EMMETRA INTERNSHIP OPPORTUNITY

ASSIGNMENT SOLUTIONS

TEAM MEMBERS:

Srivanth Srinivasan	1RV22AI058
Anvitha Anant Rao	1RV22AI066
K.M. Amogha	1RV22AI070
Pranshu Bhatt	1RV22AI071

Emmetra

INDEX

ASSIGNMENT - 1

Objective

To design an Image Signal Processing (ISP) pipeline to reconstruct and enhance a 12-bit Bayer RAW image using techniques like demosaicing, white balancing, denoising, gamma correction, and sharpening. This pipeline was developed in a MATLAB app to provide interactive controls for parameter adjustment.

Methodology

1. Preprocessing

- Loaded a 12-bit Bayer RAW image (1920x1280, GRBG format) using MATLAB's `"uigetfile"`
- Normalized the image to [0, 1] and visualized it in the UI.

2. Demosaicing

- Used MATLAB's "`demosaic`" function for RGB reconstruction with edge-aware interpolation.
- Ensured accurate channel alignment for color fidelity.

3. White Balancing

- Implemented a Gray World algorithm to balance colors.
- A slider allowed user-controlled adjustment to compensate for varying lighting conditions.

4. Denoising

- Applied Gaussian filtering to reduce noise while preserving details.
- Noise suppression strength was adjustable via a slider.

5. Gamma Correction

- Used sRGB gamma correction to improve contrast and visibility.

- Adjustable gamma values allowed enhanced shadow and highlight control.

6. Sharpening

- Enhanced edge clarity using an unsharp mask filter with adjustable strength.

7. Enhancements

- **Contrast Adjustment:** Used `imadjust` to stretch intensity values.
- **Saturation Boost:** Enhanced color vibrancy via HSV-based adjustments.

Why MATLAB Over Python?

- **Toolbox Integration:** MATLAB's ***Image Processing Toolbox*** and ***Computer Vision Toolbox*** provided optimized algorithms for RAW image processing, surpassing Python's OpenCV in precision and speed.
- **Visualization Strength:** MATLAB's built-in functions support high-bit-depth images directly.
- **App Development:** MATLAB's App Designer simplified creating an interactive UI without external dependencies..

Results

The visual result of the processed RGB image using the full ISP pipeline is shown below in figure-1.

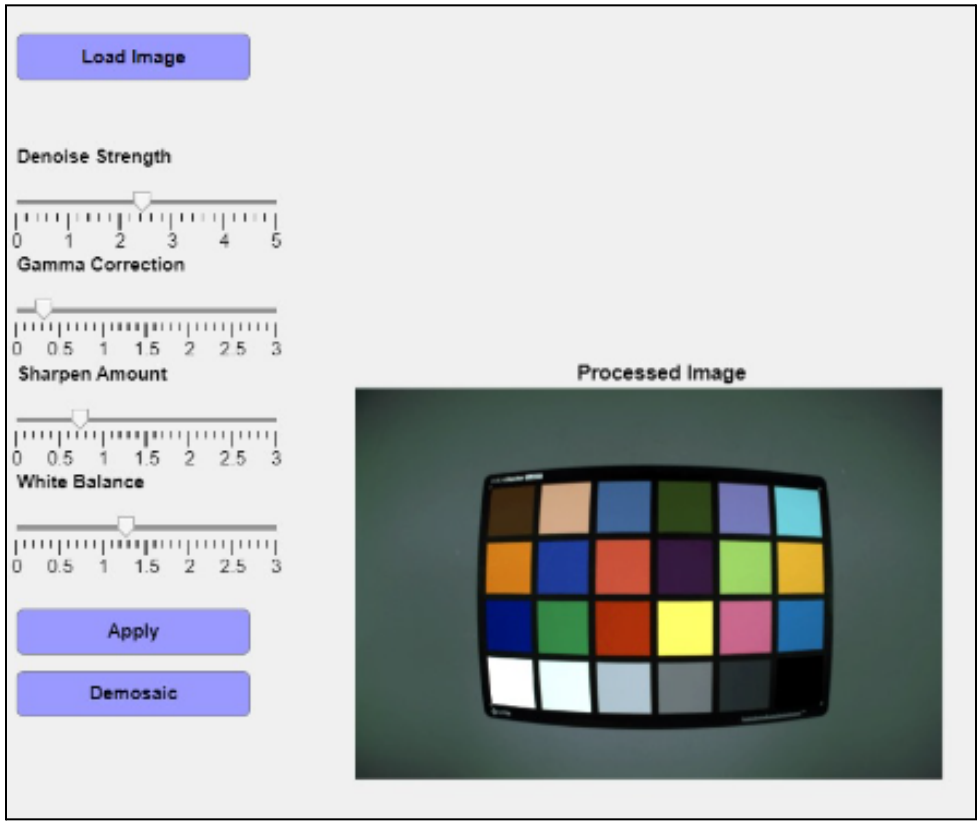


Fig 1- Results after image processing- depicts RGB image

- Demosaicing Performance:
  - Successfully reconstructed full-color image from Bayer pattern
  - Preserved spatial details and color information
  - Minimal interpolation artifacts observed

- White Balance Characteristics:
  - Effectively removed color casts
  - Maintained natural color representation
  - User-adjustable to compensate for different lighting conditions

- Noise Reduction Evaluation:
  - Gaussian filtering provided smooth noise reduction
  - Moderate detail preservation

    - Adjustable noise suppression strength

- Gamma Correction Impact:
    - Improved image visibility
    - Enhanced shadow and highlight details
    - Compensation for non-linear human visual perception

- Sharpening Effects:
    - Enhanced edge details
    - Increased perceived image resolution
    - Controllable enhancement to prevent over-sharpening

Processing Combinations Analyzed:

1. Demosaic + Gamma Correction
2. Demosaic + White Balance + Gamma Correction
3. Demosaic + White Balance + Denoising + Gamma Correction
4. Full Pipeline: Demosaic + White Balance + Denoising + Gamma Correction + Sharpening

Conclusion:

- Successfully implemented an Image Signal Processing pipeline in MATLAB for reconstructing and enhancing Bayer RAW images.
- Achieved expected results using techniques like demosaicing, white balancing, denoising, gamma correction, and sharpening.
- Included interactive controls for parameter adjustment through sliders, offering flexibility and user-defined customization.
- Leveraged MATLAB's specialized toolboxes to streamline development and improve processing precision.

ASSIGNMENT - 2

Objective

The primary objective of this assignment was to implement different image denoising and edge enhancement techniques on a 12-bit Bayer RAW image, process the input using the Image Signal Processor (ISP) pipeline, and assess image quality through Signal to Noise Ratio (SNR) and edge strength measurements.

Methodology

1. Image Loading and Preprocessing:

- **RAW Image Loading:** Bayer RAW images were loaded using the "`uigetfile`" function. Images were demosaiced using the `demosaic` function with GRBG configuration.
- **White Balancing:** Applied a Gray World Algorithm to normalize color channels, ensuring accurate color reproduction.

2. Denoising Techniques:

- **Median Filter**: Removed noise by replacing each pixel's value with the median of its neighbors.
- **Bilateral Filter**: Reduced noise while preserving edges using intensity and spatial proximity.
- **Gaussian Filter**: Applied a Gaussian blur for noise reduction while slightly softening edges.
- **DnCNN (Deep Learning)**: Leveraged a pre-trained DnCNN network to denoise images effectively with minimal detail loss.

3. Sharpening Technique:

- **Laplacian Filter**: Enhanced image sharpness by subtracting the Laplacian (second derivative) from the original image.

4. Performance Metrics:

- **SNR (Signal-to-Noise Ratio)**: Calculated for light, medium, and dark regions of the image to quantify noise suppression effectiveness.
- **Edge Strength**: Measured using gradient-based methods to assess the retention and enhancement of edge details.

5. User Interface:

- **Interactive Features**:
  - Buttons to upload a RAW image and generate processed outputs.
  - Dynamic comparison window displaying images for all methods alongside their computed SNR and edge strength.
- **Visual Layout**: Separate sections for raw, denoised, and sharpened images, allowing easy performance evaluation.

Results:

- GUI:

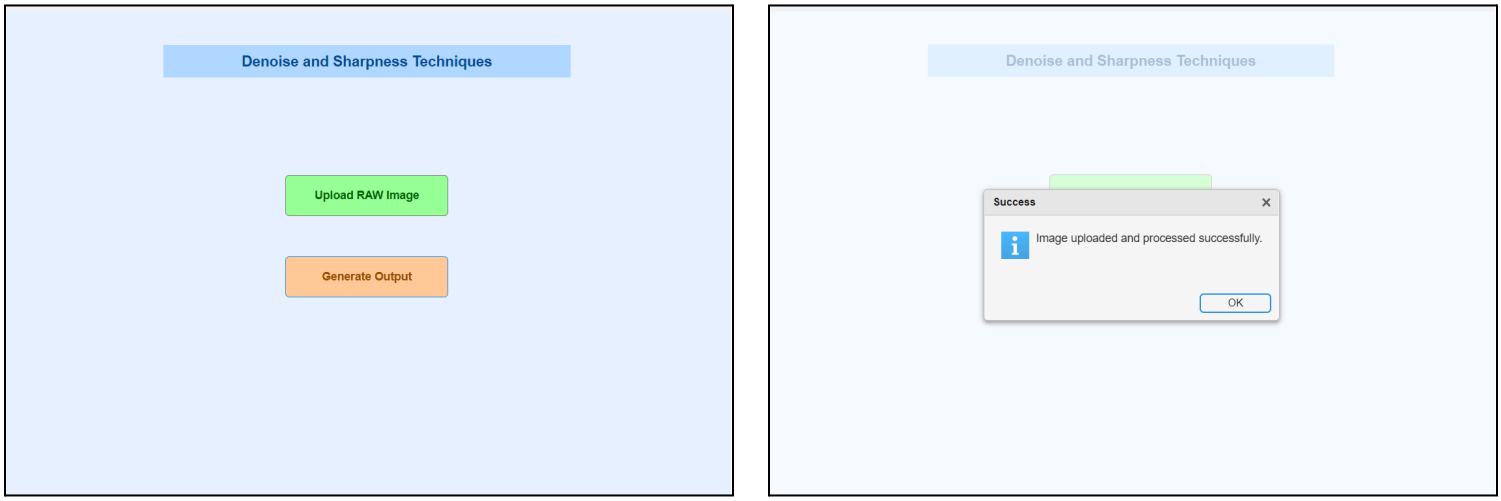


Fig 2-(a) Base GUI for assignment 2 (b)Window after successfully loading RAW image

- Final Output/Comparison:

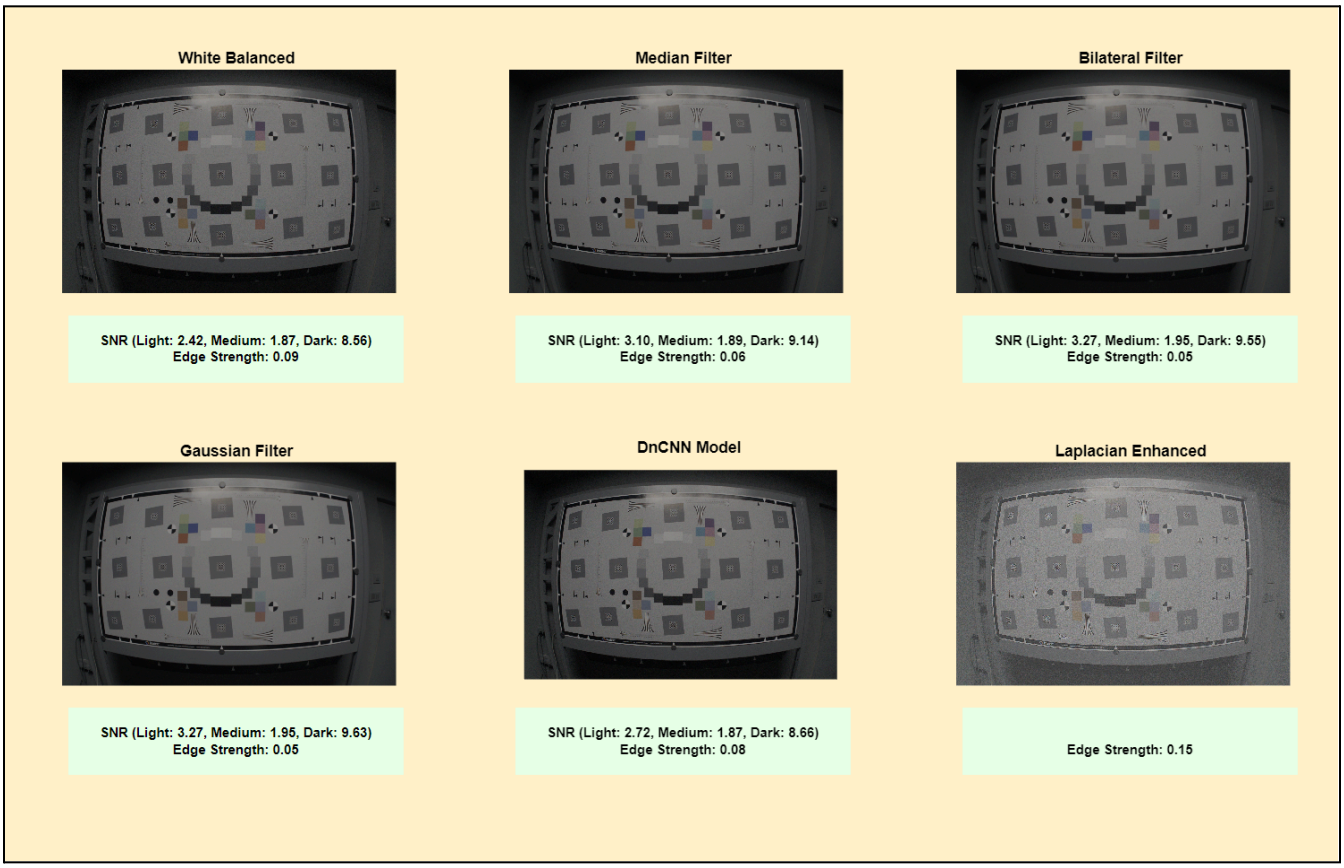


Fig 3- Comparison of images with respective SNR and Edge Strength after processing

Conclusion:

- Successfully developed a MATLAB-based tool to apply and compare denoising and sharpening techniques.
- DnCNN emerged as the most effective denoising method, combining noise suppression and detail preservation.
- The Laplacian Filter enhanced edge clarity, adding sharpness to images.
- Utilized MATLAB's robust toolboxes and visualization capabilities streamlined implementation and comparison.

ASSIGNMENT - 3

Objective:

The objective of this assignment is to implement an HDR imaging algorithm to merge and tone map three differently exposed LDR (Low Dynamic Range) images of a high-contrast scene. The goal is to preserve details in both highlights and shadows while producing a visually appealing 8-bit image suitable for display.

Implementation Details:

- Three images were taken using manual exposure control on a smartphone with exposure times of approximately 0.04 seconds (bright), 0.005 seconds (mid) and 0.01 seconds (dark).

- Exposure times for the images were obtained from the **EXIF (Exchangeable Image File Format)** data embedded in the image files. The EXIF data provides detailed metadata about the photo, including exposure details.

- The three input images were read using OpenCV.

- HDR Merging Algorithms:
    - Debevec Algorithm: Uses camera response calibration to merge images and create an HDR image.
    - Robertson Algorithm: Iterative approach to align and merge exposures into an HDR image.
    - Mertens Fusion: Exposure fusion technique to create a visually pleasing image without requiring HDR tone mapping.

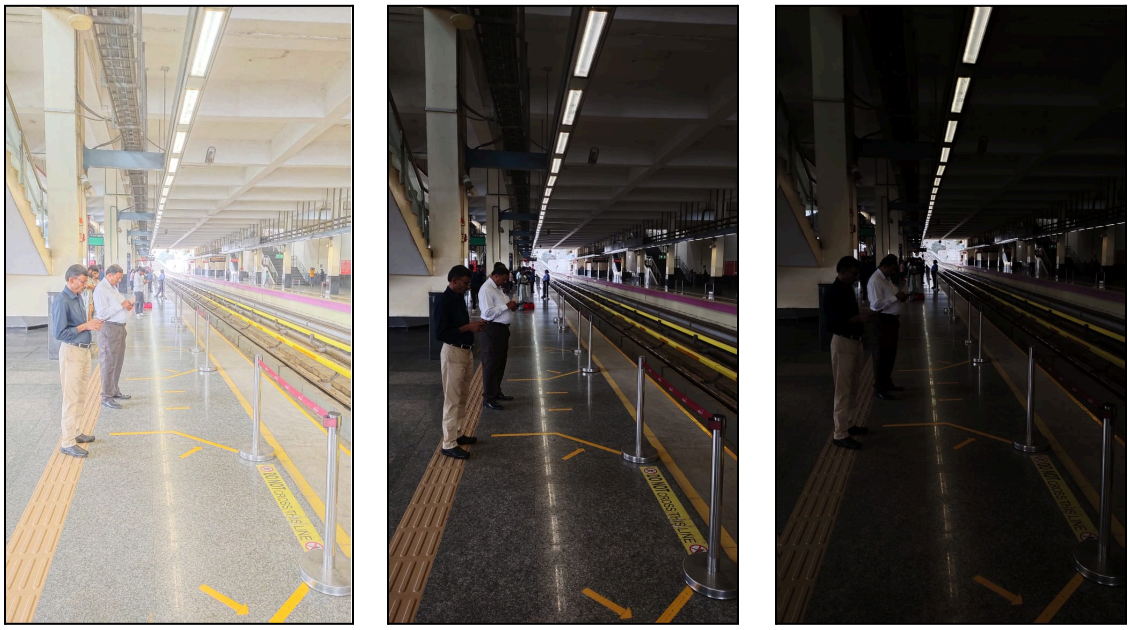
- Tone Mapping:
    - HDR images are converted to an 8-bit range for display using tone mapping.
    - Gamma Correction: Adjusts the brightness curve to retain details.
    - The resulting images are scaled to 8-bit for saving.

- Histogram Analysis:
    - Histograms for pixel intensity distributions (R, G, B channels) were plotted using Matplotlib to analyze the results.
    - Comparing the input and output histograms shows how details from dark and bright areas are combined in the HDR image, demonstrating the effectiveness of the merging process.

- UI Development:
    - The user interface was built using **Streamlit**, featuring a responsive layout with input images, histograms, and interactive buttons to view HDR outputs (Debevec, Robertson, Mertens).
    - Custom CSS styling enhanced the visual appeal, while dynamic histogram plotting and automated app launching improved user interactivity and accessibility.

Output/Comparison:

Example 1: Metro Station Platform



Bright
(Exposure time = 0.04 s)

Shadow
(Exposure time = 0.01 s)

Low light
(Exposure time = 0.005 s)

Fig 4- Three differently exposed LDR images captured from phone are given as input

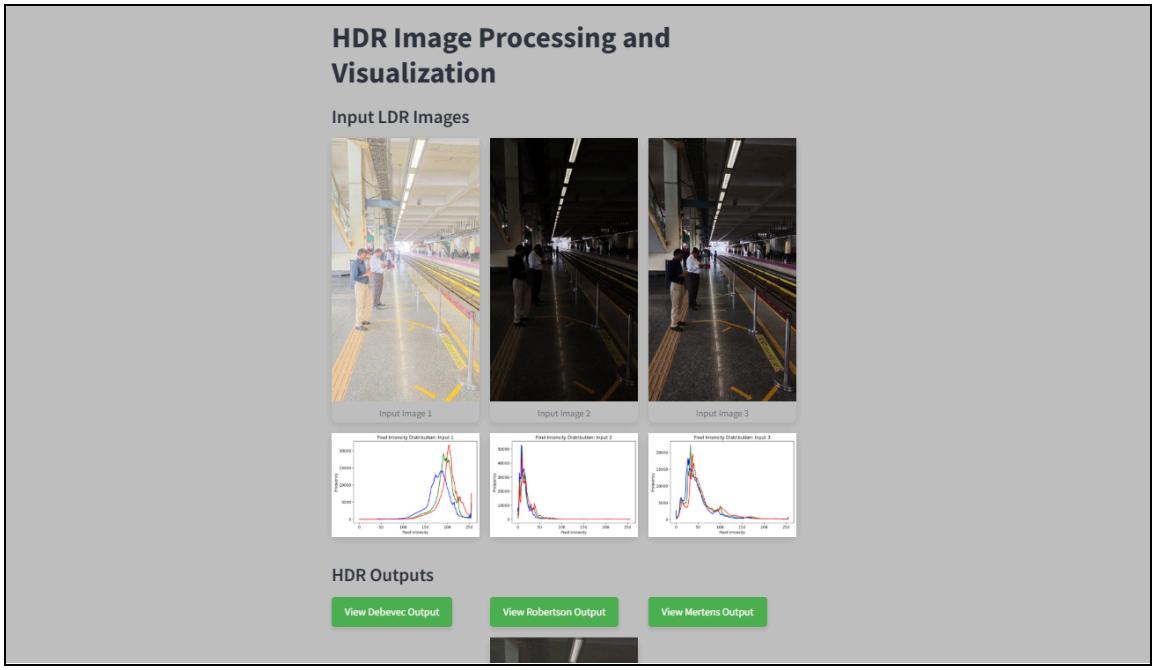


Fig 5- Streamlit UI interface

Debevec algorithm output

Robertson algorithm output

Mertens algorithm output

Fig 6- The Output section has three buttons, each for viewing the respective resultant HDR images along with the histograms

Example 2: Terrace Scene

Bright
(Exposure time = 0.04 s)

Shadow
(Exposure time = 0.01 s)

Low light
(Exposure time = 0.0025 s)

Fig 7- Three differently exposed LDR images captured from phone are given as input

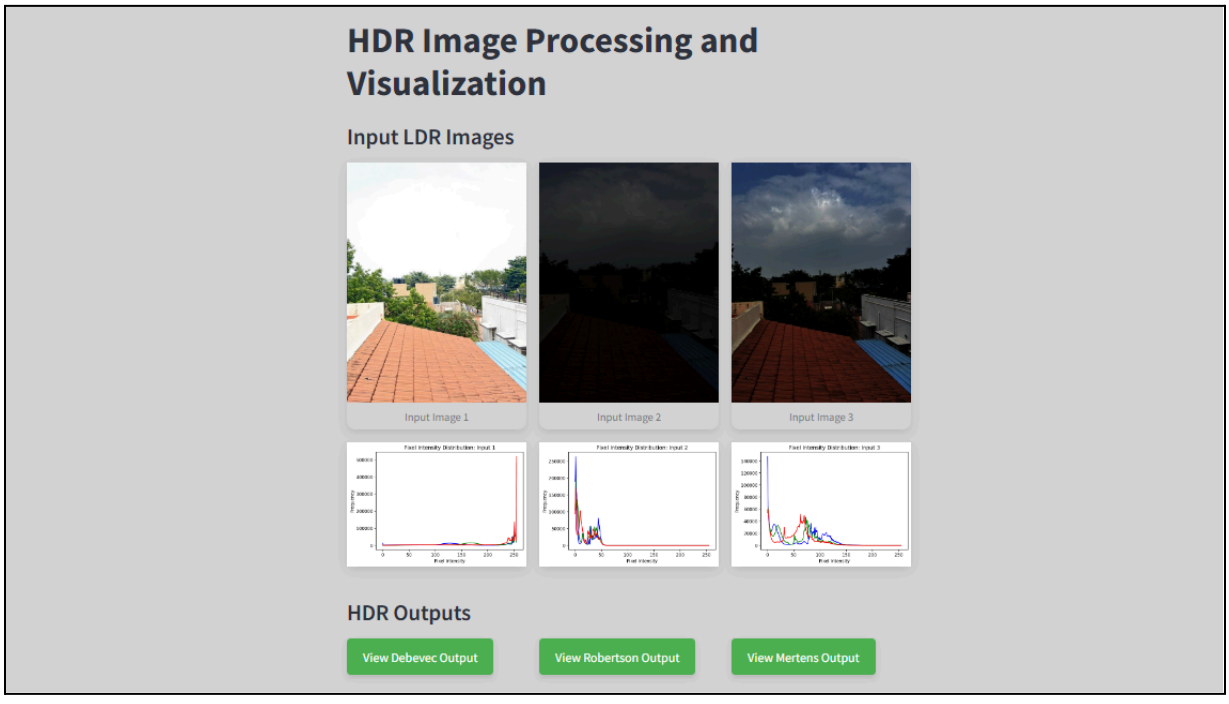
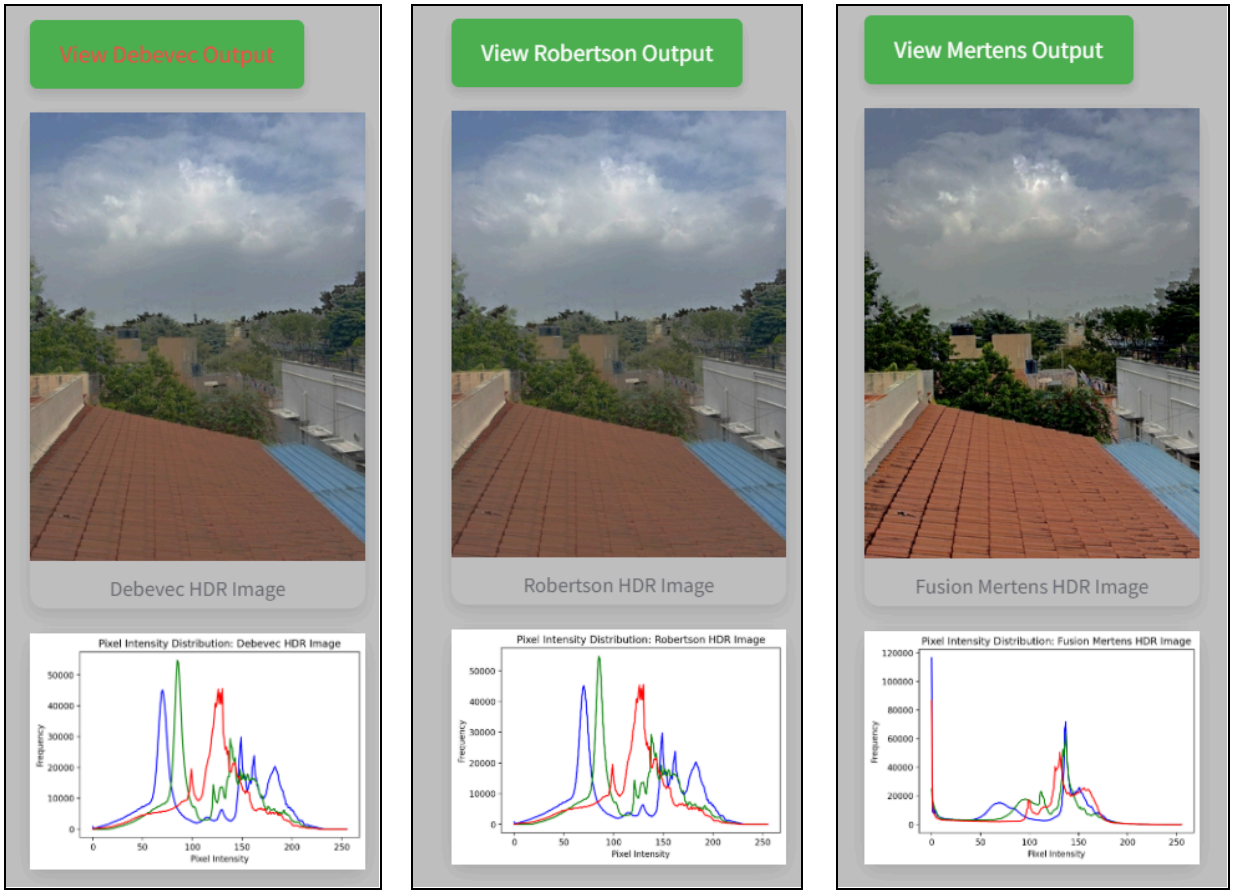


Fig 8- Streamlit UI Interface



Debevec algorithm output | Robertson algorithm output | Mertens algorithm output

Fig 9- The Output section has three buttons, each for viewing the respective resultant HDR images along with the histograms

Observations:

- **Output Images**
  - Debevec and Robertson:
    Both algorithms produced similar HDR outputs. This is because they fundamentally aim to reconstruct a high dynamic range image by merging exposure data using similar principles.
  - Mertens Fusion:
    The Mertens algorithm produced an output with balanced exposure but less dynamic range compared to Debevec and

Robertson as it focuses on visual appearance rather than true HDR merging.

- **Histograms**
  - The histograms show the distribution of pixel intensities for each color channel (Red, Green, Blue).
  - In HDR processing, the goal is to capture a wide range of intensities, from shadows to highlights, and the histogram provides a way to check how well the HDR process has worked.
  - A well-processed HDR image should show a balanced distribution of intensities without being overly skewed toward either dark or bright regions.

- **Comparison**

Algorithm	Strengths	Weaknesses
Debevec	True HDR merging, good shadow & highlight details.	Computationally intensive.
Robertson	Handles exposure alignment, robust merging.	Slightly less contrast than Debevec.
Mertens	Fast, visually pleasing results.	Not a true HDR, reduced dynamic range.

REFERENCES

[1] ***Debevec, P., & Malik, J. (1997). "Recovering High Dynamic Range Radiance Maps from Photographs"***
This paper provides insights into the Debevec algorithm for merging exposure sequences to create HDR images.

[2] ***Robertson, M. A., et al. (2003). "Estimation-Theoretic Approach to Dynamic Range Enhancement Using Multiple Exposures"***
The paper introduces the Robertson Algorithm, used for enhancing dynamic range using multiple exposures, supporting the objective of HDR image generation and tone mapping.

[3] ***A. Abdelhamed, R. Timofte, M. S. Brown, et al., "NTIRE 2019 Challenge on Real Image Denoising: Methods and Results,"***
The NTIRE 2019 challenge addressed real-world image denoising, comparing various state-of-the-art methods on realistic noisy datasets, highlighting the strengths and limitations of supervised approaches in practical scenarios.

[4] ***K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising"***
The paper proposes a residual learning framework for image denoising using a deep convolutional neural network (DnCNN), significantly improving denoising performance across various noise levels.

[5] ***J. Liu, C.-H. Wu, Y. Wang, Q. Xu, Y. Zhou, H. Huang, C. Wang, S. Cai, Y. Ding, H. Fan, and J. Wang, "Learning Raw Image Denoising with Bayer Pattern Unification and Bayer Preserving Augmentation,"***
This paper introduces a unified framework for raw image denoising that leverages Bayer pattern unification and Bayer-preserving data augmentation, optimizing models for better alignment with camera sensors' raw data.

[6] ***OpenCV Documentation: HDR and Tone Mapping***
This website demonstrates how to generate and display HDR images from an exposure sequence using various OpenCV functions. It covers how to use exposure fusion to merge an exposure sequence through Mertens algorithm.