Automated Low-Light Car Image Enhancer for Legal and Forensic Applications

Team Members:
Skanta Samvartan R-3122225001137
Vijaya Prasath L S-3122225001159
Vineeth U-3122225001160
Vishal K-3122225001161

Introduction:-

In legal and investigative scenarios, it is crucial to extract and analyze detailed information from images captured under suboptimal conditions, such as low light or nighttime environments. Images with poor lighting often lack clarity, making it challenging to identify key details like vehicle license plates, colors, and models. This project addresses the problem by employing a series of advanced image processing techniques designed to enhance the quality of low-light images, especially for automotive visuals where clarity is critical.

The system utilizes bilateral filtering to reduce noise without losing important edges, gamma correction to adjust the image brightness, and Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve contrast. Additionally, a custom sharpening algorithm is applied to emphasize important edges, making details more prominent. Each of these techniques is tailored to the specific conditions of the input image, such as its noise level, brightness, and edge density, ensuring a targeted enhancement that improves visibility while preserving key features.

By combining these preprocessing and enhancement techniques, the project produces clearer, more detailed images that can be used for legal purposes, such as identifying cars in low-light conditions, extracting number plate details, and improving overall visual clarity for analysis.

Abstract:-

Low-light image processing is a critical challenge in fields such as forensics and law enforcement, where the ability to extract meaningful details from images captured in poor lighting conditions can be crucial. This project introduces a

comprehensive image enhancement system specifically designed for low-light automotive images. The system applies a sequence of advanced preprocessing and enhancement techniques to improve key details like vehicle license plates, colors, and models for better identification.

The approach leverages bilateral filtering to reduce noise without sacrificing important edge information, followed by gamma correction to optimize brightness. Contrast Limited Adaptive Histogram Equalization (CLAHE) is employed to enhance contrast, and a custom sharpening algorithm further improves the visibility of key features. Each technique is tailored dynamically based on the image's noise levels, brightness, and edge density.

The result is a highly effective solution that enhances low-light vehicle images, producing outputs with improved clarity and detail for legal and investigative purposes, enabling better image analysis in challenging lighting conditions.

Filters and Techniques Used:-

The image enhancement techniques and filters we've applied aim to improve the visibility and detail of low-light car images, making them suitable for legal purposes such as identifying key features like number plates and vehicle color. Here’s an explanation of the methods used in your code:

1. Grayscale Conversion

- **Function**: `to_gray_image()`
- **Purpose**: Converts the input image from color (RGB/BGR) to grayscale.
- **Why it's used**: Many image processing techniques like noise estimation and edge detection work better on grayscale images. Converting to grayscale

simplifies the image data and helps focus on intensity variations rather than color.

2. Noise Estimation

- **Function**: `estimate_noise()`
- **Purpose**: Calculates the standard deviation of noise in the image using the Laplacian operator.
- **Why it's used**: Low-light images often contain significant noise due to the camera's increased sensitivity in such conditions. Estimating the noise helps adjust the strength of subsequent noise reduction techniques.

3. Edge Density Calculation

- **Function**: `edge_density()`
- **Purpose**: Uses the Canny edge detection method to compute the density of edges in the image.
- **Why it's used**: Edge density provides information about the presence of strong edges in the image. Images with higher edge density might require different processing strategies, such as different levels of sharpening.

4. Bilateral Filtering (Denoising)

- **Function**: `bilateral_filter()`
- **Purpose**: Reduces noise while preserving edges using bilateral filtering.
- **Why it's used**: Unlike other denoising techniques (like Gaussian blur), bilateral filtering smoothens flat regions while maintaining sharp edges, making it ideal for images where edge details (e.g., car outlines and number plates) are important.
- **Benefits**: Reduces noise in the image without blurring critical details such as edges.

5. Gamma Correction

- **Function**: `gamma_correction()`
- **Purpose**: Adjusts the brightness of the image using gamma correction based on the average brightness.

- **Why it's used**: Gamma correction is applied to modify the overall luminance of the image. Darker images have gamma values greater than 1 to brighten them, while very bright images use gamma values less than 1 to reduce brightness.
- **Benefits**: Improves image visibility by adjusting the brightness appropriately, especially in low-light conditions where the image may appear too dark.

6. Contrast Limited Adaptive Histogram Equalization (CLAHE)

- **Function**: `clahe()`
- **Purpose**: Enhances the contrast of the image by applying CLAHE on the luminance channel of the LAB color space.
- **Why it's used**: CLAHE improves the contrast of images, especially in low-light scenarios, by equalizing the histogram in localized regions of the image. It prevents over-amplification of noise, which is a problem with global histogram equalization.
- **Benefits**: Enhances local contrast and reveals more details in both dark and bright regions of the image.

7. Image Sharpening

- **Function**: `sharpening()`
- **Purpose**: Sharpens the image by combining the original image with a Gaussian-blurred version of itself.
- **Why it's used**: Sharpening improves the clarity of details such as text (e.g., number plates) and the car's edges. Depending on the edge density, the sharpening intensity is adjusted to avoid over-sharpening areas with already strong edges.
- **Benefits**: Highlights important details, making features like number plates and car edges more visible, which is crucial for forensic analysis.

8. Final Image Processing Pipeline

- **Function**: `image_processing()`

- **Purpose**: Applies all the above techniques in sequence to enhance the image.
- **Process**:
    - First, the image is denoised using bilateral filtering.
    - Then, gamma correction adjusts the brightness based on the average luminance.
    - CLAHE improves contrast, especially in low-light regions.
    - Finally, sharpening highlights important details and edges in the image.
- **Why it's used**: The combination of these techniques ensures that the image is clearer, with enhanced brightness, contrast, and detail, making it suitable for applications like number plate recognition and identifying vehicle features under poor lighting conditions.

Sample Input/Output Images:

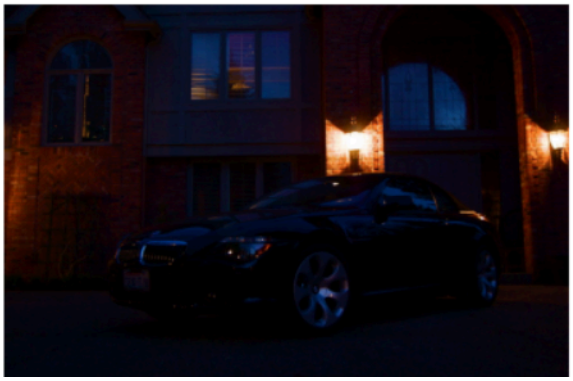
Original Image

Processed Image

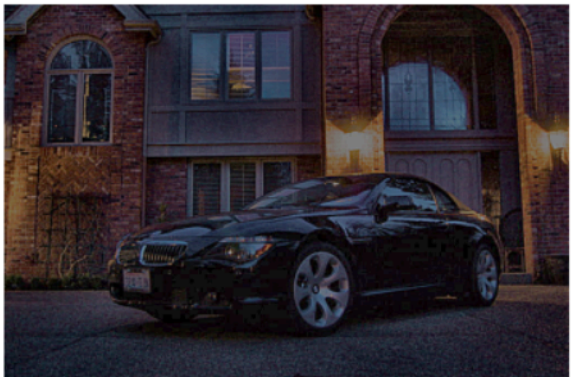
Original Image

Processed Image

Original Image



Processed Image



Original Image

Processed Image

Original Image

Processed Image

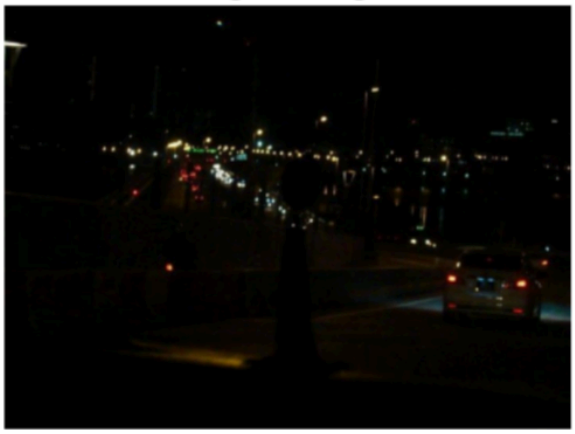
Original Image

Processed Image

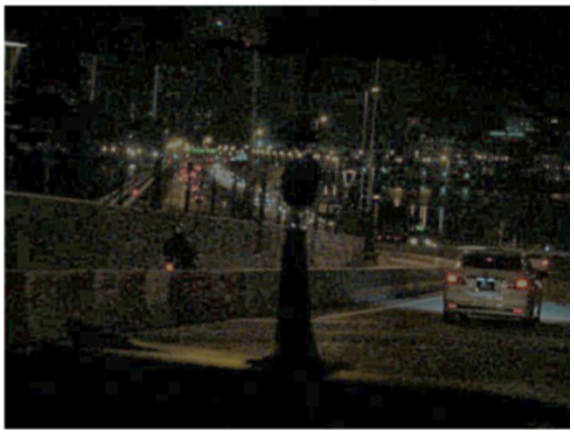
Original Image

Processed Image

Original Image



Processed Image



Original Image

Processed Image

Original Image

Processed Image

---

lmage Processing Pipeline for Car Images: Number Plate Detection and Car Color Extraction

The provided code processes car images to detect number plates and extract the car's color. It utilizes OpenCV for image manipulation and Tesseract OCR for extracting text from detected number plates.

The number plate detection uses a Haar Cascade classifier (haarcascade_russian_plate_number.xml). The image is converted to grayscale, and the classifier identifies the number plate region. OCR is applied to extract the text.

For car color extraction, the image is converted to the HSV color space. K-means clustering is used to identify the dominant color, which is then mapped to a readable color name such as "Red," "White," or "Blue." A helper function handles this mapping based on HSV values.

The main processing function enhances the image and applies both number plate and color extraction. Images are processed in batches using the process_images_in_folder function, which iterates over all image files in the specified folder, displaying the results for each image.

The code is designed to work in Google Colab, using cv2_imshow for displaying images since cv2.imshow is not supported in the notebook environment. The image enhancement function, assumed to be defined separately, further improves the quality of low-light images.

Sample Images:-

To access the dataset used for this project.Use,through the following link:
https://drive.google.com/drive/folders/1uFlFfxgsJqoMkmn0GFkXgT_wT3-HI19R?usp=sharing