshold = get(thresholdSlider, 'Value');
        binImg = imbinarize(grayscaleImage, threshold);
        imshow(binImg, 'Parent', processedAxis);
        binarizedImage = binImg;
    end

    % Morphological Filtering Function
    function morphologicalFiltering(~, ~)
        if isempty(binarizedImage)
            errordlg('Binarize the image first.');
            return;
        end
        operation = morphOperationDropdown.Value;
        se = strel('disk', 5); % Example structuring element
        if operation == 1
            filteredImg = imdilate(binarizedImage, se);
        else
            filteredImg = imerode(binarizedImage, se);
        end
        imshow(filteredImg, 'Parent', processedAxis);
    end

    % Test All Weights Function
    function testAllWeights(~, ~)
        if isempty(currentImage)
            errordlg('Please upload an image first.');
            return;
        end
        results = [];
        step = 0.1;
        for r = 0:step:1
            for g = 0:step:1
                b = 1 - r - g;
                if b < 0 || b > 1
                    continue;
                end
                grayImg = uint8(r * double(currentImage(:, :, 1)) + ...
                                g * double(currentImage(:, :, 2)) + ...
```

```
                                b * double(currentImage(:, :, 3)));
                meanIntensity = mean(grayImg(:));
                contrast = sqrt(mean((double(grayImg(:)) - meanIntensity).^2));
                results = [results; r, g, b, contrast];
            end
        end
        [~, idx] = max(results(:, 4));
        bestWeights = results(idx, 1:3);
        msgbox(sprintf('Best Weights: R=%.2f, G=%.2f, B=%.2f', ...
                       bestWeights(1), bestWeights(2), bestWeights(3)));
    end
end
```

https://github.com/mahiiyh/interactive-image-processing-tool