ASSIGNMENT 1

Basic Implementation of Image Signal Processor (ISP)

Introduction

This project is a foundational implementation of an Image Signal Processor (ISP), designed to process raw image data and apply essential image processing functions. The program demonstrates basic ISP operations, including Debayering, noise reduction, sharpening, and other signal adjustments.

Features

This ISP implementation includes the following five key algorithms for an efficient and effective image processing pipeline:

1. **Demosaic Layer with Edge-Based Interpolation**
   Reconstructs full-color images from sensor data using edge-based interpolation, enhancing detail by minimizing color artifacts and preserving natural edges.
2. **White Balance**
   Adjusts color temperature to ensure accurate color reproduction, correcting for different lighting conditions so whites appear neutral and colors are balanced.
3. **Denoise Using a Gaussian Filter**
   Reduces noise in low-light or high-ISO images by smoothing unwanted artifacts while preserving key details, resulting in a cleaner overall image.

4. **Gamma Correction**
   Applies a gamma curve to adjust brightness and contrast, ensuring a natural tonal range and accurate midtones for a visually balanced output.
5. **Sharpening Filter**
   Enhances image clarity by accentuating edges and fine details, creating a sharper, more defined look without introducing additional noise.

User Interface
This ISP implementation includes a user-friendly interface designed to simplify interaction, allowing users to visualize each algorithm's effect on the image.

Demosaicing

Demosaicing is a digital image processing technique used to reconstruct full-color images from raw data captured by image sensors, which typically employ a Bayer filter array. Each pixel in this array captures only one color (red, green, or blue), meaning the other two color values must be interpolated. Demosaicing algorithms are essential for filling in these missing color values and producing a complete color image.

Demosaicing Techniques

1. **Nearest Neighbor Interpolation**

- **Description**: This is the simplest method, where each missing color value is filled by duplicating the nearest neighbor’s color.
- **Use Cases**: Primarily used in low-cost or low-power applications where processing resources are limited.
- **Limitations**: It often leads to significant color artifacts, such as “color fringing” and low image quality, especially around edges.

2. **Bilinear Interpolation**
    - **Description**: In this technique, the missing color values are calculated by averaging the colors of the surrounding pixels.
    - **Use Cases**: Commonly used in basic image processing pipelines due to its simplicity and efficiency.
    - **Limitations**: Bilinear interpolation can produce blurry images, with noticeable artifacts around edges due to its inability to distinguish edges from flat areas.
3. **Edge-Aware Demosaicing**
    - **Description**: Edge-aware algorithms detect edges in the image and adjust the interpolation process to avoid blending colors across these edges. This helps maintain image sharpness and reduces color artifacts.
    - **Use Cases**: Suitable for high-quality imaging where edge preservation is critical, such as in photography or videography.
    - **Limitations**: Computationally more intensive, making it less ideal for real-time or low-power applications.

4. **Frequency-Based Demosaicing**
    - **Description**: Frequency-based algorithms analyze the spatial frequencies in the image and apply different interpolation methods based on these frequencies, effectively separating high-frequency (edge) and low-frequency (flat) regions.
    - **Use Cases**: Common in professional imaging devices where high accuracy is essential.
    - **Limitations**: These algorithms require significant processing power and can introduce artifacts if frequencies are misinterpreted.
5. **Machine Learning-Based Demosaicing**
    - **Description**: This technique uses trained models to predict missing color values, allowing for highly accurate color reconstruction, even in complex textures and edges.
    - **Use Cases**: Increasingly used in smartphone cameras and other devices where high-quality image processing is required.
    - **Limitations**: Requires a large amount of training data and processing power, making it challenging for devices with limited resources.

Use Cases

- **Consumer Cameras**: Demosaicing is essential in consumer cameras (smartphones, DSLRs) to deliver high-quality images from compact sensors.

- **Medical Imaging**: Edge-aware and machine learning-based techniques are often used to ensure clarity and detail in diagnostic imaging.
- **Surveillance and Security**: Lower-cost methods like bilinear interpolation may be used for cost-effectiveness, though edge-aware methods are used when detail is necessary.

Limitations of Demosaicing

While demosaicing is vital for color reconstruction, it has inherent limitations:

- **Artifacts**: Demosaicing can introduce color artifacts, such as moiré patterns, color fringing, and false colors around edges.
- **Processing Requirements**: More advanced techniques like edge-aware and machine learning-based demosaicing require significant processing power and memory, limiting their usage in real-time or low-power devices.
- **Edge and Detail Preservation**: No demosaicing algorithm perfectly preserves edges and fine details; trade-offs between sharpness, color accuracy, and noise are often required.

White Balance in Cameras

White balance (WB) is a critical process in photography and videography that ensures colors are rendered accurately under different lighting conditions. It adjusts the colors in an image so that the whites appear neutral, allowing other colors to be reproduced correctly. Without proper white balance, images can take on an unnatural color cast, such as yellowish or bluish tones.

Understanding Color Temperature

Color temperature is measured in Kelvin (K) and refers to the hue of a specific type of light source. Different light sources emit varying colors:

- **Tungsten Light (Incandescent)**: ~2800K (warm, yellowish)
- **Fluorescent Light**: ~4000K (cool, greenish)
- **Daylight**: ~5500K to 6500K (neutral, white)
- **Shade**: ~7000K to 8000K (cool, bluish)

Denoising with Gaussian Filter

Denoising is the process of reducing unwanted noise from an image, which can be caused by various factors, including low light, high ISO, or sensor limitations. The Gaussian filter is a common method for denoising, using a Gaussian function to blur the image and smooth out noise.

- **Description**: A Gaussian filter applies a weighted average to pixels, giving more weight to central pixels. This smooths out noise but can blur edges slightly.
- **Use Cases**: Suitable for applications where noise reduction is important, such as low-light photography.
- **Limitations**: Excessive denoising can cause a loss of fine details and texture.

Gamma Correction

Gamma correction adjusts the brightness of an image by applying a nonlinear transformation to its pixel values. This process is crucial for displaying images on different devices, as gamma correction helps maintain consistent brightness and contrast.

- **Description**: Gamma correction modifies image intensity to ensure the midtones appear balanced.
- **Use Cases**: Essential for ensuring accurate color and brightness on screens, especially in photography and digital displays.
- **Limitations**: Incorrect gamma settings can result in images that are either too dark or too washed out.

Sharpening

Sharpening enhances the clarity of an image by accentuating edges and fine details. This process can make images appear crisper and more defined, although excessive sharpening can introduce noise and artifacts.

- **Description**: Uses a sharpening filter to amplify edges, enhancing detail.
- **Use Cases**: Often applied in post-processing for digital photography and video.
- **Limitations**: Over-sharpening can cause unnatural artifacts and noise. **Example 1**

image 1

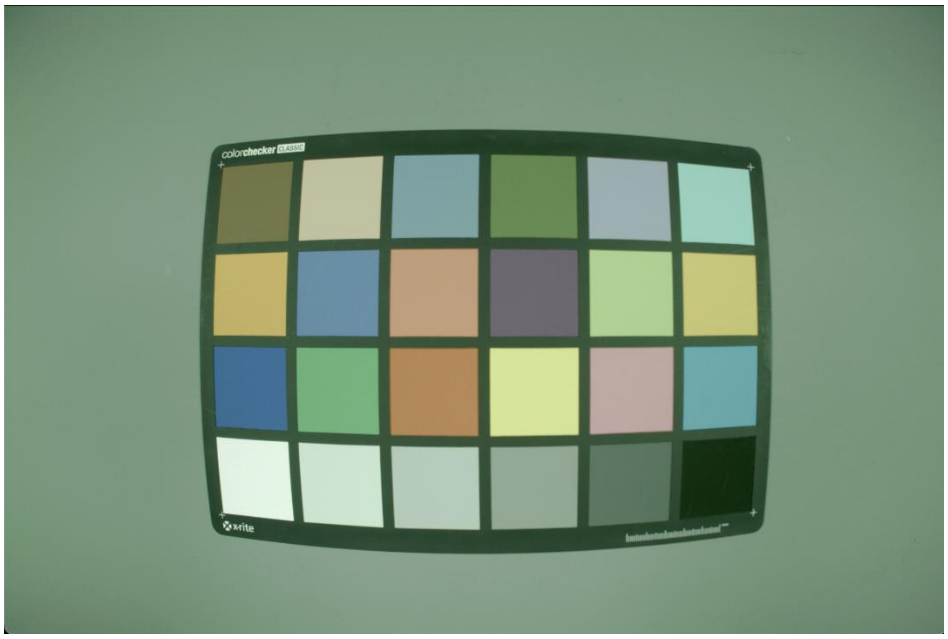
Image Type	Image
With Demosaicing And Gamma Correction	

image 2

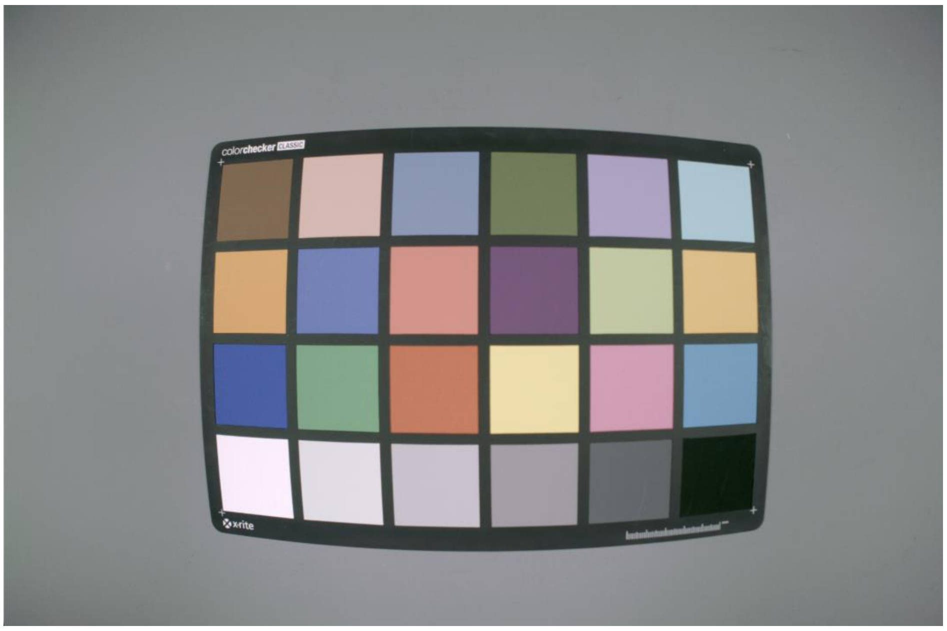
Image Type	Image
With Demosaicing, White Balance And Gamma Correction	

image 3

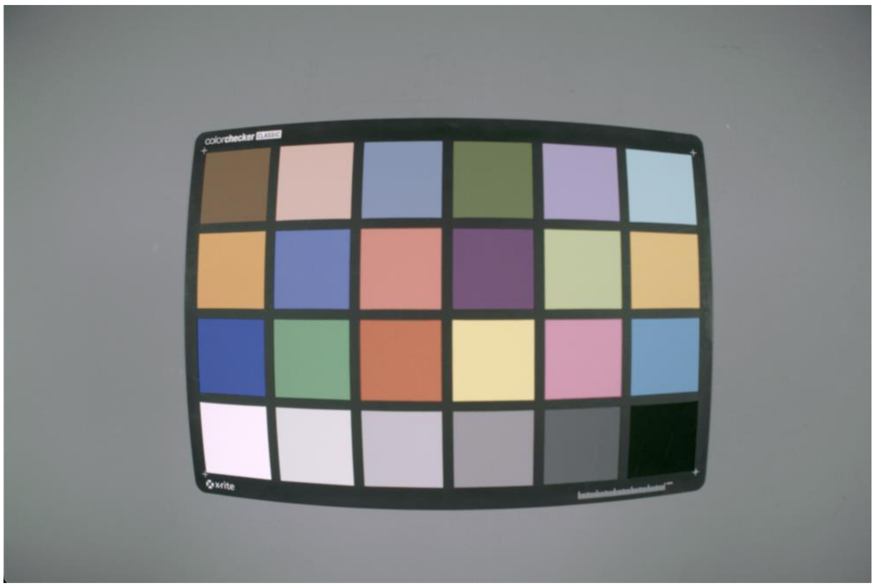
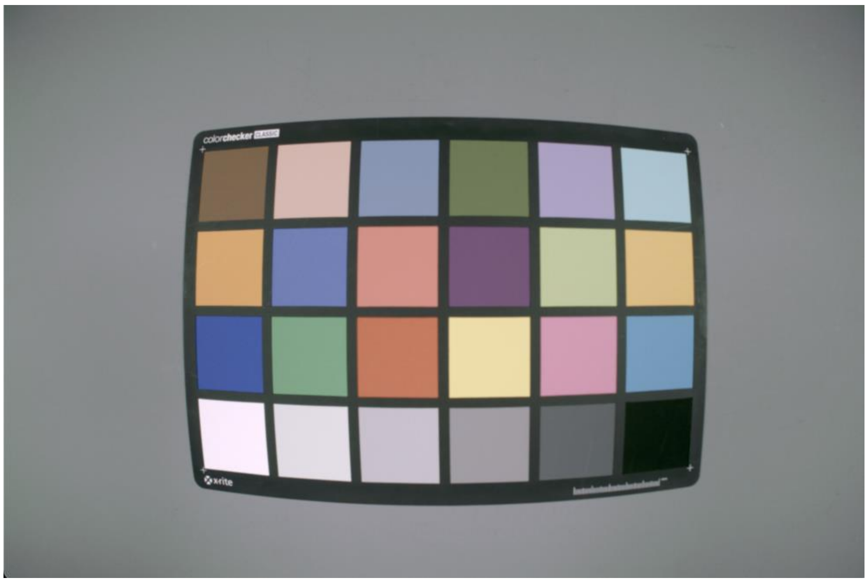
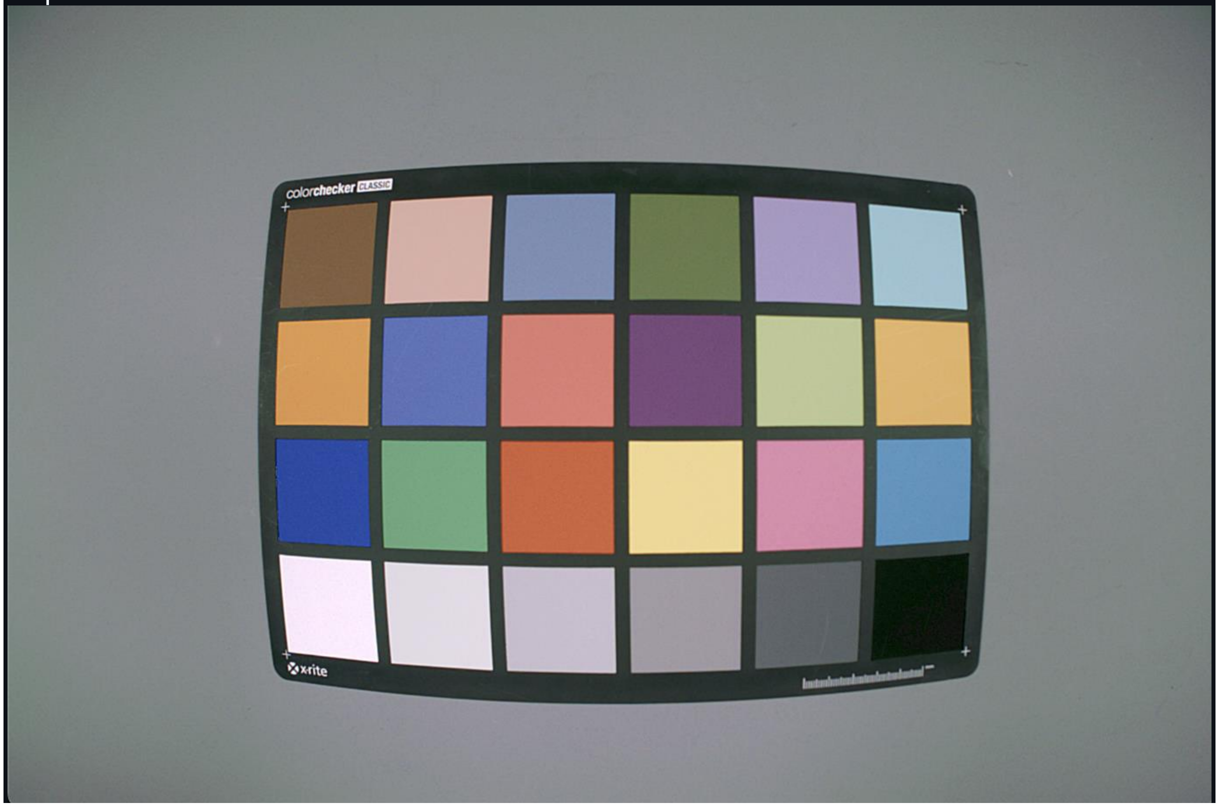
Image Type	**Image**
With Demosaicing and Denoising,White Balance, and Gamma	

image 4

Image Type	**Image**
With Demosaicing, White Balance, Denoising, Gamma Correction, and Sharpening	

#Final Image After Basic ISP
Implementation



ASSIGNMENT 2

Image Denoising and Enhancement Techniques

This repository implements various denoising and enhancement techniques for 12-bit RAW images. The goal is to apply and compare methods, including AI-based denoising, median filtering, bilateral filtering, and edge enhancement techniques using a Laplacian filter. The output is a 24-bit RGB image, suitable for further processing and analysis.

Overview

- **AI Denoising**: Utilizes a U-Net Neural Network model.
- **Traditional Filtering Techniques**: Includes median and bilateral filters.
- **Edge Enhancement**: Uses a Laplacian filter for edge enhancement.
- **Evaluation Metrics**: Measures spatial signal-to-noise ratio (SSNR) and edge strength.
- **Code Structure**: Organized and modular for easy understanding and contributions.

Technologies Used

- **Python**
- **OpenCV**
- **PyTorch**
- **NumPy**
- **Matplotlib** (for visualization)

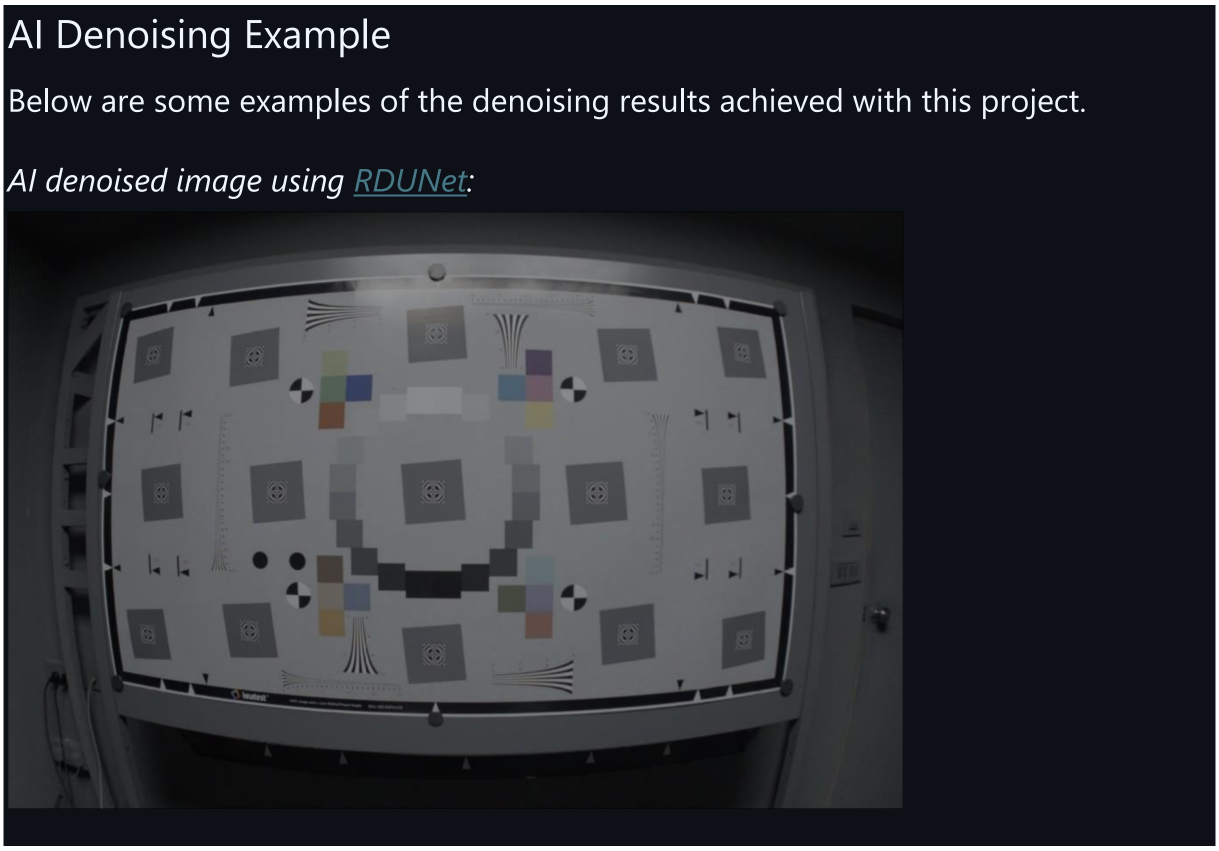
Pre-trained Model

- **Model**: The pre-trained model model_color.pth can be downloaded and used for the denoising process.
- **Location**: Place model_color.pth in the model_color/ directory.

Denoising Results

AI Denoising Example

- **Model**: RDUNet AI-based denoising.



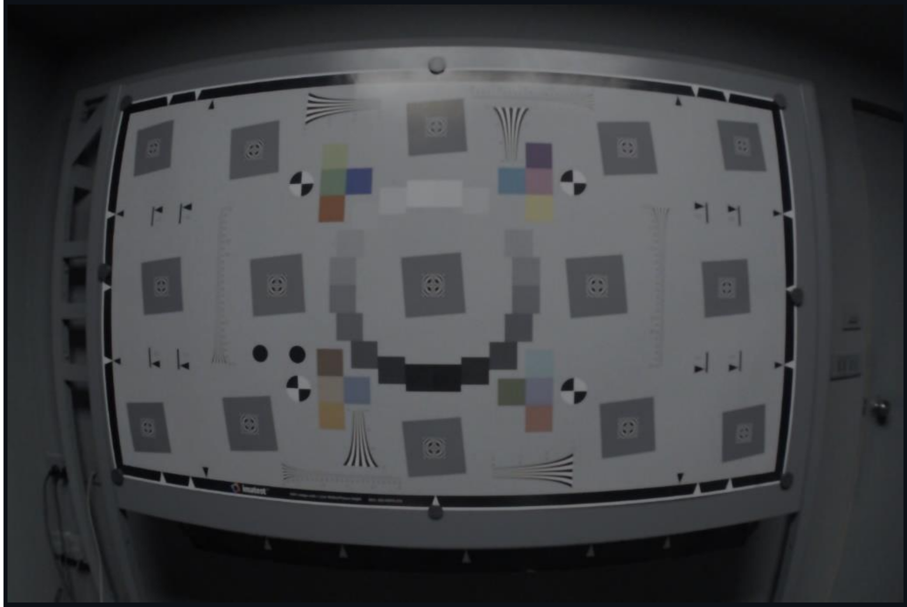
AI Denoising Example

Below are some examples of the denoising results achieved with this project.

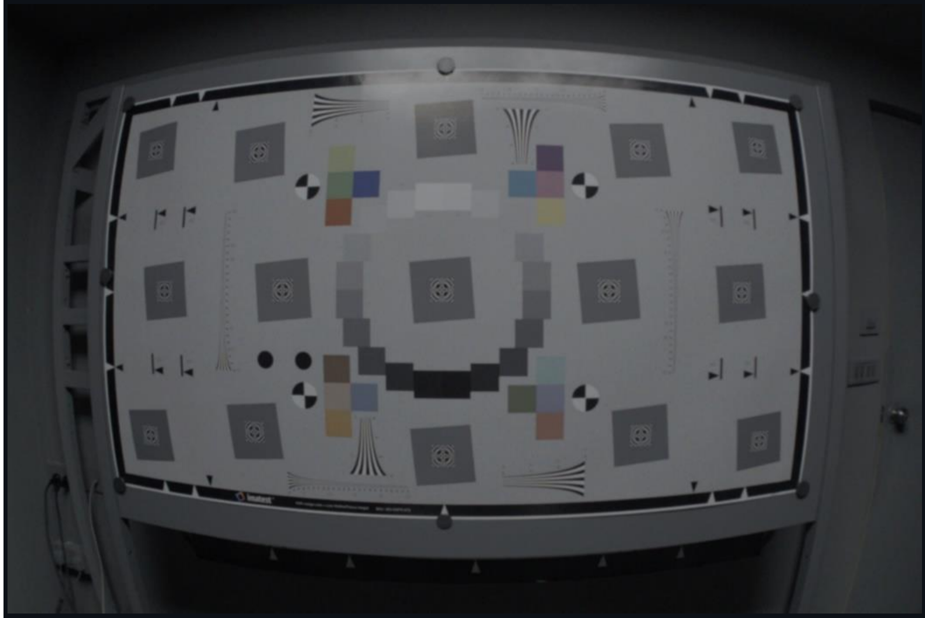
AI denoised image using RDUNet:

Tradtional Example

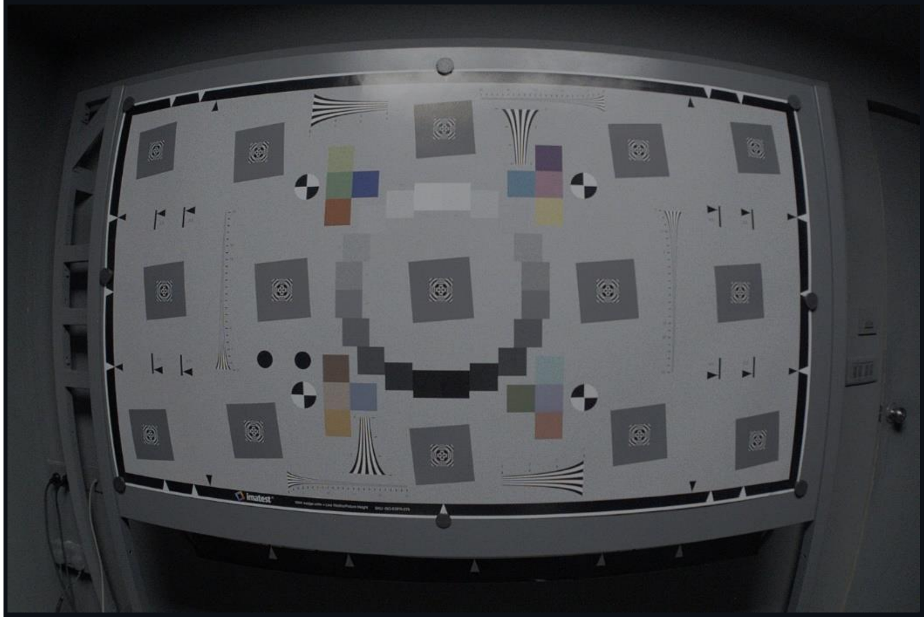
Bilateral filter:



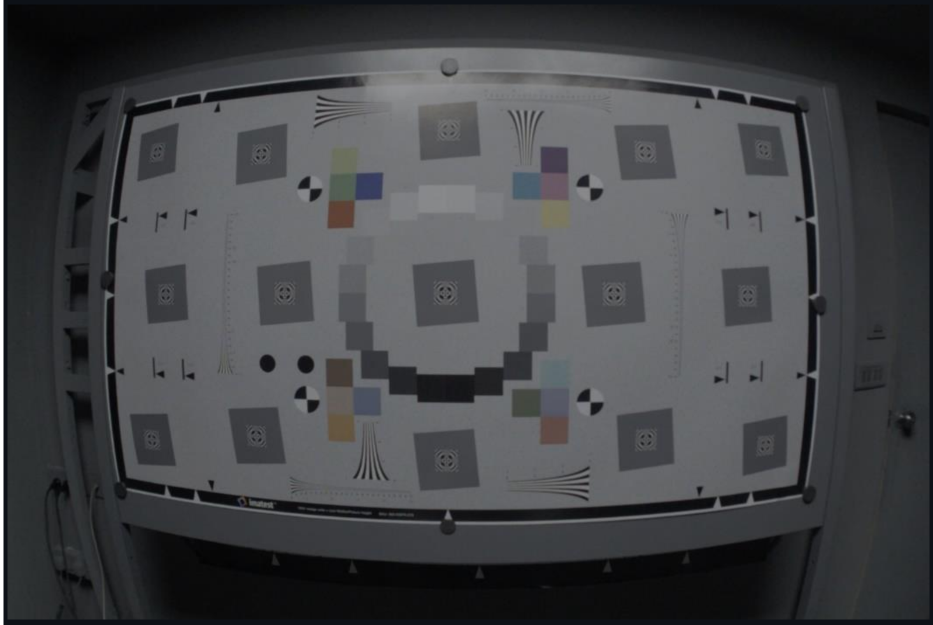
Gaussian filter (Implemented in Assignment 1):



Laplacian filter



Median filter:



Evaluation Metrics

Traditional Examples

- **Bilateral Filter**: Traditional filter that smooths noise while preserving edges.

- **Gaussian Filter**: A basic smoothing filter (implemented in Assignment 1).
- **Laplacian Filter**: Emphasizes edges by calculating the second derivative of the image.
- **Median Filter**: Reduces noise by replacing pixel values with the median of surrounding pixels.

Evaluation Metrics

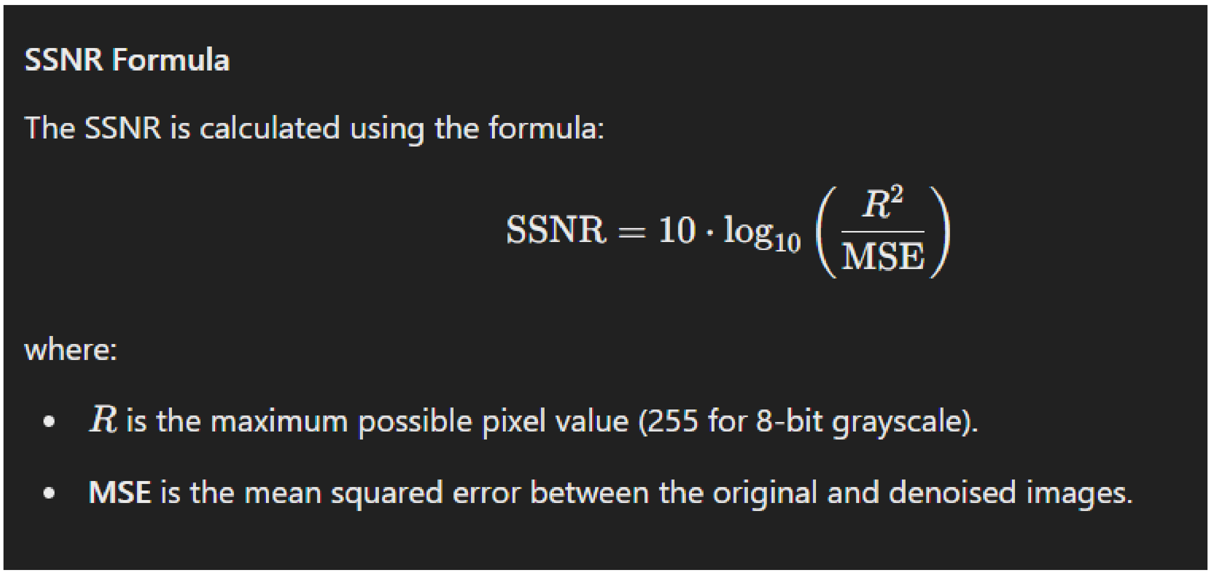
Spatial Signal-to-Noise Ratio (SSNR)

This section compares SSNR for different gray tones using various denoising and enhancement methods.

Methods Implemented

1. **Laplacian Filter for Edge Enhancement**
   Highlights rapid intensity changes by calculating the second derivative of the image.
2. **Denoising Techniques Compared**
   - **Median Filter**: Reduces noise and maintains edge sharpness.
   - **Bilateral Filter**: Smooths noise while preserving edges by considering spatial and intensity differences.
   - **Gaussian Filter**: Smooths the image by averaging pixel values, but may blur edges.
   - **Edge Enhancement (from Assignment 2)**: Previously implemented edge enhancement technique.

SSNR Formula



SSNR Formula

The SSNR is calculated using the formula:

$$\mathrm{SSNR} = 10 \cdot \log_{10}\left(\frac{R^2}{\mathrm{MSE}}\right)$$

where:

- R is the maximum possible pixel value (255 for 8-bit grayscale).
- MSE is the mean squared error between the original and denoised images.

Image Filter Performance Analysis

Configuration

Input Parameters

- **Format**: Bayer 12-bit
- **Pattern**: GRBG
- **Resolution**: 1920x1280

Output Parameters

- **Format**: RGB
- **Bit Depth**: 24-bit (8 bits per channel)

Tools Used

- **PixelViewer**
- **Irfanview** (with RAW plugin)
- **TensorFlow** (for CNN-based denoising)

Performance Metrics

Filter Type	PSNR (dB)	SNR (dB)	Edge Strength (Original)	Edge Strength (Processed)
Median	42.31	32.26	261.88	211.09
AI Denoised	40.06	30.02	261.88	159.42
Gaussian	38.98	28.94	261.88	178.64
Bilateral	37.92	27.88	261.88	145.96
Laplacian	28.39	18.35	261.88	477.54

Key Findings

Filter Rankings (by PSNR)

1. **Median Filter**: 42.31 dB
2. **AI Denoised**: 40.06 dB
3. **Gaussian Filter**: 38.98 dB
4. **Bilateral Filter**: 37.92 dB
5. **Laplacian Filter**: 28.39 dB

Edge Preservation Analysis

- **Best Edge Retention**: Median Filter (211.09)
- **Strongest Edge Enhancement**: Laplacian (477.54)
- **Most Edge Smoothing**: Bilateral (145.96)

Notable Observations

- **Median Filtering** provides an optimal balance between noise reduction and edge preservation.

- **AI-based Denoising** shows promising results with the second-best PSNR.
- **Laplacian Filter** amplifies edges significantly but introduces noise, resulting in a lower PSNR.

Model Information

- **Framework**: TensorFlow
- **Architecture**: CNN-based denoising
- **Implementation**: Uses a pre-trained model

ASSIGNMENT 3

HDR Image Processing Application

This repository provides an application for HDR (High Dynamic Range) image processing using multiple techniques. The application is written in Python and utilizes OpenCV.

Introduction

HDR imaging allows capturing images with a broader range of luminosity levels than standard digital imaging. This application implements several HDR techniques to create realistic images by merging multiple exposures and applying different tone-mapping algorithms.

Features

- **Multi-Exposure Support**: Load multiple images with different exposures (underexposed, normal, overexposed) to create a single HDR image.
- **Automatic Exposure Detection**: Identifies and categorizes input images based on their exposure levels.
- **HDR Merging Algorithms**: Supports multiple merging algorithms, including Merten’s and Debevec's methods, for blending exposures.
- **Image Alignment**: Automatically aligns input images to correct for slight camera movements between shots, ensuring a seamless HDR result.
- **Tone Mapping Techniques**: Offers different tone mapping methods, such as Reinhard, Drago, Mantiuk, and Local Tone Mapping.
- **Real-Time Preview**: Enables users to see the effect of tone mapping adjustments on the HDR output.
- **Intuitive UI & Enhanced UX**: Designed with a user-friendly interface, making HDR creation straightforward for users at all skill levels.
- **Image Export**: Allows saving the processed HDR images in common formats.

Usage

1. **Load Images**: Upload up to three images with different exposures (e.g., underexposed, properly exposed, overexposed) for merging.
2. **Select HDR Merging Method**: Choose from available merging algorithms, including Merten's merge, Debevec, and simple weighted.

3. **Apply Tone Mapping**: Customize tone mapping settings to enhance the final image's detail, brightness, and contrast.
4. **Save Output**: Export your HDR image to your desired format.

HDR Merging Algorithms

Simple Weighted Merge

- **Description**: Combines images by assigning weights to pixel values based on brightness, resulting in a balanced image that preserves details across the range of exposures.
- **How It Works**:
    - **Exposure-Based Weighting**: Bright, well-exposed areas have more influence, while underexposed or overexposed regions are given lower weights.
    - **Merging**: Combines all weighted images, emphasizing well-exposed areas and reducing the effect of under- or overexposed regions.
- **When to Use**: Works well for scenes with moderate lighting differences, such as indoor spaces with natural light.
- **Limitations**: May struggle with scenes with significant contrast, like those with direct sunlight and deep shadows.

Merten's Merge

- **Description**: Enhances HDR merging by assigning weights based on contrast and saturation, preserving more fine-grained details.
- **How It Works**:
    - **Contrast and Saturation Weighting**: Gives higher weights to pixels with more contrast and color richness.

    - **Blending Process**: Prioritizes detail-rich areas, creating a visually compelling HDR output.
- **When to Use**: Ideal for scenes with textures or varied lighting, like landscapes or architectural photos.
- **Limitations**: Slower than simpler methods and may introduce artifacts in low-contrast scenes.

Debevec

- **Description**: A classic HDR approach that uses a camera response function to account for the non-linear relationship between pixel brightness and light intensities.
- **How It Works**:
    - **Response Function Calculation**: Uses multiple exposure images to calculate a response function for each pixel, accurately representing scene brightness.
    - **HDR Creation**: Merges images based on the response function and exposure levels to produce a realistic HDR output.
- **When to Use**: Useful for scenes with extreme lighting variations, like sunsets, where detail preservation is essential.
- **Limitations**: Requires calibration and is resource-intensive.

Tone Mapping Techniques

Overview

Tone Mapping converts the high dynamic range of an HDR image to a standard dynamic range for display while preserving visual details. This application offers both local and global tone mapping methods.

Local Tone Mapping

- **Description**: Adjusts brightness and contrast in small regions, enhancing textures and details in specific image parts.
- **How It Works**:
  - **Adaptive Adjustments**: Evaluates brightness and contrast on a pixel-by-pixel basis, preserving local contrast.
- **When to Use**: Effective for scenes with complex lighting or where textures are critical, like forests or architectural interiors.
- **Limitations**: Can be computationally intensive and may introduce halo artifacts around objects.

Global Tone Mapping

Applies a uniform adjustment across the image, providing a consistent look without focusing on local contrast. Techniques include:

1. **Reinhard Tone Mapping**
   - **Description**: Aims for realistic results, simulating the human eye's brightness perception.
   - **Advantages**: Produces natural-looking images with balanced contrast and brightness.
   - **Limitations**: May not highlight fine details as effectively.
2. **Drago Tone Mapping**
   - **Description**: Enhances visibility of details in dark areas without compromising brightness in lighter areas.
   - **Advantages**: Retains shadow detail and highlights in high-contrast scenes.

    - **Limitations**: Can lead to desaturation in very bright areas.
3. **Mantiuk Tone Mapping**
    - **Description**: Maximizes perceived detail by enhancing local contrast, making it ideal for detailed HDR applications.
    - **Advantages**: Produces highly detailed images with sharp contrasts.
    - **Limitations**: May introduce artifacts if over-applied and is computationally demanding.

Example Outputs

Global Tone Mapping

Input Images

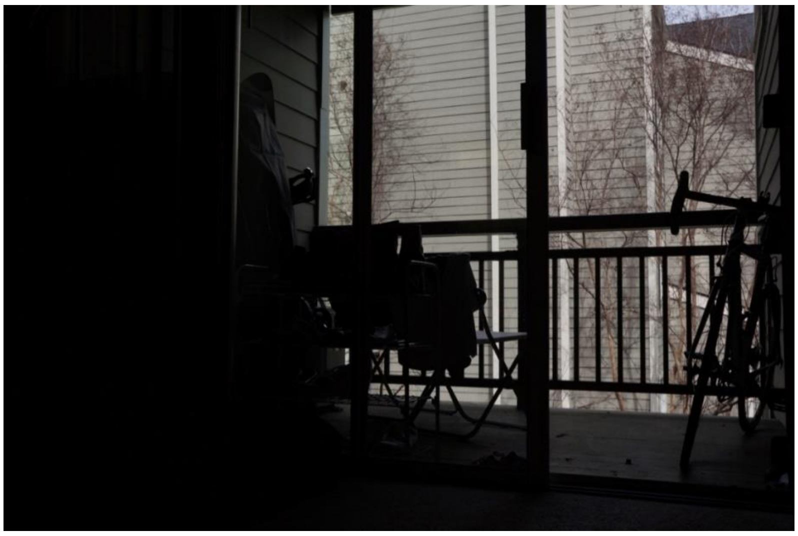
The three input images below were captured at different exposure levels:

- **Image 1**: Underexposed to capture bright highlights.
- **Image 2**: Balanced exposure for midtones.
- **Image 3**: Overexposed to capture shadow details.

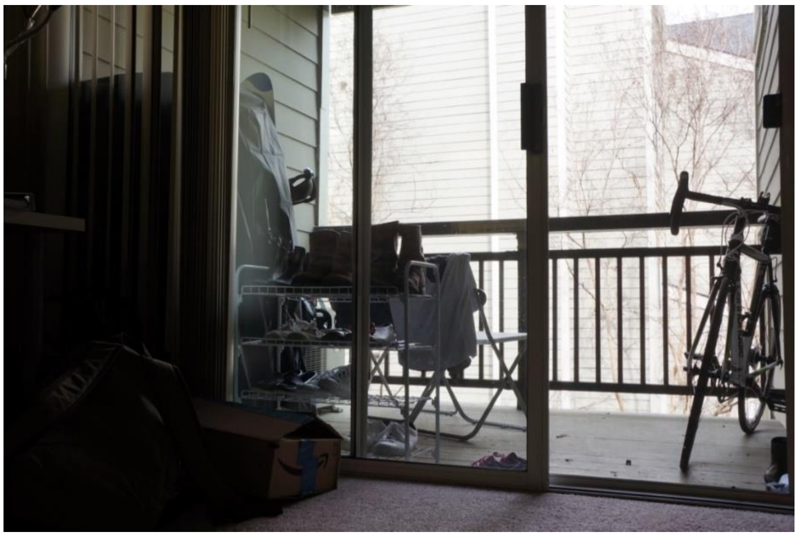
Input Images

Here are the three input images captured at different exposures:

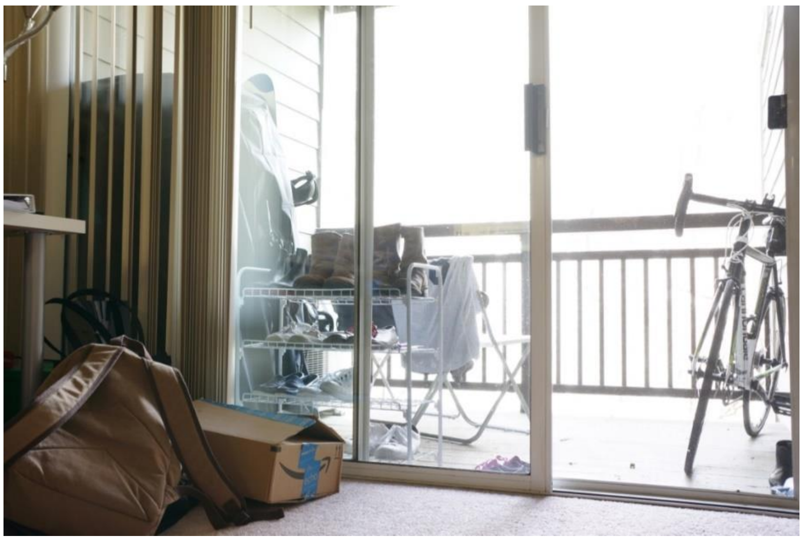
Underexposed



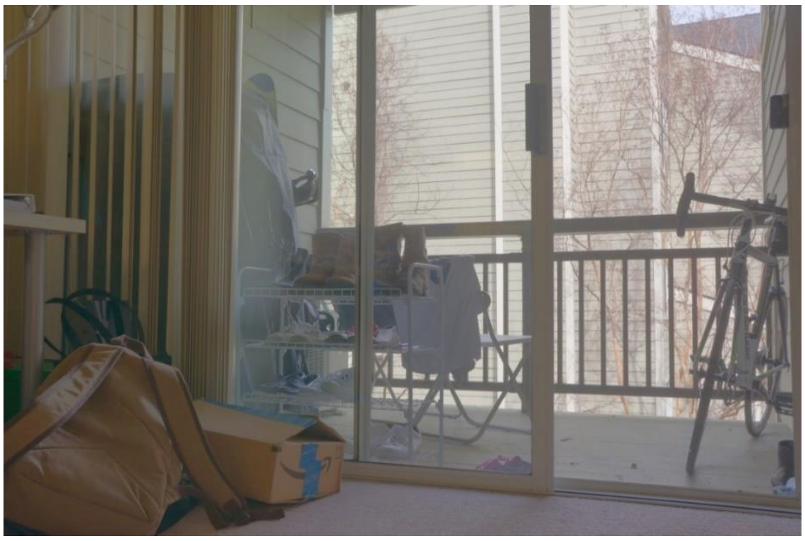
Midtones



Overexposed



Final HDR Output Using Debevec's Merge and Mantiuk Tone Mapping



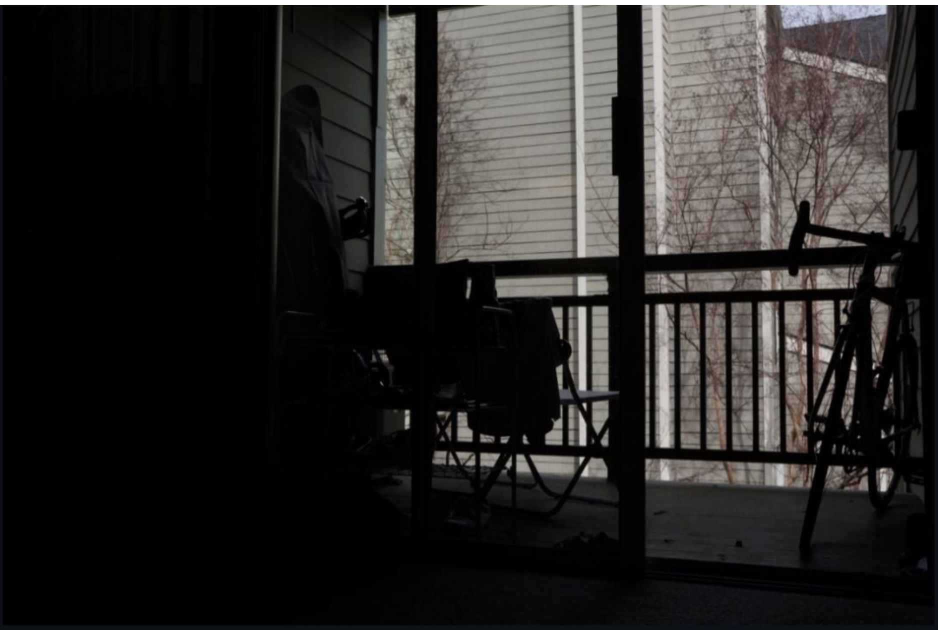
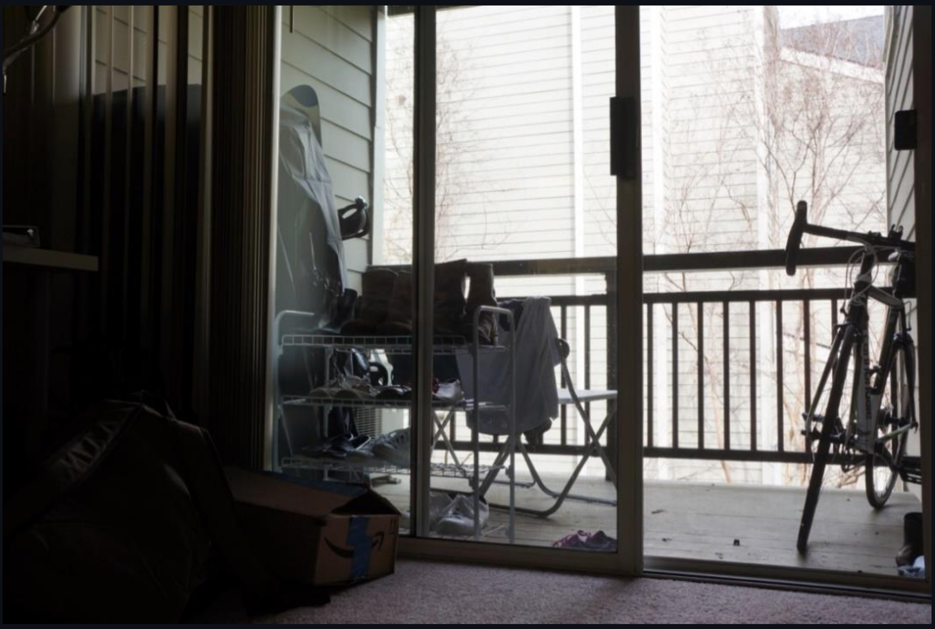
Local Tone Mapping

Input Images

The three input images below were captured at different exposure levels:

- **Image 1**: Underexposed to capture bright highlights.
- **Image 2**: Balanced exposure for midtones.
- **Image 3**: Overexposed to capture shadow details.

- **Input Images**
- Here are the three input images captured at different exposures:
- **Underexposed**

- 
- **Midtones**
- 
- **Overexposed**

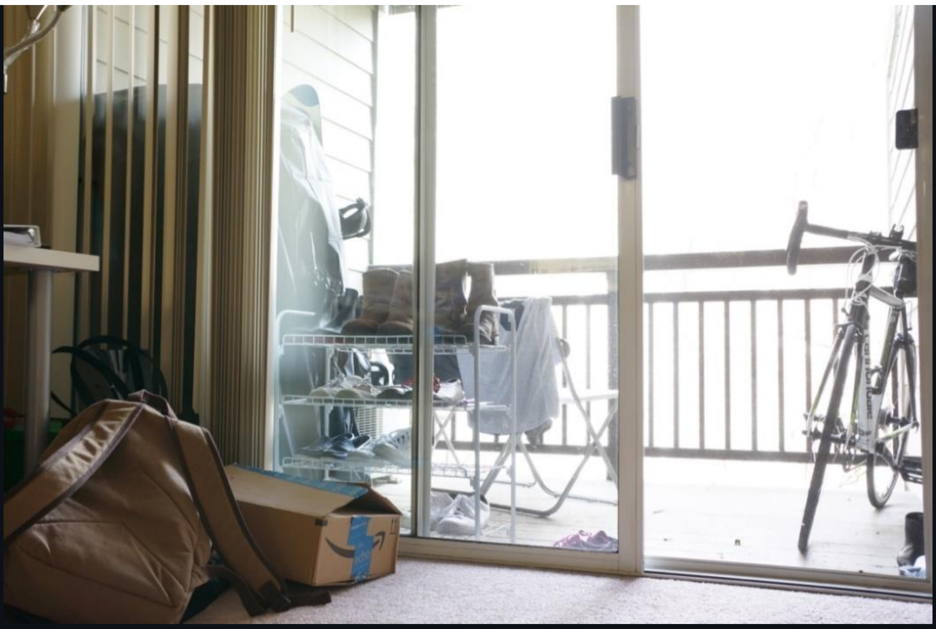
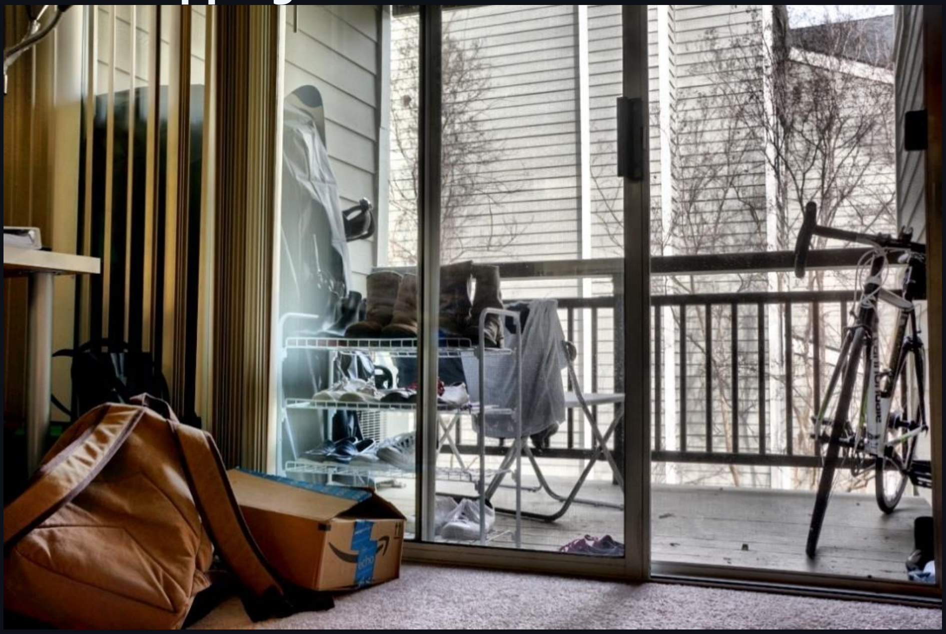
- 
- **Final HDR Output Using Merten's Merge and Local Tone Mapping**
- 
- **Other examples**
- **Example 1**

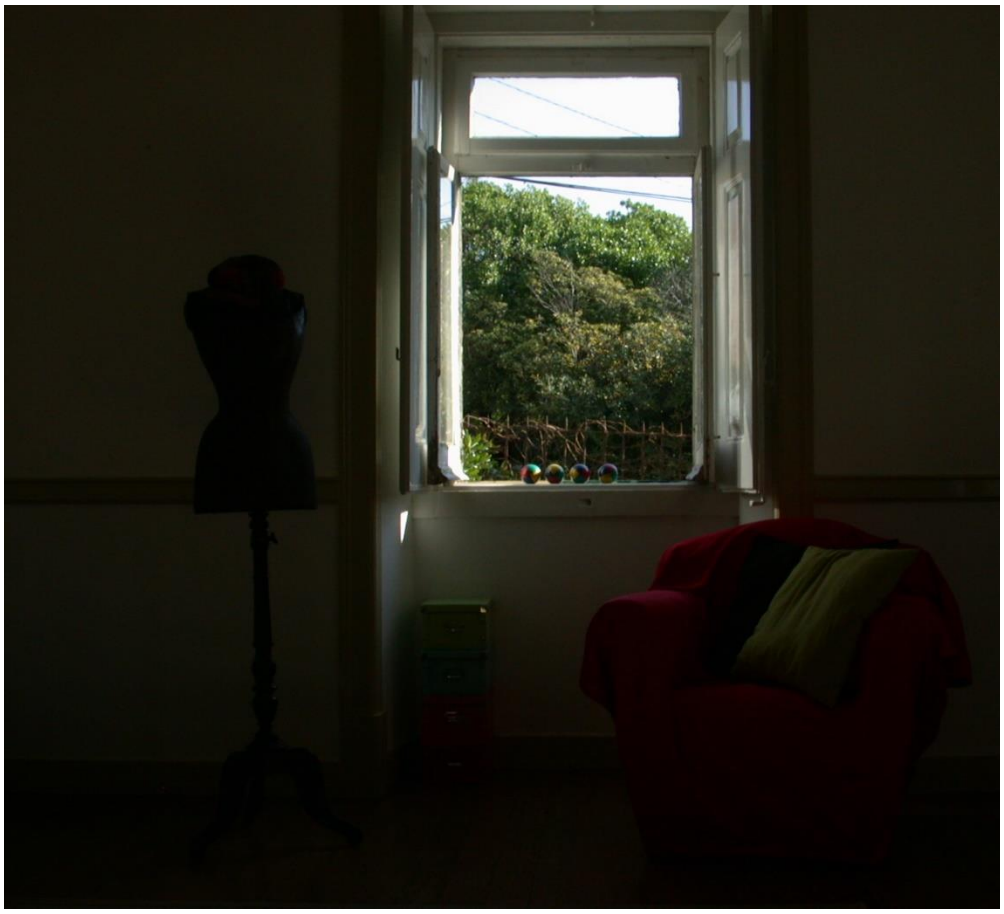
Image Type	**Image**
Low Exposure	

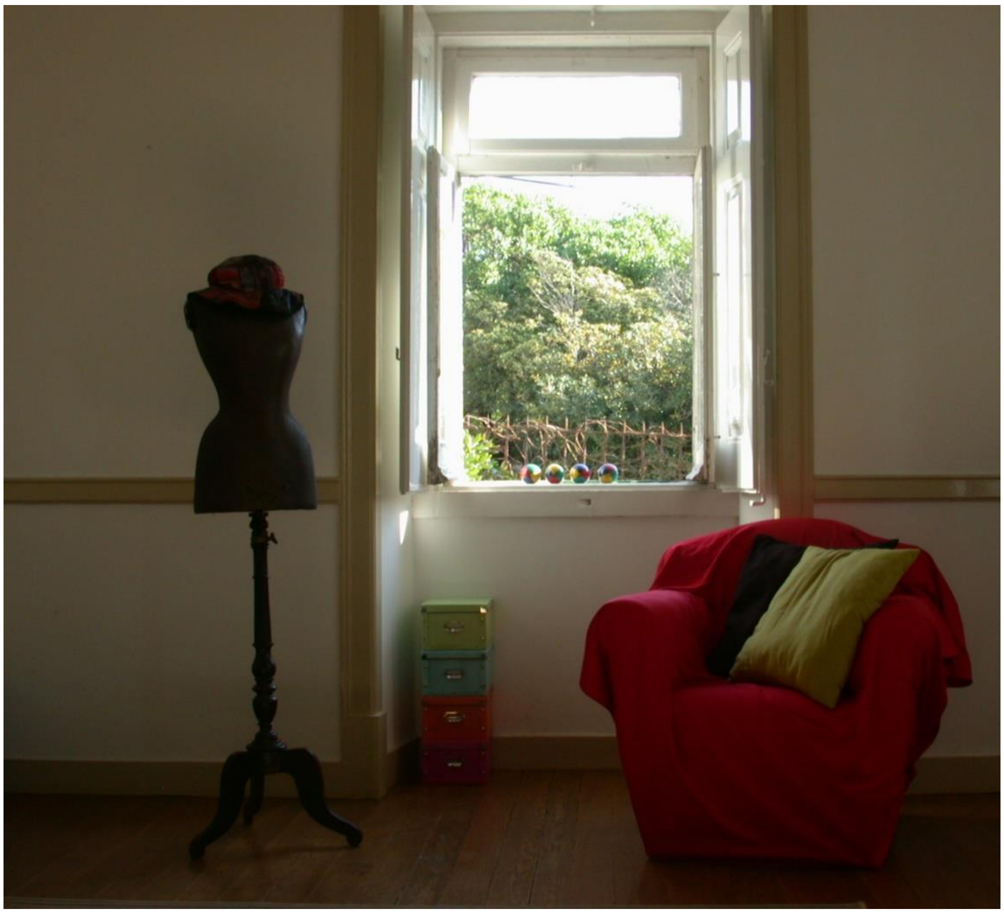
Image Type	**Image**
Mid-Tone	

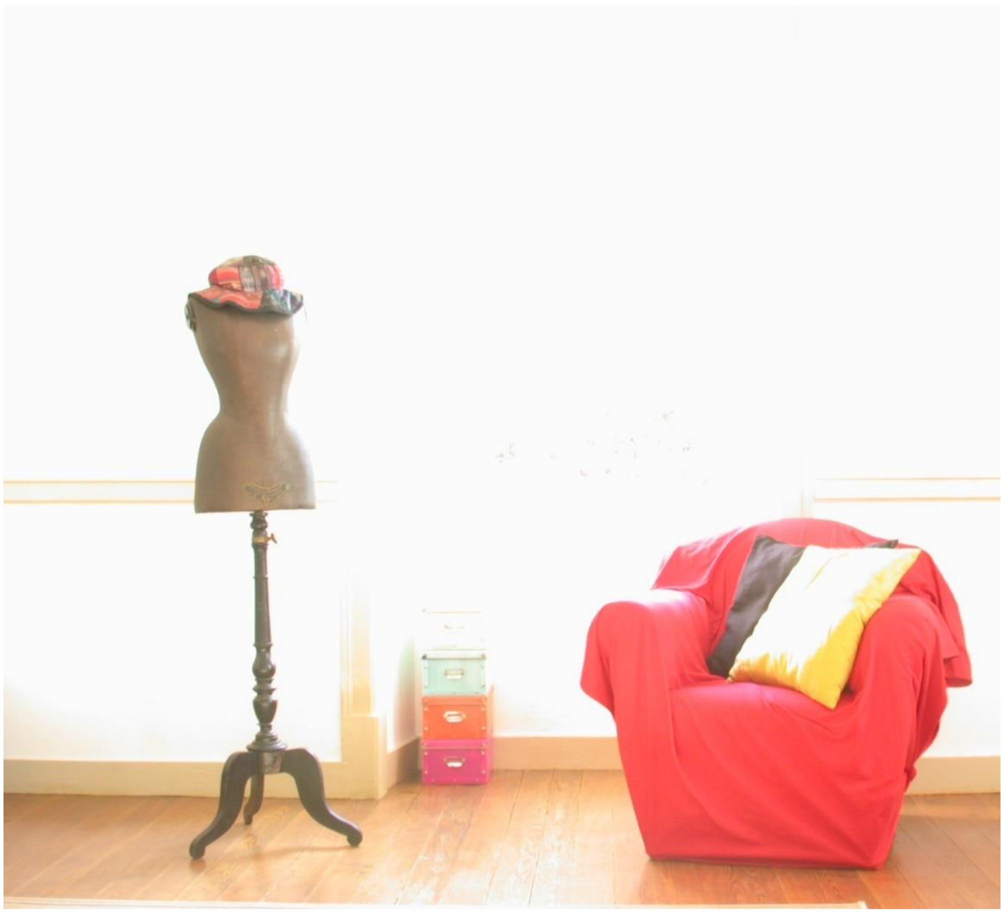
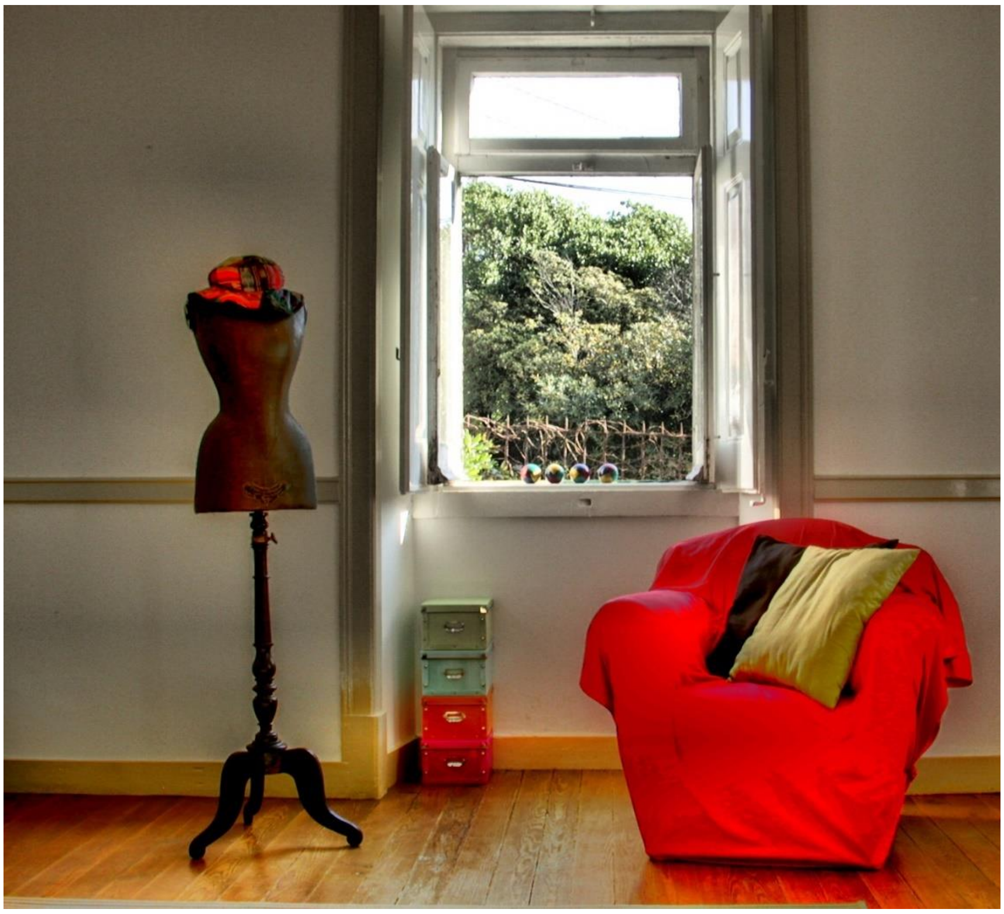
Image Type	**Image**
High Exposure	

Image Type	**Image**
HDR Output	

- **Example 2**

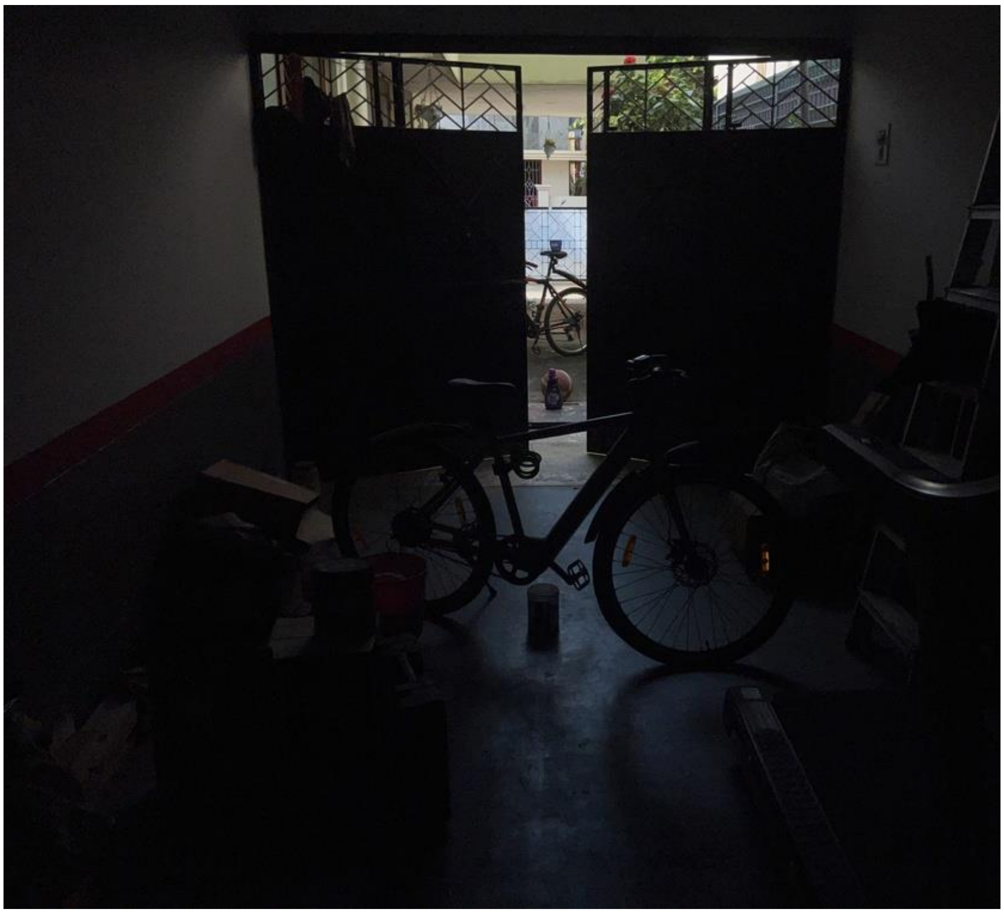
Image Type	**Image**
Low Exposure	

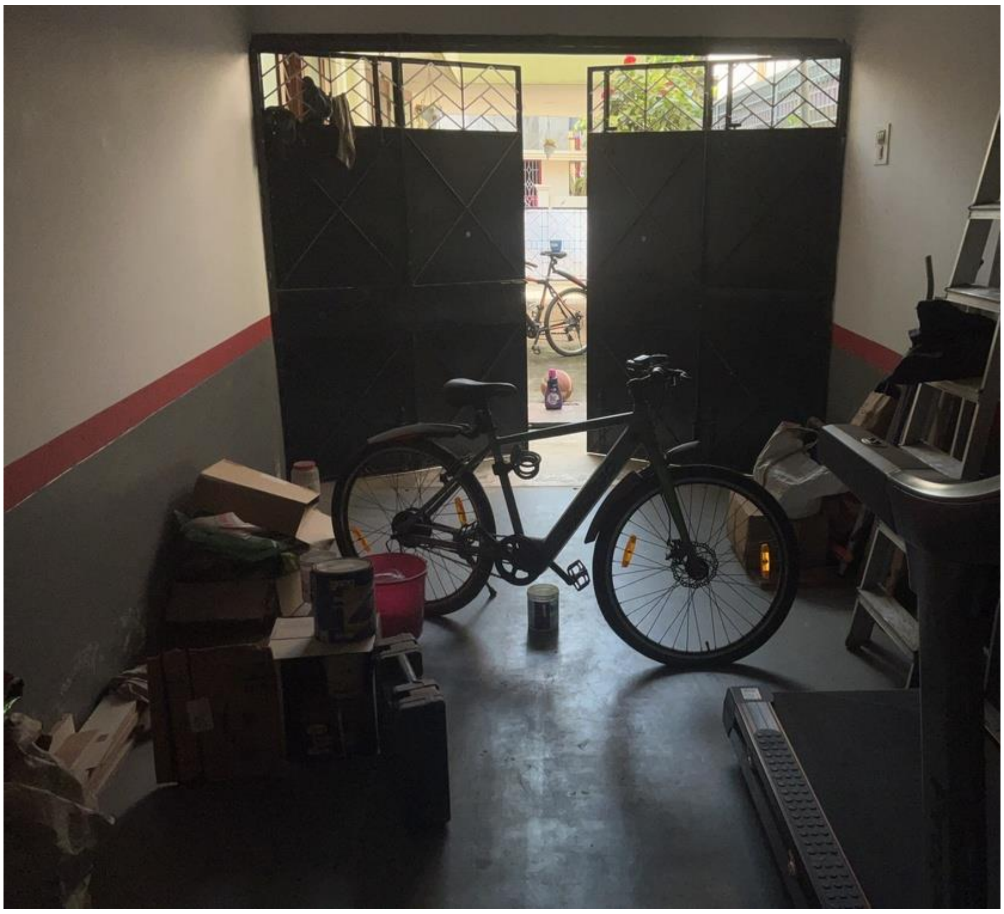
Image Type	Image
Mid-Tone	

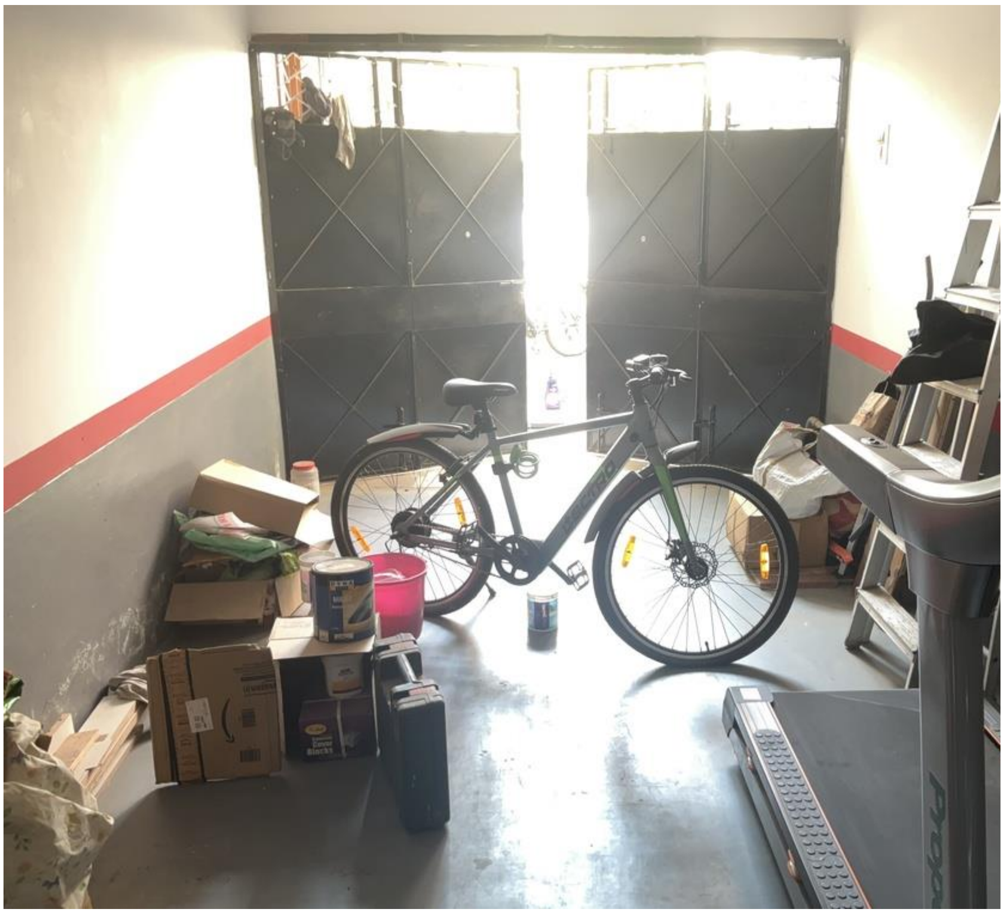
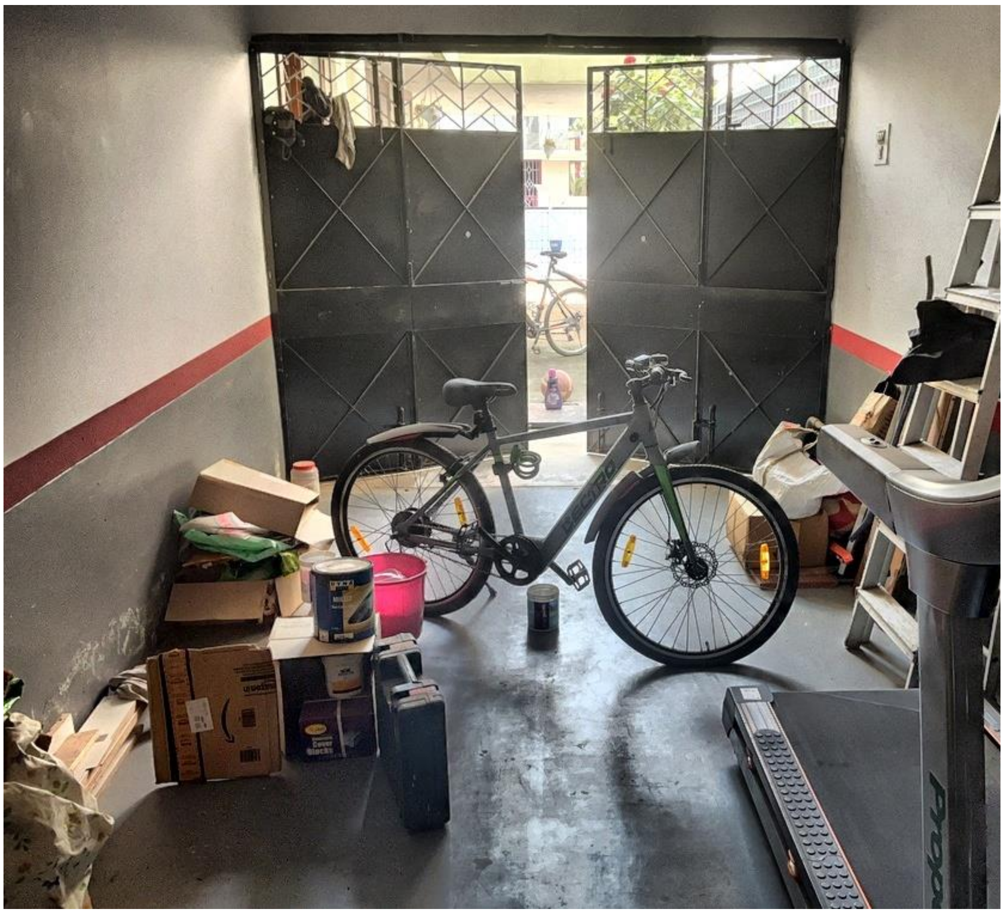
Image Type	**Image**
High Exposure	

Image Type	Image
HDR Output	

- **Example 3**

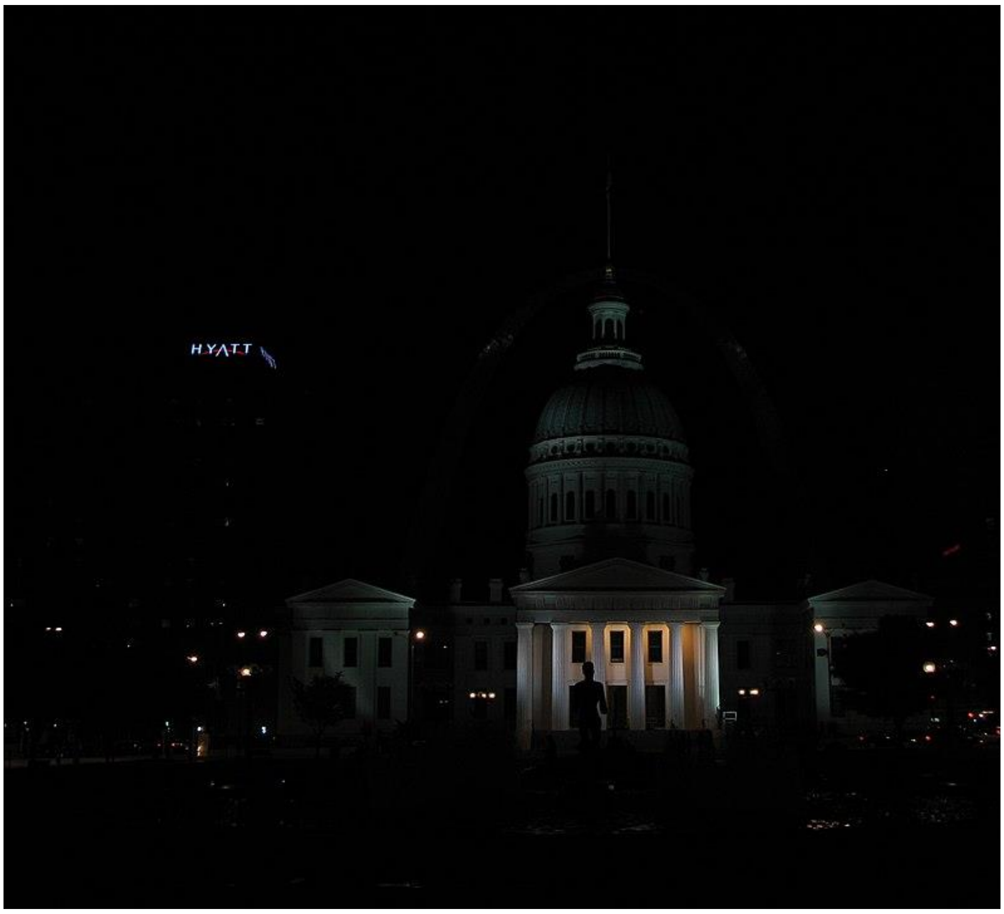
Image Type	**Image**
Low Exposure	

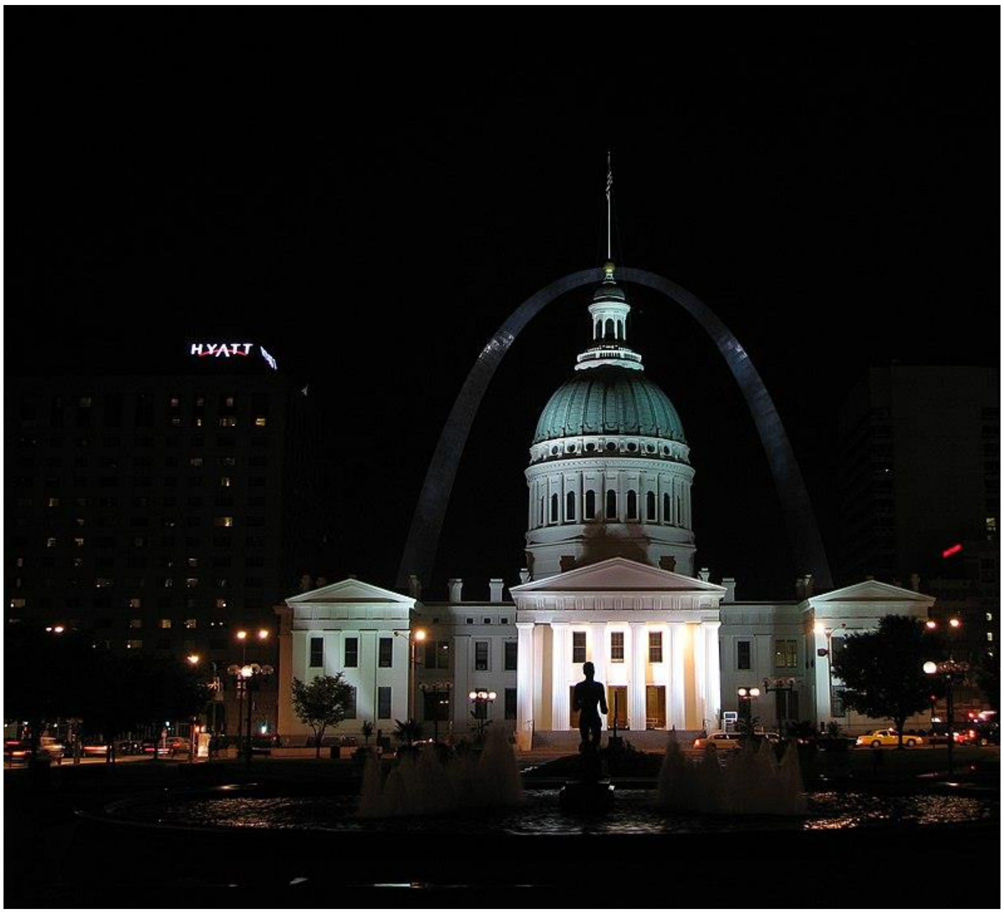
Image Type	Image
Mid-Tone	

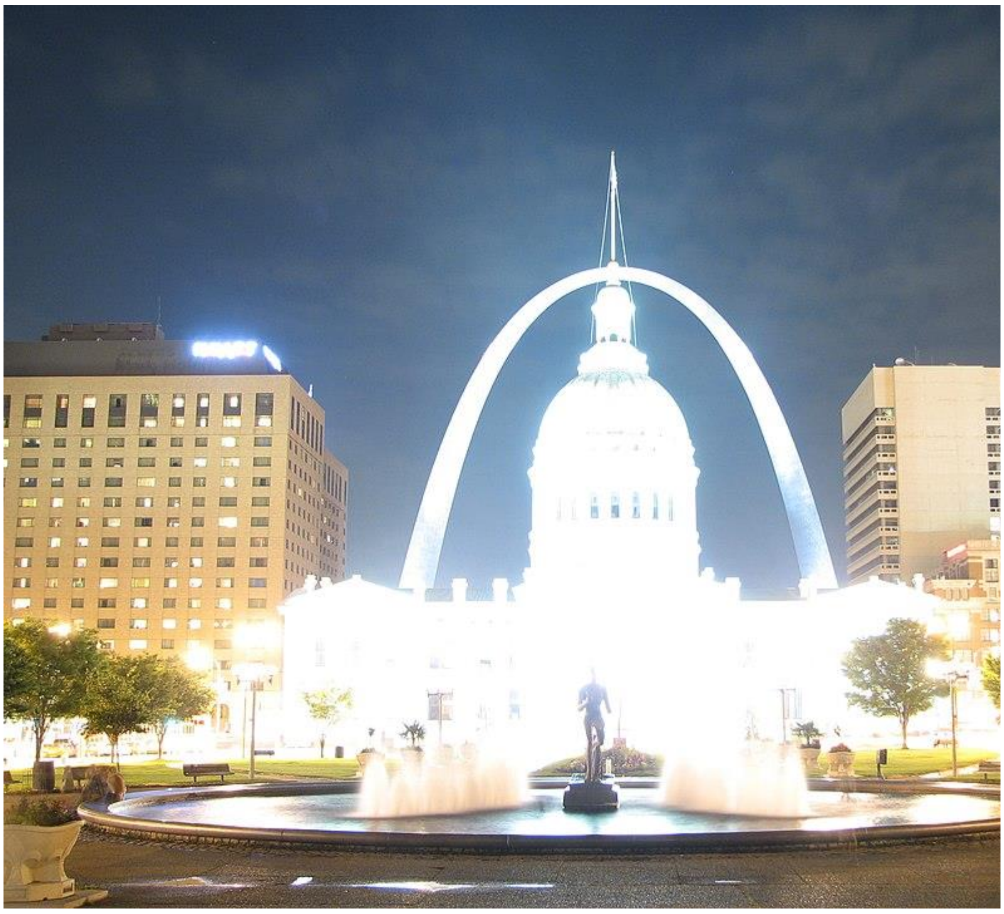
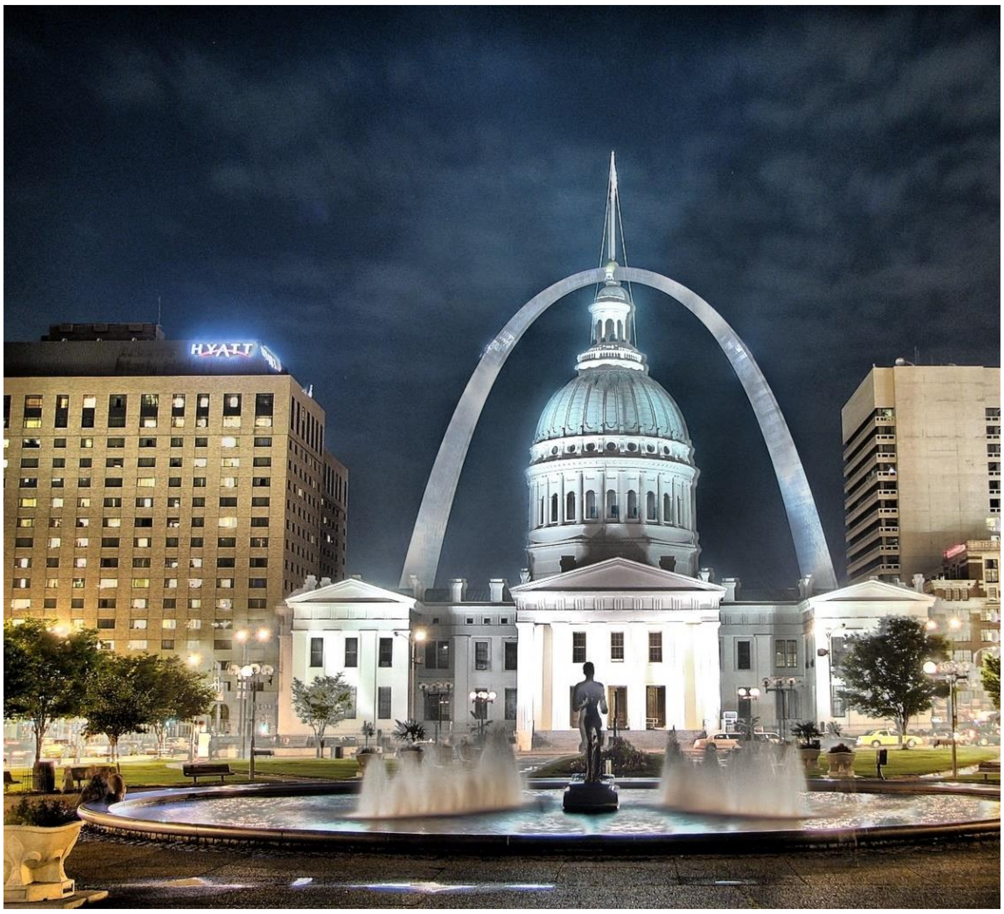
Image Type	**Image**
High Exposure	

Image Type	Image
HDR Output	

- **Example 4**

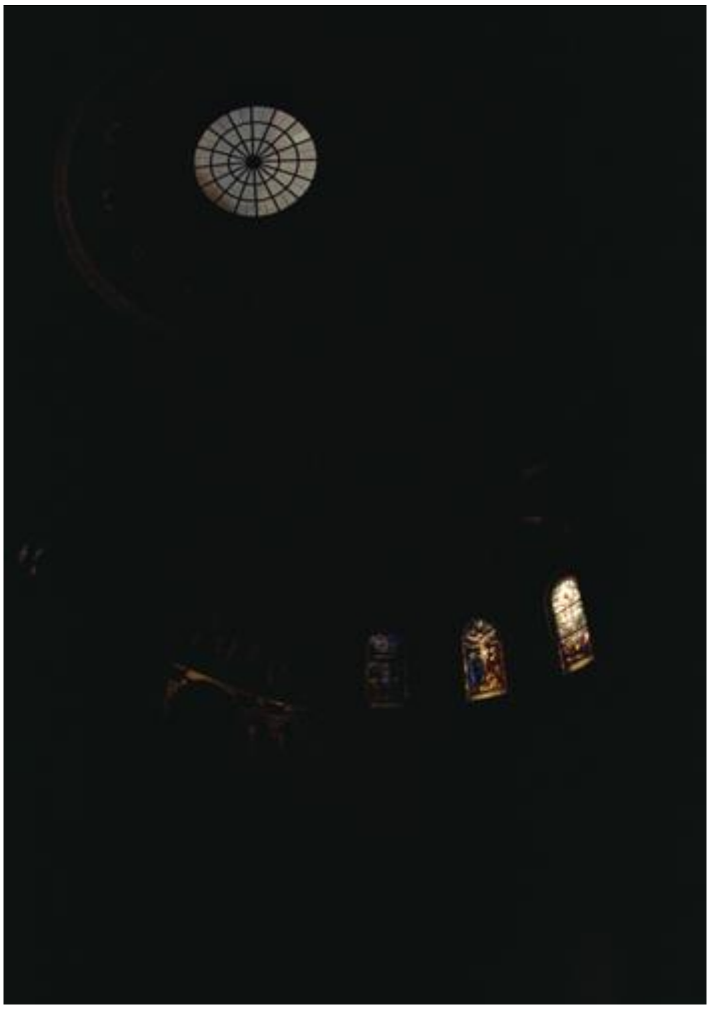
Image Type	**Image**
Low Exposure	

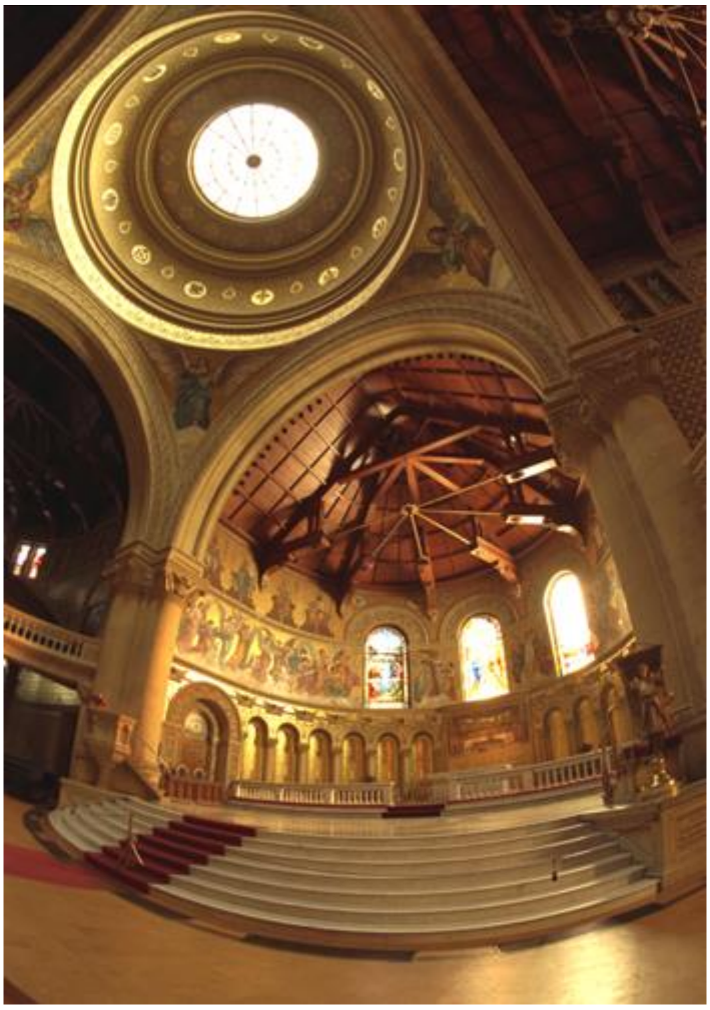
Image Type	**Image**
Mid-Tone	

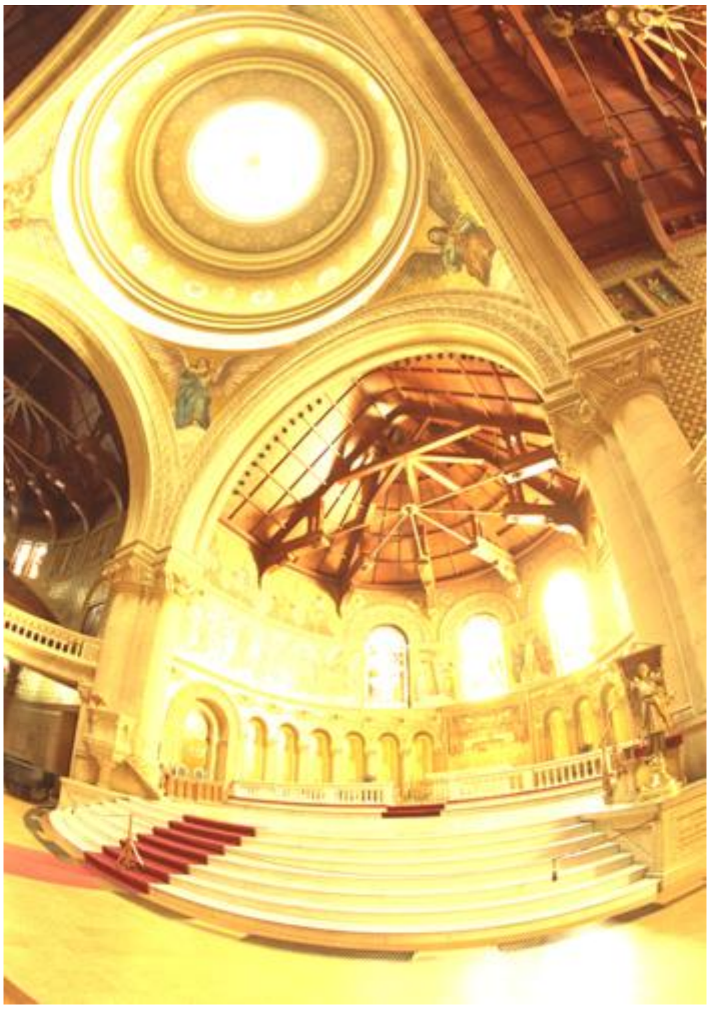
Image Type	Image
High Exposure	

Image Type	Image
HDR Output