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 an image-processing application developed in MATLAB. The application features a graphical user interface (GUI) designed to analyse and preprocess images, particularly for text or road sign detection tasks. The implemented functionalities include grayscale conversion with customisable RGB weights, contrast calculation, histogram visualisation, binarisation with adjustable thresholds, and morphological filtering with selectable operations. The GUI is responsive, intuitive, and visually appealing, enabling efficient and user-friendly interaction. The application demonstrates the integration of image processing algorithms into a cohesive system and evaluates their practical use.

Table of Contents

Table of Figures

1. Introduction

1.1 Objective

This coursework aims to design and implement a MATLAB-based GUI for processing and analysing images, particularly for road signs or text detection in natural scenes. The system uses various image processing techniques to integrate image acquisition, analysis, and feature extraction.

1.2 Relevance

This application addresses practical use cases such as enhancing road safety, improving autonomous driving systems, and supporting accessibility tools. It aligns with the following learning outcomes:

- Using image acquisition tools and image processing algorithms.
- Understanding the societal and ethical implications of intelligent sensor solutions.
- Presenting technical designs and results effectively.

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).

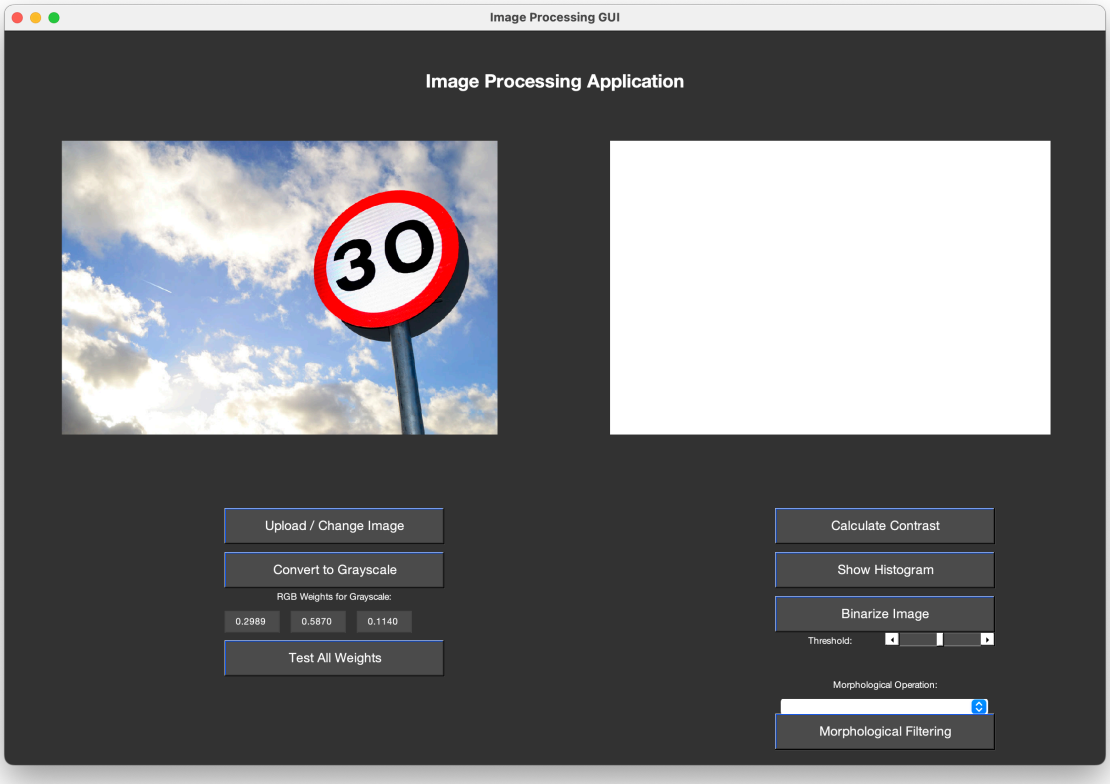


Figure 1: Original Image Uploaded to the GUI

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.

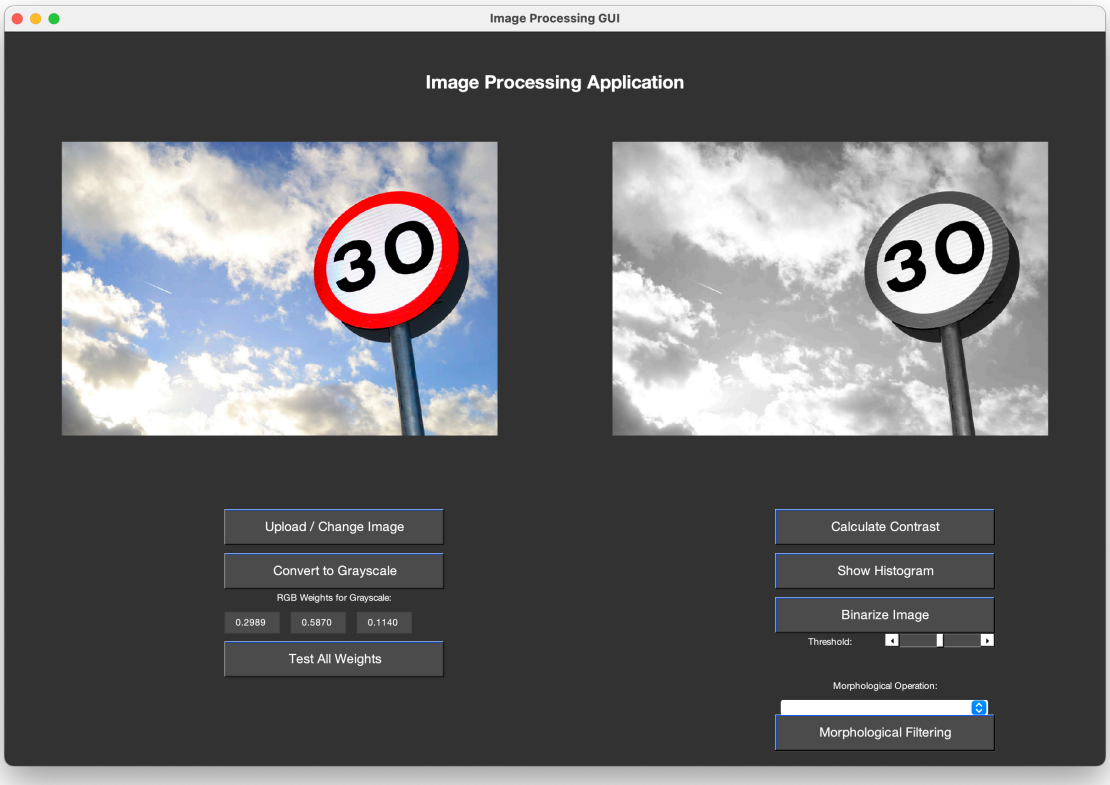


Figure 2: Grayscale Image Displayed in the GUI

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.

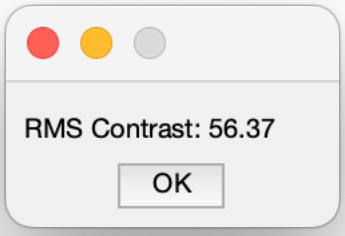


Figure 3: RMS Contrast Calculation Result

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.

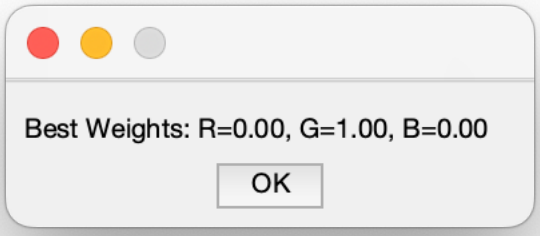


Figure 4: Best RGB Weights for Maximum Contrast

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.

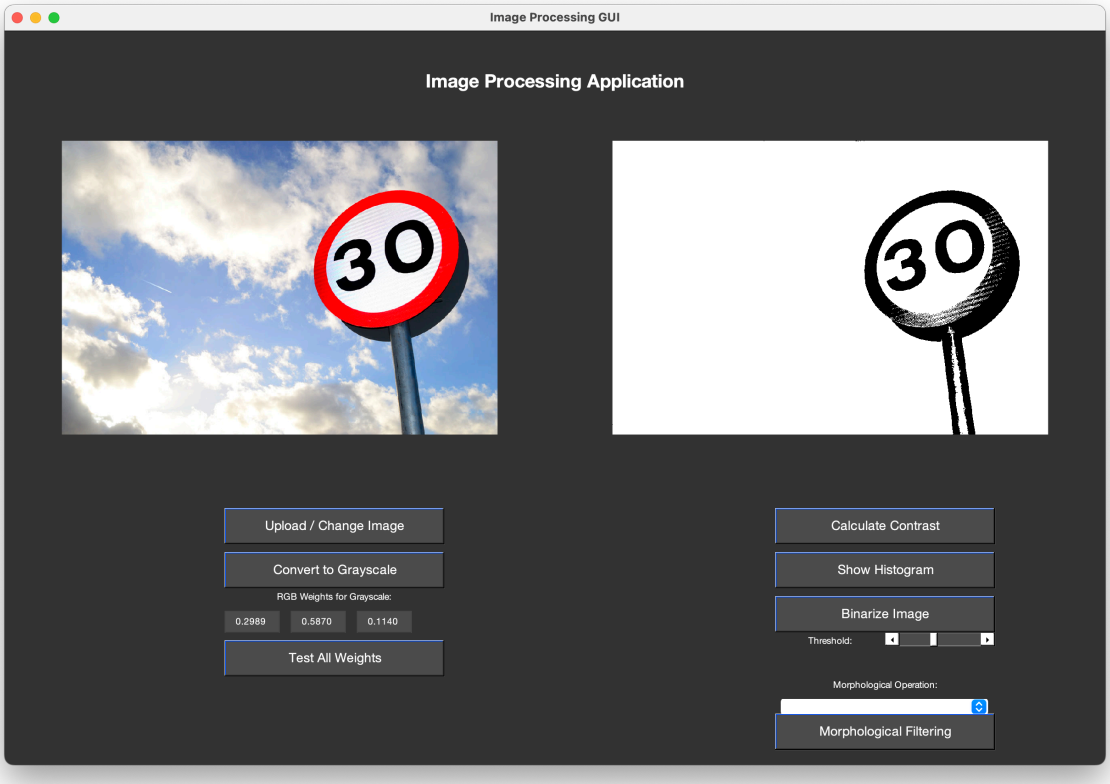


Figure 6: Binarized Image After Applying Threshold

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.

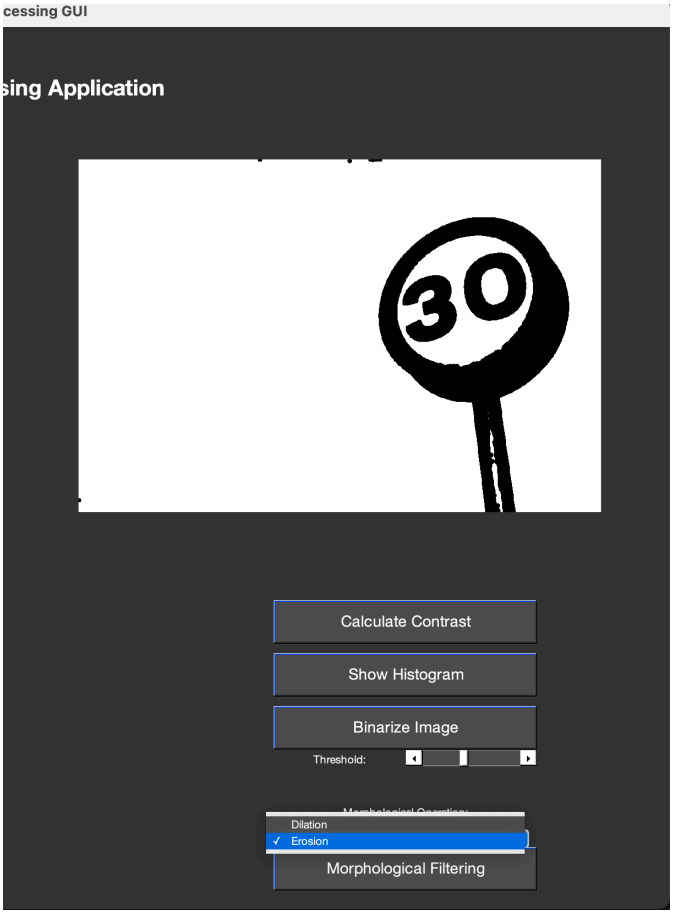


Figure 7: Morphologically Filtered Image

3. Results

3.1 GUI Usability

The user-friendly GUI has well-organized buttons, sliders, and dropdown menus. It provides inline visual feedback for all operations.

3.2 Image Processing Performance

- **Grayscale Conversion**: Successfully converts images with customisable weights.
- **Contrast Calculation**: Accurately computes RMS contrast.
- **Histogram Visualization**: Provides precise intensity distribution.
- **Binarization**: Threshold slider enables precise binary image generation.
- **Morphological Operations**: Effectively enhances features while reducing noise.

4. Ethical and Societal Impact

4.1 Environmental Impact

The application reduces manual labor and energy consumption for analysing road signs, contributing to sustainable solutions.

4.2 Societal Benefits

- Improves road safety by enhancing sign detection for autonomous vehicles.
- Supports accessibility tools like text-to-speech systems for visually impaired users.

4.3 Ethical Considerations

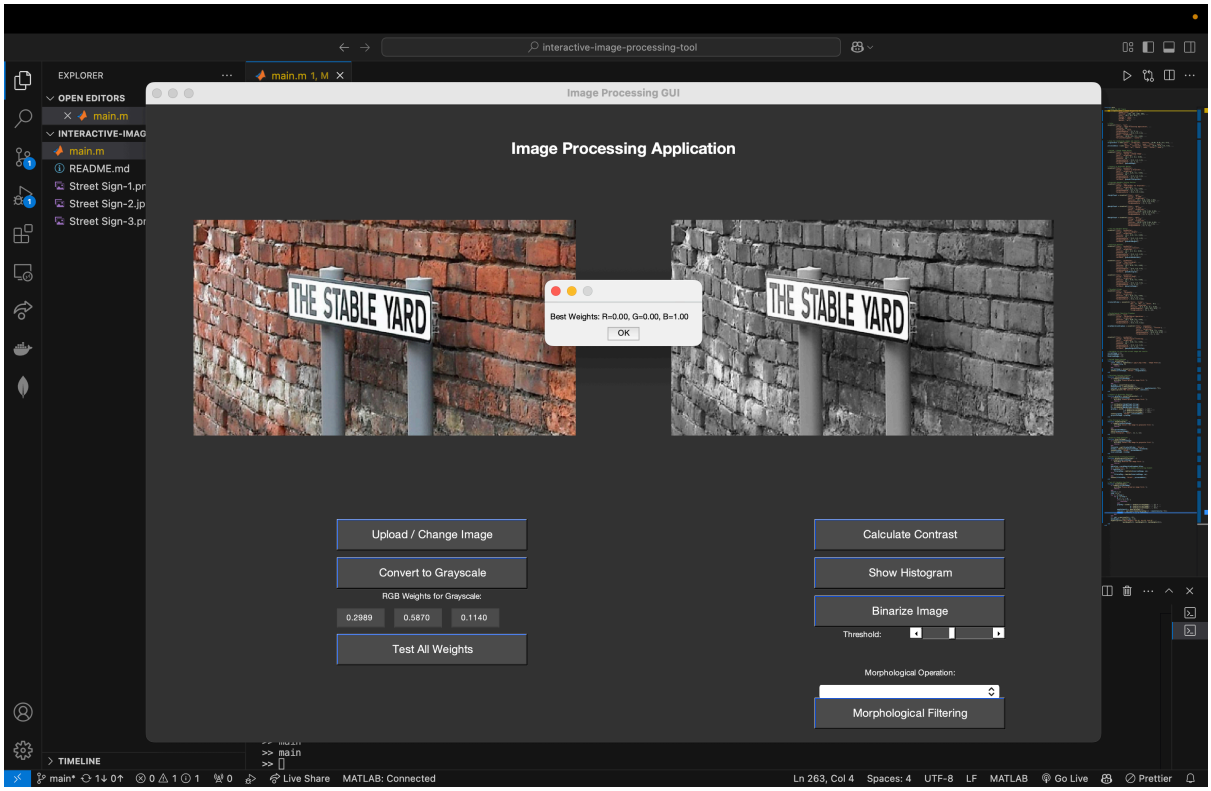
- **Privacy**: Ensuring data protection when processing personal or sensitive information.
- **Bias**: Using diverse datasets to prevent bias in the detection process.

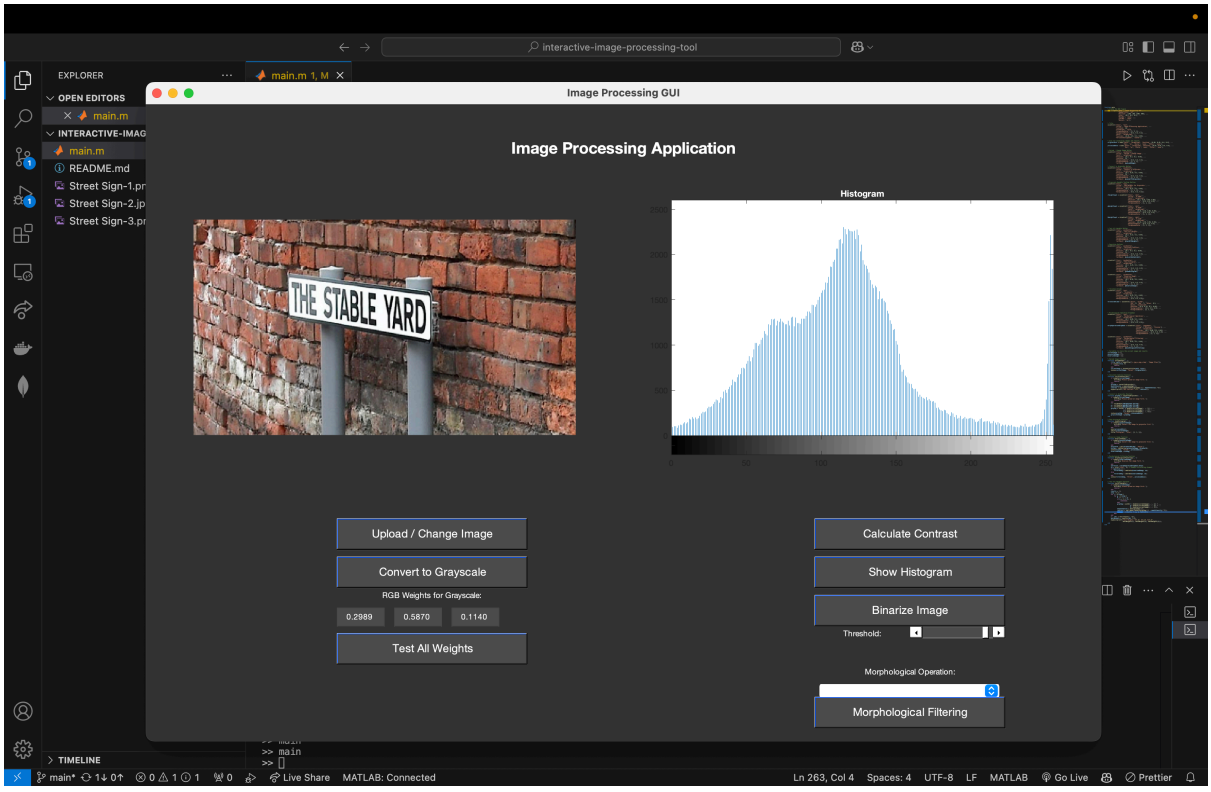
5. Conclusion

The Image Processing GUI fulfils the project requirements by integrating key image processing techniques into a cohesive and interactive tool. It successfully preprocesses and analyses images to detect features like text and road signs. The intuitive GUI and inline visualisation enhance its usability, making it a valuable tool for innovative sensor applications.

6. Appendices

Appendix A: GUI Screenshots





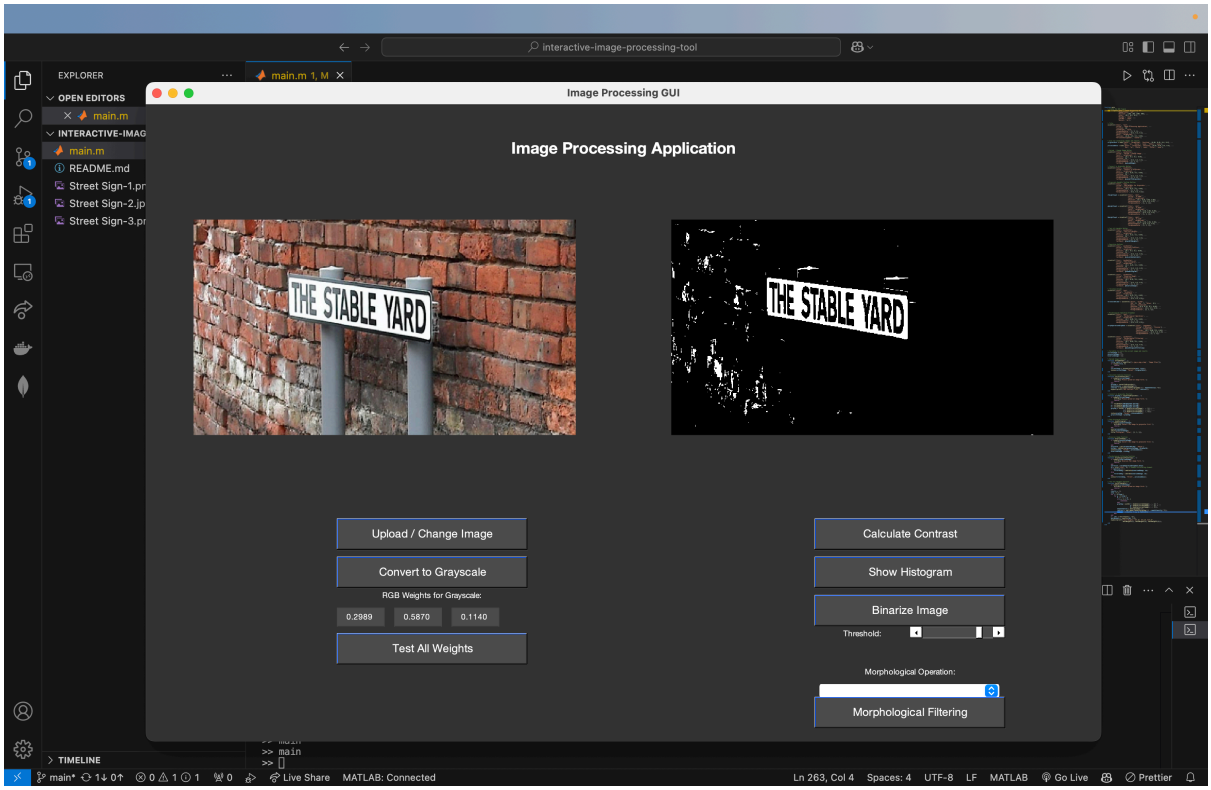


Image Processing GUI
Image Processing Application
THE STABLE YARD
THE STABLE YARD
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

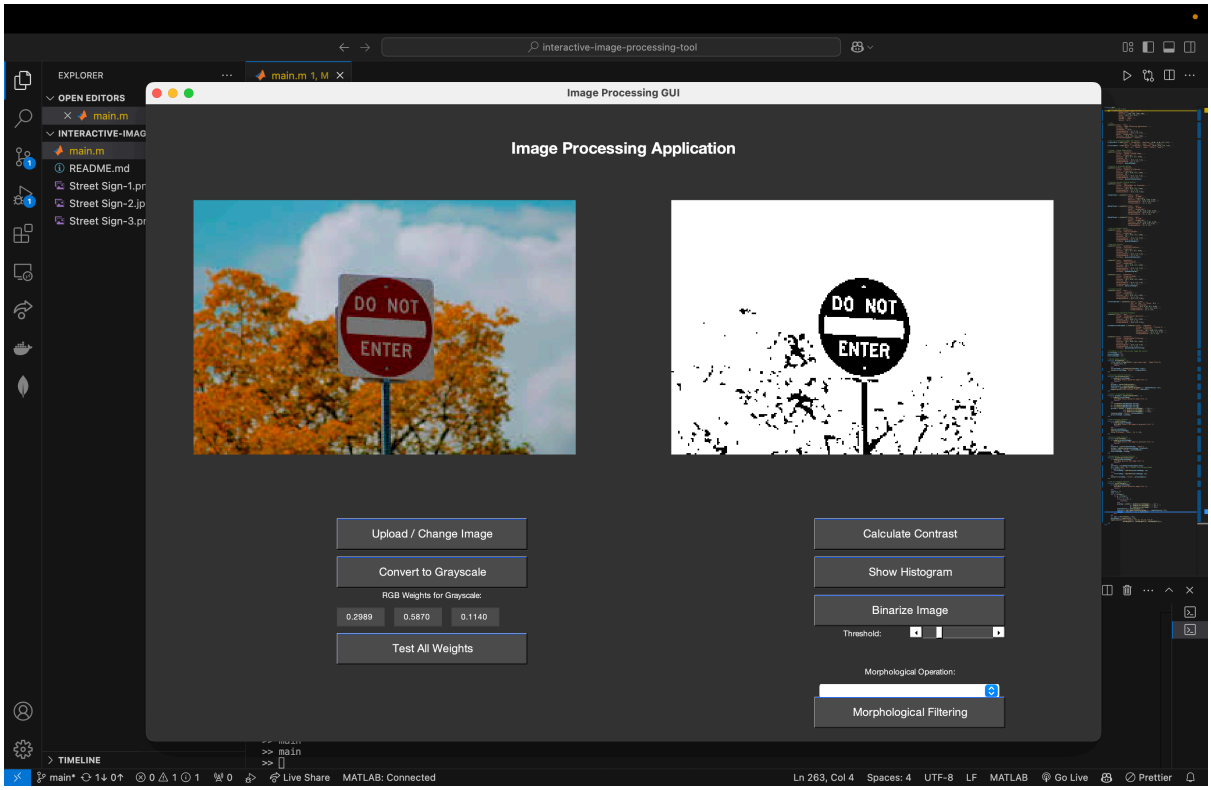


Image Processing GUI
Image Processing Application
DO NOT
ENTER
DO NOT
ENTER
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

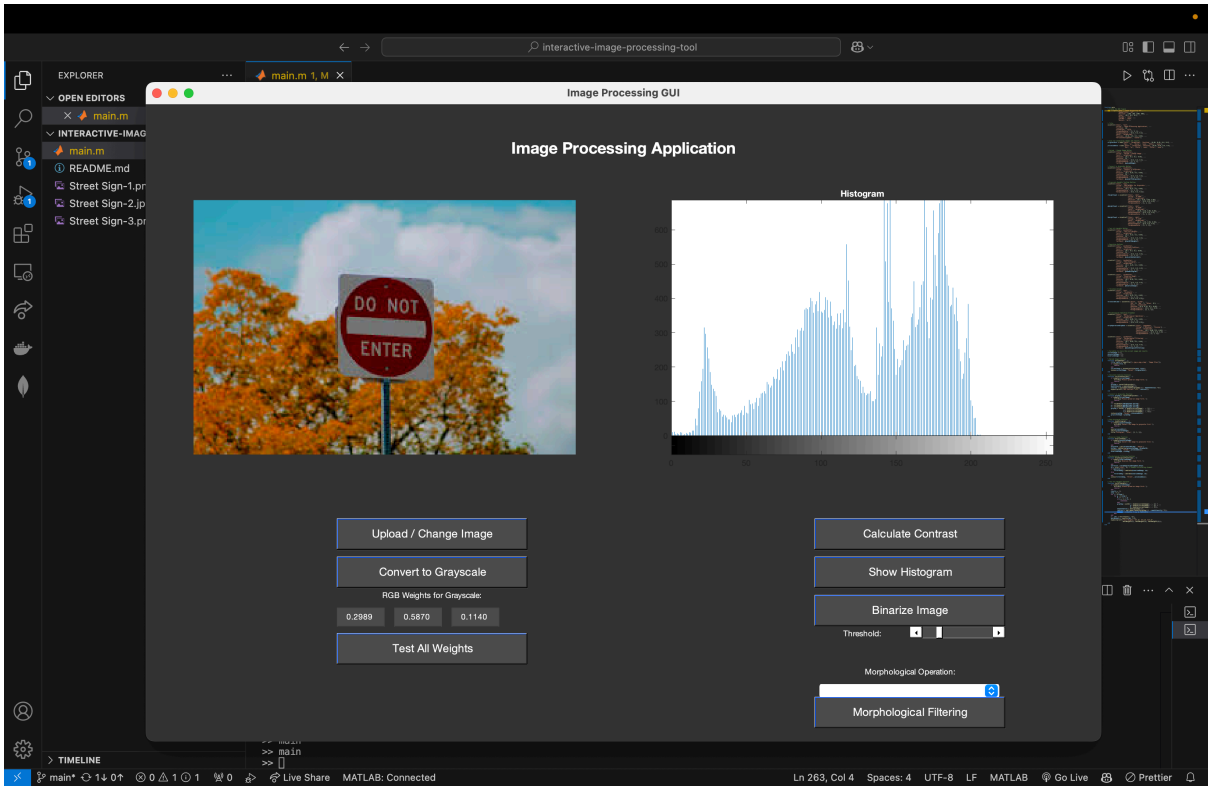


Image Processing GUI
Image Processing Application
Histogram
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

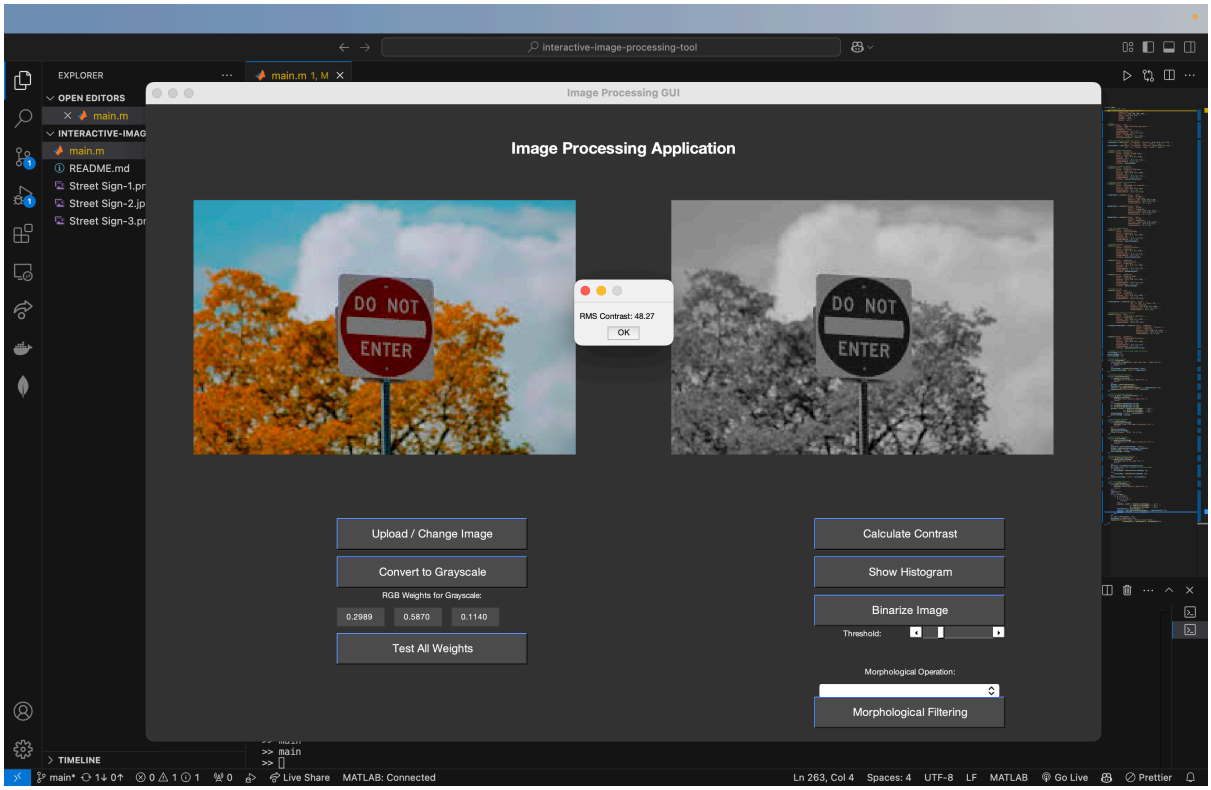


Image Processing GUI
Image Processing Application
RMS Contrast: 48.27
OK
Upload / Change Image
Convert to Grayscale
RGB Weights for Grayscale:
0.2989
0.5870
0.1140
Test All Weights
Calculate Contrast
Show Histogram
Binarize Image
Threshold:
Morphological Operation:
Morphological Filtering

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