

An Industrial Oriented Mini Project Report
On

**ADAPTIVE IMAGE ENHANCEMENT TECHNIQUES FOR
SUPERIOR VISUAL QUALITY IN PHOTOGRAPHIC AND
SURVEILLANCE APPLICATIONS**

Submitted to Kommuri Pratap Reddy Institute of Technology for the partial fulfillment of
the Requirement for the Award of the Degree of

**Bachelor of Technology
in
Computer Science and Engineering (AI&ML)**

By

M. HARSHA VARDHAN REDDY (22RA1A6628)

Under the guidance of
MRS.G.SUJATHA
M.Tech.,Research Scholar (Ph.D.)
Assistant Professor,Dept.OfCSE(AI&ML)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI&ML)
KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(UGC Autonomous Institution,Affiliated to JNTUH ,Hyderabad)

Ghanpur (V), Ghatkesar (M), Medchal (D)-500088

2024-2025

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(UGC Autonomous Institution, Affiliated to JNTUH, Hyderabad)

Ghanpur (V), Ghatkesar (M), Medchal (D)-500088

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)



CERTIFICATE

This is to certify that the dissertation entitled “**ADAPTIVE IMAGE ENHANCEMENT TECHNIQUES FOR SUPERIOR VISUAL QUALITY IN PHOTOGRAPHIC AND SURVEILLANCE APPLICATIONS**” being submitted by **M.HARSHA VARDHAN REDDY (22RA1A6628)** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering (AI&ML) to the Kommuri Pratap Reddy Institute of Technology is a record of confined work carried out by them under my guidance and supervision.

Guide's signature

Mrs. G. SUJATHA

M.Tech., Research Scholar (Ph.D.)

Assistant Professor

Dept. of CSE (AI&ML)

Signature of the Head of the Dept.

Dr. PULIME SATYANARAYANA

M.Tech., Ph.D.

Associate Professor

Dept. of CSE (AI&ML)

Co-Ordinator

Mr. A. VEERABABU

Assistant Professor

Dept. of CSE (AI&ML)

External Examiner

Place: Ghanpur

Date:

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(UGC Autonomous Institution, Affiliated to JNTUH, Hyderabad)

Ghanpur (V), Ghatkesar (M), Medchal (D)-500088

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI&ML)



DECLARATION

I, **M.HARSHA VARDHAN REDDY (22RA1A6628)** here by declare that the project report titled **“IMAGE ENHANCEMENT”** under the guidance of **Mrs.G. SUJATHA**, Assistant Professor, Kommuri Pratap Reddy Institute of Technology, Ghanpur, is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AI&ML).

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this thesis have not been submitted to any other University and Institute for the award of any Degree or Diploma.

SUBMITTED BY

M. HARSHA VARDHAN REDDY (22RA1A6628)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without acknowledging the individuals who made it possible.

I am extremely indebted to our internal guide, **Mrs.G.SUJATHA,Mtech.,Research Scholar(Ph.D.)** Assistant Professor, for her valuable help. I would like to express my profound gratitude to her for providing me the opportunity to work under his supervision and guidance. Her guidance helped us in the successful completion of my project.

I would like to express my heart felt gratitude to our **HoD , Dr. PULIME SATYANARAYANA, M.Tech.,Ph.D** Associate Professor, for providing the guidance and necessary infrastructure to complete this project.

I would also like to express my deep-felt gratitude to our DIRECTOR, **Dr. B. Sudheer Prem Kumar,M.Tech.,Ph.D** of Kommuri Pratap Reddy Institute of Technology, for his kind cooperation and valuable suggestions.

I would also like to express my deep-felt gratitude to **Dr.SRINATH KASHYAP, M.Tch.,Ph.D.,** PRICIPAL of Kommuri Pratap Reddy Institute of Technology, for his kind cooperation and valuable suggestions.

This Acknowledgement will be incomplete without mentioning our sincere gratefulness to our honorable CHAIRMAN **Shri. KOMMURI PRATAP REDDY**, and VICE-CHAIRMAN **Shri KOMMURI PRASHANTH REDDY** of Kommuri Pratap Reddy Institute of Technology for providing the necessary infrastructure to complete this project.

I also thank the staff of the CSE(AI&ML) department who helped us morally for the successful completion of this project.

SUBMITTED BY

M. HARSHA VARDHAN REDDY (22RA1A6628)

VISION OF THE INSTITUTE

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the nation

MISSION	STATEMENT
IM₁	To have holistic in curriculum and pedagogy through industry interface to meet the needs of global competency
IM₂	To develop students with knowledge attitude employability skills entrepreneurship research potential and professionally ethical citizens
IM₃	To contribute to advancement of Engineering & Technology that would help to satisfy the societal needs
IM₄	To preserve promote cultural heritage humanistic values and spiritual values thus helping in peace and harmony in the society

VISION OF THE DEPARTMENT

To provide quality education in computer science for the innovative professionals to work for the development of the nation

MISSION OF THE DEPARTMENT

MISSION	STATEMENT
DM1	Laying the path for rich skills in computer science through the basic knowledge of mathematics and fundamentals of engineering
DM2	Provide latest tools and technology to the students as a part of learning infrastructure
DM3	Training the students toward employability and entrepreneurship to meet the societal needs.
DM4	Grooming the students with professional and social ethics.

PROGRAM EDUCATIONAL OBJECTIVES(PEOS)

PEO'S	STATEMENT
PE01	Demonstrate technical skills, competency in AI&ML and exhibit team management capability with proper communication in a job environment.
PE02	Support the growth of economy of a country by starting enterprise with a lifelong learning attitude.
PE03	Carry out research in the advanced areas of AI&ML and address the basic needs of the society.
PE04	Carry out research in the advanced areas of AI&ML and address the basic needs of the society.

PROGRAM OUTCOMES

PO1	Engineering Knowledge: Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex Engineering problems.
PO2	Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

ABSTRACT

Image enhancement is a cornerstone of digital image processing, vital for improving visual quality across various applications, including photography and surveillance. Traditional methods such as filtering-based enhancement have long been used to adjust image attributes like contrast and noise. However, these methods often fall short in preserving fine details and adapting to diverse image characteristics, particularly in challenging environments with high dynamic ranges or low-light conditions. This study proposes an advanced approach using Adaptive Histogram Equalization (AHE). These techniques divide images into smaller tiles for localized contrast adjustments, ensuring balanced enhancements across both bright and dark regions. By incorporating a contrast-limiting parameter, It addresses noise amplification issues, delivering superior results compared to conventional filtering methods. Comprehensive evaluations using metrics like Peak Signal to Noise Ratio (PSNR), Mean Square Error (MSE), and Structural Similarity Index Metric (SSIM) demonstrate the efficacy of the proposed method in retaining details and improving visual quality. The findings underscore the potential of AHE in applications demanding high-quality image outputs, such as professional photography and advanced surveillance systems. The proposed approach not only addresses the limitations of traditional methods but also introduces a robust framework for real-time image enhancement in modern multimedia applications.

TABLE OF CONTENTS

ABSTRACT	x
1. INTRODUCTION	1
1.1 OVERVIEW	1
1.2 RESEARCH MOTIVATION	2
1.3 PROBLEM DEFINATION	2
1.4 SIGNIFICANE.....	3
1.5 RESEARCH OBJECTIVE	4
1.6 ADVANTAGES.....	5
1.7 APPLICATIONS.....	6
2. LITERATURE SURVEY	7
3. EXISITING SYSTEM.....	9
3.1 MEDIAN FILTER BASED ENHANCEMENT	9
3.2 DRAWBACKS.....	11
4. PROPOSED SYSTEM	12
4.1 HYBRID ADAPTIVE HE.....	12
4.2 CLAHE APPROACH	15
4.3 ADVANTAGES	17
5. UML DIAGRAMS	19
5.1 CLASS DIAGRAM	19
5.2 SEQUENCE DIAGRAM	20
5.3 ACTIVITY DIAGRAM	21
5.4 DEPLOYMENT DIAGRAM	21
5.5 USE CASE DIAGRAM	22
5.6 COMPONENTDIAGRAM.....	23
5.7 DATA FLOW DIAGRAM.....	23
5.8 SYSTEM ARCHITECTURE	24
6. REQUIREMENTS	25
6.1 SOFTWARE REQUIREMENTS	25
6.2 HARDWARE REQUIREMENTS	33

7. FUNCTIONALREQUIREMENTS	35
8. SOURCECODE	39
9. RESULTS AND DISCUSSIONS.....	47
9.1 IMPLEMENTATION DESCRIPTION.....	47
9.2 EXISTING RESULTS	48
9.3 PROPOSED RESULTS	49
10. CONCLUSION.....	54
10.1 CONCLUSION	54
10.2 FUTURE SCOPE	54
11. REFERENCES	55

LIST OF FIGURES

FIGURE 3.1 EXISTING BLOCK DIAGRAM.....	10
FIGURE 4.1 PROPOSED SYSTEM ARCHITECHURE.....	14
FIGURE 4.2 PROPOSED CLAHE FLOWCHART	16
FIGURE 5.1 CLASS DIAGRAM	20
FIGURE 5.2 SEQUENCE DIAGRAM	20
FIGURE 5.3 ACTIVITY DIAGRAM	21
FIGURE 5.4 DEPLOYMENT DIAGRAM	22
FIGURE 5.5 USE CASE DIAGRAM	22
FIGURE 5.6 COMPONENT DIAGRAM.....	23
FIGURE 5.7 DATA FLOW DIAGRAM	24
FIGURE 5.8 SYSTEM ARCHITECTURE	24

LIST OF SCREENSHOTS

SCREENSHOT 6.1 PYTHON URL.....	28
SCREENSHOT 6.2 PYTHON 3.7.6 SETUP	29
SCREENSHOT 6.3 ADD PYTHON 3.7 TO PATH	29
SCREENSHOT 6.4 INSTALL PYTHON	30
SCREENSHOT 6.5 COMMAND PROMPT	31
SCREENSHOT 6.6 EXIT COMMAND PROMPT	31
SCREENSHOT 6.7 INSTALL PACKAGES.....	31
SCREENSHOT 9.1 EXISTING CURRENCY NOTE IMAGE	48
SCREENSHOT 9.2 PSNR,MSE,SSIM GRAPHS FOR VARIOUS LIMITS.....	49
SCREENSHOT 9.3 EXISTING CCTV IMAGE	49
SCREENSHOT 9.4 PROPOSED AHE RESULTS ON CURRENCY NOTE	50
SCREENSHOT 9.5 PROPOSED PSNR,MSE,SSIM GRAPHS FOR VARIOUS LIMITS.....	51
SCREENSHOT 9.6 PROPOSED AHE RESULTS ON CURRENCY NOTE FOR VARIOUS LIMITS.....	51
SCREENSHOT 9.7 PROPOSED AHE RESULTS ON CCTV IMAGE	52
SCREENSHOT 9.8 PROPOSED AHE RESULTS ON CCTV IMAGE FOR VARIOUS LIMITS	52

1. INTRODUCTION

1.1 Overview

In the modern digital era, the demand for high-quality visual content has soared significantly due to the explosive growth in image-based communication, e-commerce, social media platforms, and visual analytics. Statistically, more than 1.8 billion images are uploaded every day across digital platforms, reflecting the crucial need for consistent and visually pleasing imagery. However, the visual quality of these images can be degraded due to various environmental factors such as poor lighting conditions, low contrast, noise, or motion blur. The degradation is even more prominent in fields like surveillance and security monitoring, where lighting conditions are rarely controllable. Hence, image enhancement techniques have emerged as critical tools for improving visibility, interpretability, and visual fidelity of digital images.

Photographic and surveillance imaging systems play a pivotal role in numerous sectors such as journalism, law enforcement, autonomous vehicles, and healthcare diagnostics. Surveillance cameras, for instance, are expected to perform consistently across varying lighting conditions such as nighttime or foggy environments. According to recent surveys, over 85% of cities globally rely on video surveillance systems for public safety, yet more than 60% of these systems record under suboptimal lighting, leading to degraded footage. These conditions necessitate robust adaptive enhancement methods to restore or elevate image quality dynamically. Enhancing image clarity can directly improve decision-making processes in areas such as facial recognition, vehicle tracking, and threat detection.

Additionally, consumer expectations for photographic clarity and vibrancy continue to grow. Mobile devices are now the primary tools for photography, and over 92% of all photographs are taken on smartphones, which often struggle with varying dynamic ranges and lighting scenarios. In such cases, image enhancement algorithms can significantly improve user satisfaction by correcting underexposed or overexposed regions, balancing color distributions, and ensuring that fine details are retained or accentuated. As more applications demand real-time enhancement with minimal computational cost, adaptive methods become essential for balancing visual quality and performance efficiency.

1.2 Research Motivation

Organizations today rely heavily on visual data for strategic operations, safety, analytics, and quality control. For example, retail giants like Amazon and Walmart use surveillance systems not only for security but also for customer behavior analysis and loss prevention. However, when visual data from cameras lacks clarity due to poor lighting or harsh environmental conditions, valuable information is lost. Adaptive image enhancement methods are necessary to retrieve meaningful data from such low-quality imagery. Without clear footage, analytics such as facial recognition or customer heatmaps are prone to errors, leading to faulty insights. This underscores the urgency for dynamic and efficient enhancement techniques that maintain visual consistency in fluctuating environmental settings.

Similarly, healthcare imaging companies and diagnostic centers depend on medical images such as X-rays, MRIs, and retinal scans, where contrast and detail visibility are paramount for accurate diagnostics. Companies like Siemens Healthineers and GE Healthcare integrate advanced image processing to improve medical image readability. In low-light scenarios or when capturing images from portable scanning equipment, essential features might appear blurred or washed out. Here, adaptive enhancement can amplify minute details, aiding radiologists in making accurate diagnoses and reducing the need for repeated scans. Reliable data analysis in such domains is directly correlated with patient outcomes and treatment efficiency.

In the field of autonomous driving, companies such as Tesla, Waymo, and NVIDIA utilize a combination of visual cameras and LIDAR systems for real-time environmental understanding. During dusk, nighttime, or fog, camera sensors can struggle to maintain visibility. If the onboard systems receive low-contrast images, object detection and road segmentation tasks can fail, potentially leading to accidents. Adaptive image enhancement allows the on-board vision systems to preprocess camera input and ensure consistently interpretable frames, contributing to the safety and operational accuracy of autonomous systems. Real-time enhancement is no longer a luxury but a necessity in safety-critical domains that rely on consistent visual perception.

1.3 Problem Definition

Low contrast and uneven illumination are persistent challenges in photographic and surveillance images. The core problem lies in the inability of traditional static enhancement techniques to adapt to diverse environmental conditions and image characteristics. In

scenarios such as nighttime surveillance, backlight photography, or foggy road captures, the visibility of essential features such as facial details, license plates, or object boundaries becomes compromised. Standard histogram equalization often leads to over-enhancement or information loss due to global processing without regard to local image structure.

Moreover, many existing enhancement techniques are not computationally efficient or scalable for real-time deployment, especially on resource-constrained devices like smartphones, drones, or embedded surveillance systems. As a result, these systems may either underperform or delay crucial operations like face detection, intruder alerts, or scene reconstruction. There is a strong need for techniques that can dynamically adapt to local brightness variations and amplify features selectively without introducing noise or distortion.

In addition, the quality of visual content has a direct impact on machine vision and artificial intelligence (AI) applications such as object tracking, behavior analysis, and emotion recognition. Poor-quality input images can mislead algorithms, resulting in incorrect classifications or failed detections. These inaccuracies are particularly critical in law enforcement, industrial inspection, and autonomous navigation. Therefore, addressing the problem of poor contrast and low-detail images with adaptive and context-aware enhancement strategies is essential for downstream application reliability.

1.4 Significance

Image enhancement serves as the foundation for a multitude of downstream vision applications ranging from personal photography to public surveillance. Improving image clarity enhances not only human visual perception but also the accuracy and reliability of automated systems. Given the widespread use of cameras in safety-critical sectors like healthcare, smart cities, and autonomous navigation, even marginal improvements in image quality can have disproportionately positive effects on performance and safety.

Furthermore, adaptive enhancement techniques contribute significantly to digital forensics and evidence validation, where image authenticity and clarity are crucial. In legal and investigative domains, enhanced images can make the difference between identifying a suspect or missing a vital lead. Additionally, in social media and creative industries, superior image quality can enhance viewer engagement and brand perception, thereby improving marketing efficacy and user retention.

From a computational standpoint, enhancement techniques that adapt to local image properties ensure that only the necessary regions are processed intensively, thereby reducing processing time and energy consumption. This characteristic makes them suitable for edge computing environments where resource optimization is essential. Hence, the development and deployment of such adaptive image enhancement techniques is not only technologically significant but also socially and economically impactful.

1.5 Research Objective

The proposed method utilizes Adaptive Histogram Equalization (AHE) to address the limitations of filtering-based methods. AHE enhances image contrast by dividing the image into smaller regions or tiles and applying histogram equalization to each tile individually. This localized approach ensures better contrast adjustment across the entire image, particularly in areas with varying brightness levels. It introduces a contrast-limiting parameter, ensuring a balance between contrast enhancement and noise suppression. This method effectively adapts to diverse image characteristics, delivering superior visual quality in challenging conditions such as low-light environments and high dynamic range scenes.

Proposed Method Advantages: The proposed method offers significant advantages over filtering-based techniques. By focusing on localized regions, AHE ensure that both bright and dark areas of the image are enhanced simultaneously, resulting in a balanced and natural-looking output. The adaptability of this method makes it suitable for a wide range of applications, from photography to surveillance.

Furthermore, the contrast-limiting capability of AHE reduces noise amplification, preserving fine details and ensuring image integrity. This advantage is particularly beneficial in surveillance applications, where accurate detail retention is crucial for object recognition. The proposed method's ability to deliver high-quality results with minimal computational overhead makes it an ideal choice for real-time applications.

Module Split-Up

- **Input Module:** Acquire raw images from photographic or surveillance sources.
- **Existing Module:** Apply filtering-based enhancement techniques. Analyze results in terms of noise suppression.

- **Proposed Module:** Implement Adaptive Histogram Equalization. Perform localized contrast enhancement with noise control.
- **Output Module:** Generate enhanced images with improved visual quality.
- **Performance Evaluation:** Compare results using metrics such as PSNR, MSE, and SSIM.

1.6 Advantages

- Adaptive enhancement methods respond to local brightness levels, allowing for region-specific contrast improvement without global over-saturation or detail loss.
- They offer significant improvements in visual perception under challenging lighting conditions, such as low-light or high-glare environments, commonly encountered in surveillance footage.
- Unlike static methods, adaptive approaches prevent noise amplification in already bright areas, maintaining overall visual integrity.
- These techniques enable better feature extraction for downstream tasks such as object detection and face recognition in AI pipelines.
- In real-time systems, adaptive methods can be optimized to run efficiently on hardware with limited computational power such as mobile processors or embedded boards.
- They help reduce the need for expensive hardware upgrades (e.g., higher-end sensors) by maximizing image utility through software-based enhancements.
- In legal and forensic settings, enhanced images allow better identification and tracking of suspects or objects, increasing the credibility of visual evidence.
- They contribute to reduced human error in monitoring environments, making visual inspections more reliable and less dependent on operator expertise.
- Adaptive enhancement improves user satisfaction in consumer applications such as smartphone photography by automatically balancing dynamic ranges.
- The techniques support a wide range of image formats and camera types, making them versatile and easy to integrate into diverse imaging systems.

1.7 Applications

- Surveillance systems in public transportation hubs such as airports, subways, and railway stations use adaptive enhancement for better visibility during low-light hours.
- Law enforcement agencies enhance night patrol footage and body cam videos for improved evidence gathering and suspect identification.
- Smart city monitoring systems apply image enhancement to ensure uninterrupted and clear surveillance during fog, rain, or nighttime.
- Healthcare facilities utilize adaptive contrast enhancement in radiology for better visibility of soft tissue anomalies in X-ray or ultrasound images.
- Automotive industries integrate enhancement modules into vehicle cameras to improve road visibility in adverse weather for advanced driver-assistance systems.
- Military and defense applications use adaptive enhancement in drone and satellite imagery to identify targets in camouflage or low-contrast terrain.
- News and media companies enhance raw footage taken under poor conditions to deliver clearer visual narratives to audiences.
- Consumer electronics, particularly in smartphone cameras, use adaptive enhancement to deliver vibrant photos in backlit or indoor scenes.
- Wildlife monitoring and conservation efforts employ enhancement tools to capture and analyze nocturnal animal behavior in low-visibility environments.
- Industrial automation and inspection systems improve the accuracy of defect detection in manufacturing by enhancing image contrast and detail clarity.

2. LITERATURE SURVEY

The military, earth sciences, agriculture, and astronomy industries are experiencing a surge in demand for high-quality remote sensing images. Nonetheless, less-than-ideal environmental circumstances reduce the brightness and sequester the critical elements in remote sensing images. Since brightness is a major quality component in remote sensing photos, low-light enhancement techniques are required for improved information representation and visual perception [1].

The two image-enhancement techniques are image spatial domain and transform domain methods. Traditional histogram equalization [2] is the most popular image spatial domain algorithm due to its simplicity and efficiency. However, the primary disadvantage of histogram equalization is that if the histogram contains peaks, the generated results are enhanced, resulting in saturation issues and a highly sharpened image. To overcome this issue, histogram-based methods in the spatial domain of the image, such as dynamic histogram equalization [3] and a histogram modification framework [4], have been proposed. Although both can prevent over-enhancement, the details are not emphasized because these methods preserve the input histogram. Recently, adaptive gamma correction with weighted distribution (AGCWD) [5] has been proposed as a new contrast enhancement technique, which produces similar results and may also cause a loss of detail in light areas and an increase in saturation. A previous study proposed a two-dimensional (2D) histogram that uses contextual information to improve the contrast of the input image [6,7,8]. However, generating a two-dimensional histogram has a high computational cost and is not suitable for many practical applications.

The transform domain method disassembles input images into distinct sub-bands and improves contrast by modifying specific components [6]. In the study by Demirel et al. [9], a singular value equalization method was proposed for adjusting image brightness. The combination of this method with the discrete wavelet transform (Gonzalez et al.) [1] improved the contrast enhancement. Another method, enhancing contrast in remote sensing images with discrete wavelet transforms and adaptive intensity transformations, was proposed by Lee et al. [10]; however, this method requires specific parameter settings, rendering it impractical in the real world. A sub-band decomposition multiscale Retinex method, coupled with a hybrid intensity transfer function, was introduced (Jang et al.) [11] to

improve the optical remote sensing images. Also, a generic method of illumination normalization for multiple remote sensing images was proposed (Zhang et al.) [12]. This method first enhanced the contrast in the gradient domain and then adjusted the brightness by equalizing singular values. However, contrast and details are not emphasized because this algorithm focuses primarily on maintaining illumination consistency.

Lore et al. [13] proposed Low-Light Net (LLNet), a deep autoencoder for contrast enhancement and denoising of low-light images caused by accelerated development of deep learning. The authors of [14] introduced transformative neural networks and argued that the conventional multiscale Retinex algorithm is a feedforward transformative neural network with various Gaussian transformation kernels. Retinex-Net is a deep learning method based on Retinex image decomposition [15], and the entire model is implemented using transformative neural networks. It was also used to establish the Low-Light Paired (LOL) dataset under natural conditions. Some cutting-edge end-to-end methods (Han et al., Zhang et al.) [16,17] used U-Net (Ronneberger et al.) [18] as their fundamental structure and added dense residual blocks to each layer to incorporate multiscale information. Although these methods provide competitive performances on benchmark datasets, their lengthy inference time is not conducive to widespread application.

3. EXISTING SYSTEM

3.1 Median Filter Based Enhancement

Figure 3.1 shows the block diagram of existing median filter and structured approach ensures an effective implementation of median filter-based image enhancement while maintaining balance between noise reduction and image quality preservation.

Step 1: Image Acquisition and Preprocessing: The first step in median filter-based image enhancement involves obtaining the input image, which can be a grayscale or color image. If the image is in color, it is often converted to a grayscale format for easier processing. Preprocessing may involve resizing the image to a standardized size, converting it to a compatible format, and adjusting its intensity values to optimize enhancement. Additionally, noise types such as salt-and-pepper noise, Gaussian noise, or speckle noise are identified to determine the need for filtering.

Step 2: Selection of Neighborhood Window Size: A key decision in applying the median filter is selecting the appropriate neighborhood window size, typically a square-shaped kernel (e.g., 3×3 , 5×5 , or 7×7). The size of the kernel determines the filtering strength, where smaller kernels preserve image details but may not remove noise effectively, while larger kernels provide stronger noise reduction but risk blurring fine details. The window size is chosen based on the noise level in the image, with adaptive selection strategies sometimes used to balance noise removal and detail preservation.

Step 3: Sliding Window Application: The median filter operates by applying a sliding window approach across the entire image. The kernel is centered on each pixel, and the surrounding pixel values within the window are extracted. For edge pixels, padding techniques such as zero-padding, mirror padding, or replication padding are applied to handle boundary cases. The extracted pixel values from the local neighborhood are stored in an array for further processing.

Step 4: Median Value Computation: Once the neighborhood pixel values are collected, they are sorted in ascending order. The median value, which is the middle value in the sorted list, is selected as the new pixel intensity for the center pixel. If the neighborhood size is even, the average of the two middle values is used. This step ensures that extreme intensity variations, such as noise spikes, are eliminated while maintaining the original structure of the image.

Step 5: Pixel Replacement and Image Reconstruction: After determining the median value for each pixel, the new intensity value is assigned to the center pixel, replacing its original value. This process is repeated iteratively for every pixel in the image, effectively smoothing out noise while preserving important details such as edges and textures. The reconstructed image is then formed with the new filtered values, producing a cleaner, enhanced version of the original image.

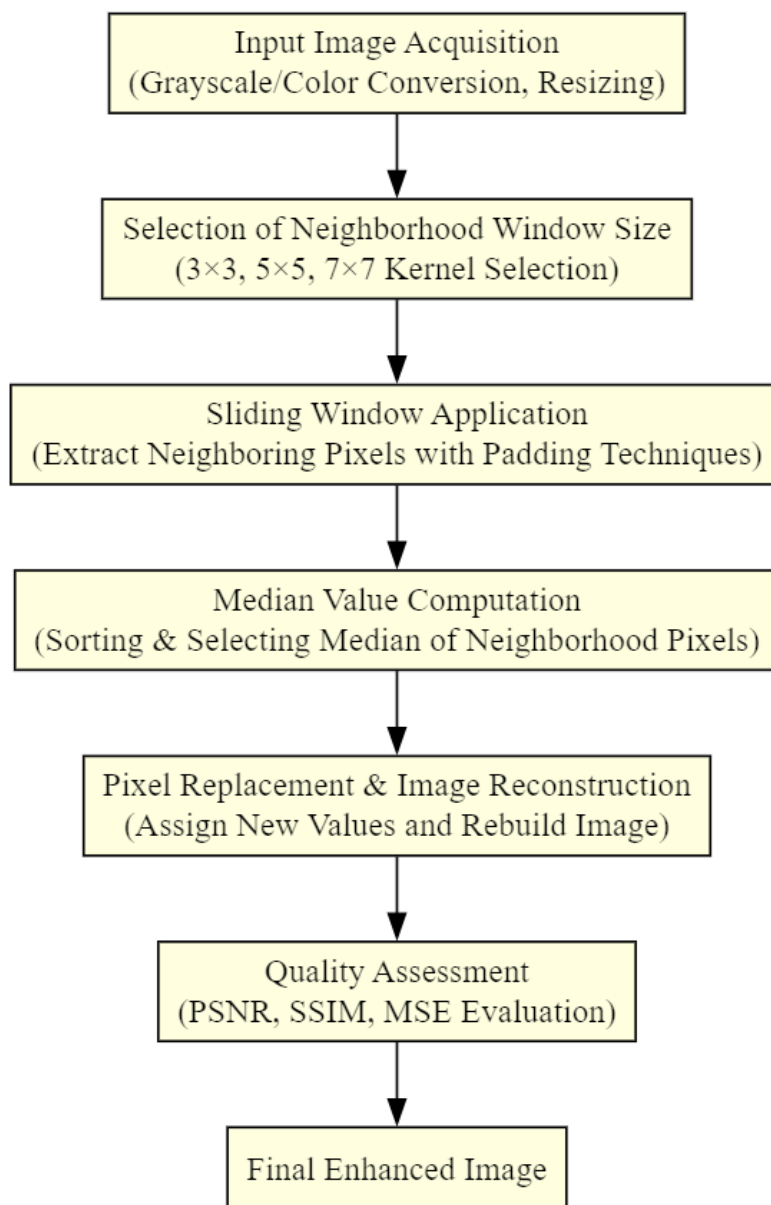


Figure 3.1. Existing Block Diagram.

Step 6: Quality Assessment: Once the median filtering is applied, additional post-processing techniques may be used to refine the enhanced image. These may include contrast stretching, histogram equalization, or sharpening to enhance visibility further. Quality assessment

metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), or Mean Squared Error (MSE) are used to evaluate the effectiveness of the enhancement. The output image is then compared to the original to ensure that noise reduction does not compromise essential image details.

3.2 Drawbacks

- **Loss of Fine Texture Details:** Since the median filter replaces each pixel value with the median of its surrounding neighborhood, it may eliminate small but essential texture details in an image. This is particularly noticeable in images with fine patterns or intricate textures, where filtering may cause excessive smoothing, leading to a loss of important visual information.
- **Edge Distortion with Large Window Sizes:** While the median filter is known for preserving edges, using larger window sizes can lead to distortion or unwanted artifacts near edges and object boundaries. This happens because a larger filter size averages more neighboring pixels, which may blur sharp edges or introduce unnatural transitions in the image.
- **Sensitivity to Window Size Selection:** Choosing an appropriate filter window size is crucial for optimal performance. A small window may not effectively remove noise, while a large window may introduce excessive smoothing. Selecting the wrong window size can negatively impact the balance between noise reduction and detail preservation, requiring trial-and-error adjustments or adaptive filtering techniques.

4.PROPOSED SYSTEM

4.1 Hybrid Adaptive HE

The proposed Hybrid Adaptive HE approaches effectively balances contrast enhancement while maintaining natural image quality, making it suitable for applications such as medical imaging, surveillance, and low-light photography enhancement.

Figure 4.1 shows the block diagram of proposed Hybrid Adaptive HE. The detailed operational procedure is given as follows

Step 1: Input Image Acquisition: The first step involves obtaining the input image, which can be a grayscale or color image in standard formats such as PNG, JPEG, or BMP. If the image is in a raw format, it must be preprocessed to ensure proper readability. At this stage, any preliminary adjustments such as resizing or noise removal can also be performed to enhance the effectiveness of subsequent processing steps.

Step 2: Conversion to YCrCbColor Space: Since Histogram Equalization (HE) is most effective in intensity-based channels rather than color channels, the image is converted from the standard RGB color space to the YCrCbcolor space. This transformation separates the luminance (Y) component, which represents the brightness of the image, from the chrominance components (Cr and Cb), which represent color information. Processing only the Y channel helps preserve the original colors while improving contrast and details in the image.

Step 3: Local Contrast Calculation in the Y Channel: To ensure adaptive contrast enhancement, local contrast variations within the image are analyzed. This involves dividing the Y channel into small non-overlapping blocks to facilitate localized processing.

- **Block Division:** The image is divided into smaller regions or blocks. The block size is chosen based on image resolution and content, ensuring that it captures local variations without being too small to introduce artifacts.
- **Variance Calculation for Local Contrast:** Within each block, the variance of pixel intensities is computed. Variance is a statistical measure of intensity dispersion, representing the level of contrast in a specific region. Higher variance indicates strong local contrast, while lower variance suggests uniform intensity. This information is

crucial for dynamically adjusting the clip limit for Contrast Limited Adaptive Histogram Equalization (CLAHE).

Step 4: Normalization of Local Contrast: The calculated variance values across all blocks are normalized to ensure a consistent scaling range. Normalization helps in mapping the variance values to a defined range, making it easier to use them for controlling the contrast enhancement parameters. This step ensures that the adaptive histogram equalization process remains stable across different image regions, preventing excessive enhancement in highly contrasted areas while improving visibility in low-contrast regions.

Step 5: Dynamic Adjustment of Clip Limit: Using the normalized local contrast values, a dynamic clip limit is set for each block. The clip limit determines the maximum height of histogram bins during the equalization process, preventing over-amplification of specific intensity ranges.

- In regions with high local contrast (high variance), the clip limit is set lower to avoid excessive enhancement, preserving natural details.
- In regions with low local contrast (low variance), the clip limit is increased to enhance visibility and improve details in dark or uniform areas.

This dynamic histogram control mechanism ensures that contrast enhancement is adaptive and context-aware rather than uniformly applied across the entire image.

Step 6: Application of CLAHE with the Dynamically Adjusted Clip Limit: After defining the clip limits for each block, CLAHE is applied. CLAHE operates by redistributing pixel intensities within each block using histogram equalization while preventing over-enhancement through the defined clip limit. The algorithm ensures that local contrast is improved without causing unnatural brightness variations or noise amplification.

Step 7: Merging Back the Y, Cr, and Cb Channels: Once CLAHE is applied to the Y channel, the enhanced luminance component is merged back with the original Cr and Cb channels. This reconstruction ensures that the original color information is preserved while the brightness and contrast are improved. The image is then converted back from the YCrCb color space to the standard RGB format for visualization and further processing.

Step 8: Output Image Generation: The final enhanced image is generated, exhibiting improved contrast, better visibility in darker regions, and preservation of natural details. The

dynamically controlled CLAHE ensures that artifacts such as over-brightening or noise amplification are minimized. The output image is evaluated for quality improvements using visual inspection and quantitative metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

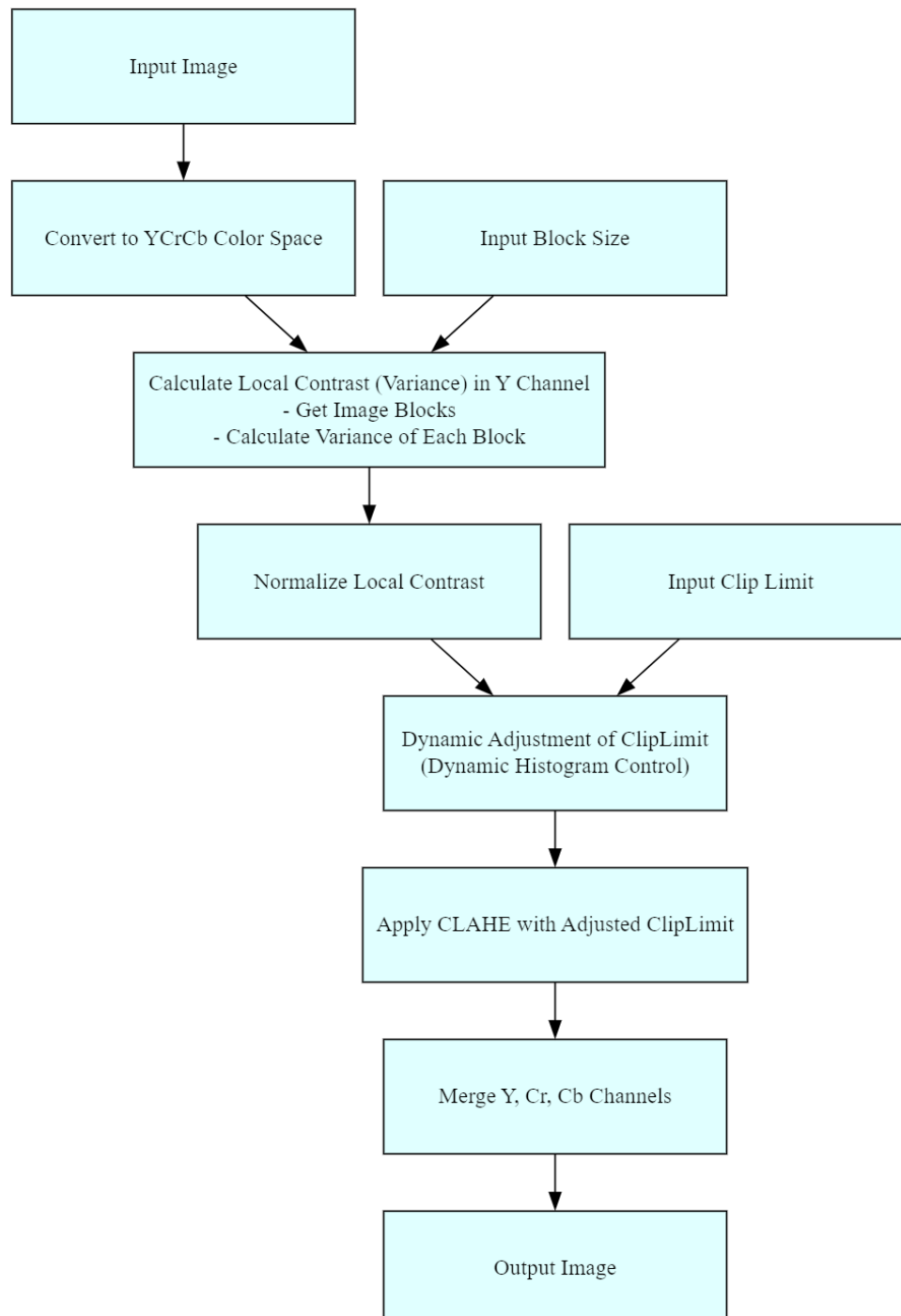


Figure 4.1. Proposed System Architecture.

4.2 CLAHE Approach

Figure 4.2 shows the structured CLAHE approach, which ensures balanced contrast enhancement while preventing over-amplification of noise and preserving fine image details, making it ideal for applications like medical imaging, surveillance, and low-light photography.

Step 1: Input Image Acquisition: The first step involves obtaining the input image, which can be in grayscale or color format. If the image is in color (RGB), it is typically converted to a suitable color space, such as YCrCb or Lab, where only the luminance (intensity) channel is processed. This ensures that contrast enhancement is applied without distorting the original color distribution.

Step 2: Image Division into Small Non-Overlapping Blocks (Tiles): CLAHE operates on small regions of the image rather than processing the entire image at once. The input image is divided into smaller, non-overlapping rectangular or square blocks called **tiles** (e.g., 8×8 , 16×16 pixels). These tiles are independently processed to improve local contrast while preventing over-enhancement in already well-contrasted areas. The choice of tile size affects the enhancement result:

- **Smaller tiles** improve local contrast but may introduce noise or artifacts.
- **Larger tiles** result in smoother transitions but may reduce fine details in low-contrast regions.

Step 3: Histogram Computation for Each Tile: For each tile, a histogram of pixel intensity values is computed. The histogram represents the frequency distribution of grayscale values within the tile, showing how pixel intensities are spread across the region. If the image is in color, only the luminance (Y) channel is used for histogram calculation.

Step 4: Contrast Limiting by Clipping Histogram: Unlike traditional Histogram Equalization (HE), which stretches the entire histogram, CLAHE introduces a clip limit to avoid excessive amplification of high-frequency intensity levels.

- A predefined clip limit is set, representing the maximum allowed height for any histogram bin.
- If any histogram bin exceeds this limit, the excess pixel values are redistributed across the other bins, ensuring a more balanced histogram without extreme brightness variations.

- This step prevents over-enhancement and reduces noise amplification, which is common in standard HE.

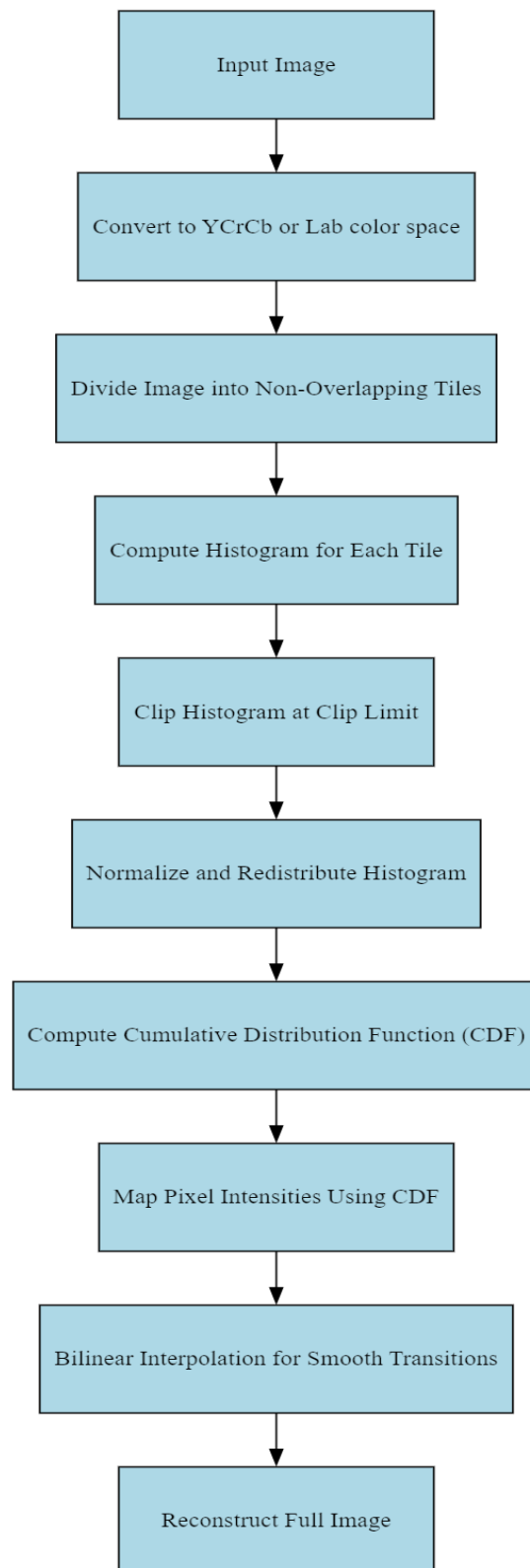


Figure 4.2. Proposed CLAHE Flowchart.

Step 5: Histogram Redistribution (Normalization): After clipping, the modified histogram is normalized by distributing the excess pixel values evenly across all intensity bins. This step ensures that the histogram remains smooth and that brightness variations are controlled across the entire tile. The goal is to create a more uniform intensity distribution while preserving natural details.

Step 6: Computation of Cumulative Distribution Function (CDF): Once the clipped histogram is normalized, the CDF is computed. The CDF represents the cumulative sum of histogram values, mapping each original intensity level to an enhanced intensity level based on the equalized distribution. This transformation ensures that pixel values are adjusted to enhance local contrast effectively.

Step 7: Histogram Mapping and Pixel Intensity Transformation: Using the CDF, pixel intensities within each tile are mapped to their new intensity values. This process adjusts brightness levels, making darker areas more visible and improving contrast in low-exposure regions. Since each tile is processed independently, local details are enhanced without affecting global brightness distribution.

Step 8: Bilinear Interpolation for Seamless Transition Between Tiles: After processing individual tiles, visible block artifacts may appear at the boundaries due to independent equalization. To ensure smooth transitions between adjacent tiles, bilinear interpolation is applied:

- The pixel values in overlapping regions between neighboring tiles are blended gradually.
- This prevents abrupt intensity changes and maintains a seamless appearance in the enhanced image.

Step 9: Reconstruction of the Full Image: Once all tiles are processed and merged, the final image is reconstructed. If the input was a color image, the enhanced luminance (Y) channel is recombined with the original chrominance (Cr, Cb) channels, and the image is converted back to the RGB color space for visualization.

4.3 Advantages

- **Improved Local Contrast Enhancement:** Unlike traditional Histogram Equalization (HE), which applies a global transformation, Hybrid Adaptive HE dynamically adjusts the enhancement in different regions of the image. By computing local contrast

(variance) and adapting the clip limit accordingly, it enhances details in both bright and dark regions while preserving natural variations.

- **Prevention of Over-Enhancement and Artifacts:** Standard HE can cause over-enhancement in already well-contrasted regions, leading to unnatural brightness shifts and noise amplification. By dynamically adjusting the clip limit based on local contrast, Hybrid Adaptive HE prevents over-enhancement in high-contrast areas while effectively boosting visibility in low-contrast regions.
- **Color Preservation:** By performing enhancement only on the luminance (Y) channel in the YCrCbcolor space, this method prevents unwanted color distortions that occur in direct RGB histogram equalization. This ensures that the image retains its natural colors while improving its brightness and contrast.

5. UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software researchs.

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

5.1 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram be capable of providing certain

functionalities. These functionalities provided by the class are termed “methods” of the class. Apart from this, each class have certain “attributes” that uniquely identify the class

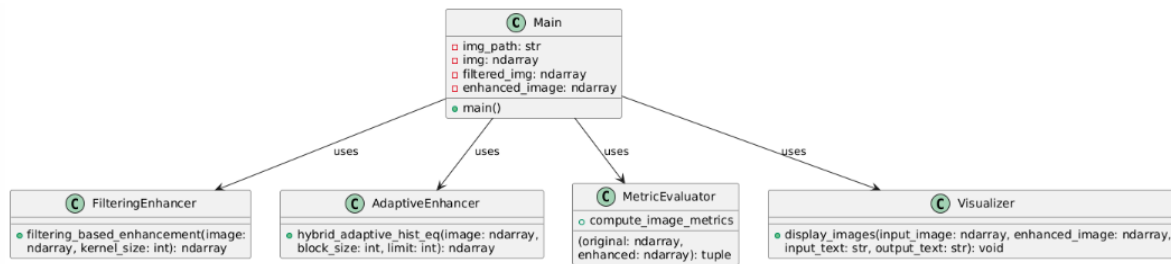


Figure 5.1. Class Diagram.

5.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

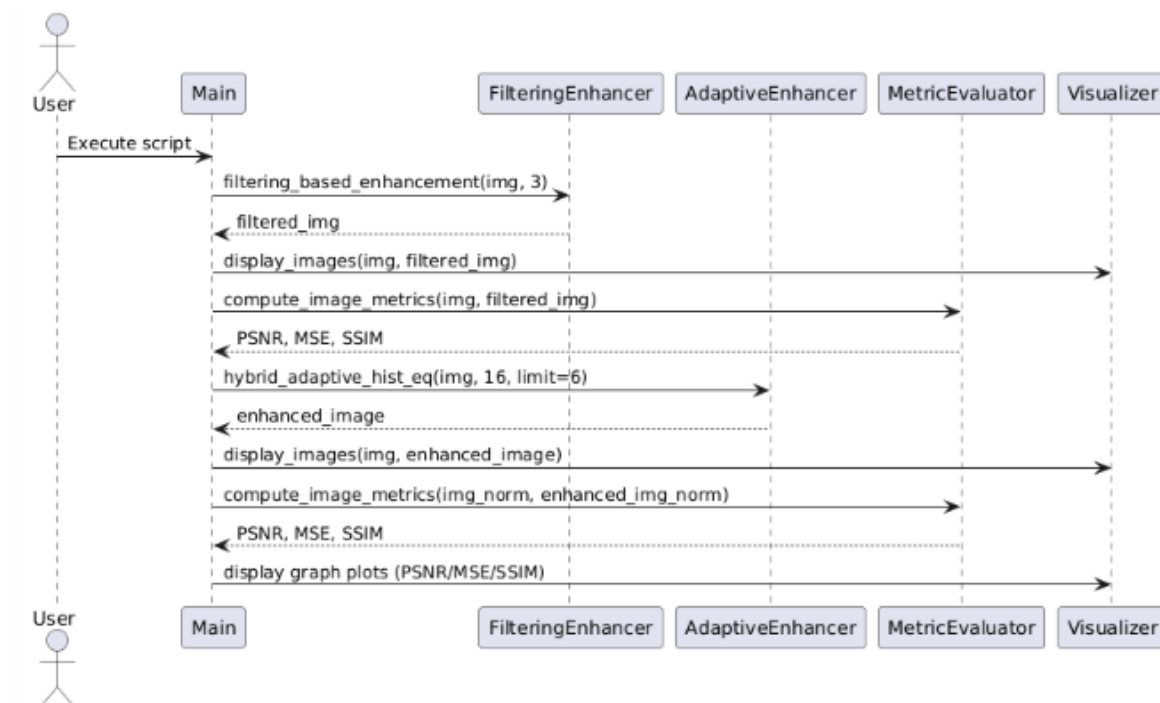


Figure 5.2. Sequence Diagram.

5.3 Activity diagram: Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

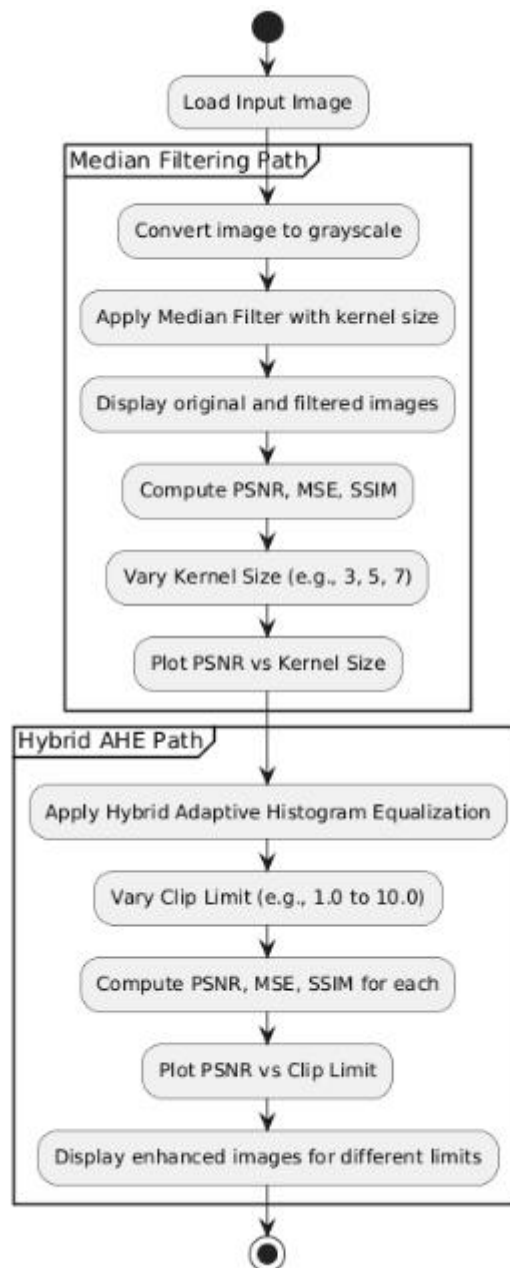


Figure 5.3. Activity Diagram.

5.4 Deployment diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components (“nodes”) exist (e.g., a web server, an application server, and a database server), what software components (“artifacts”) run on each node (e.g., web

application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes have sub nodes, which appear as nested boxes. A single node in a deployment diagram conceptually represent multiple physical nodes, such as a cluster of database servers.

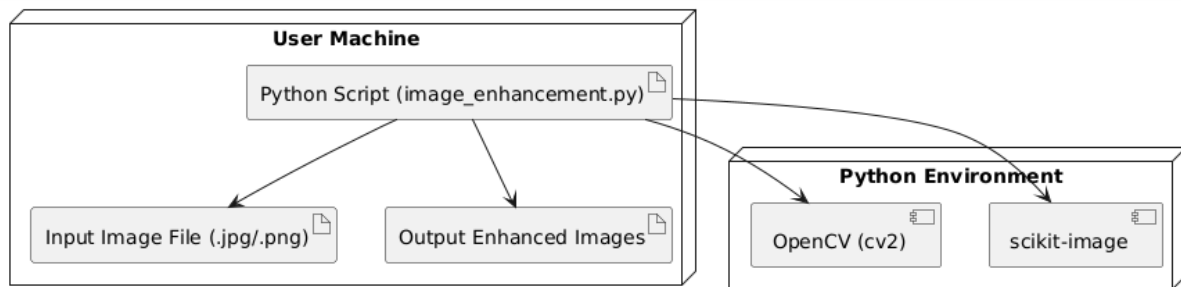


Figure 5.4. Deployment Diagram.

5.6 Use case diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

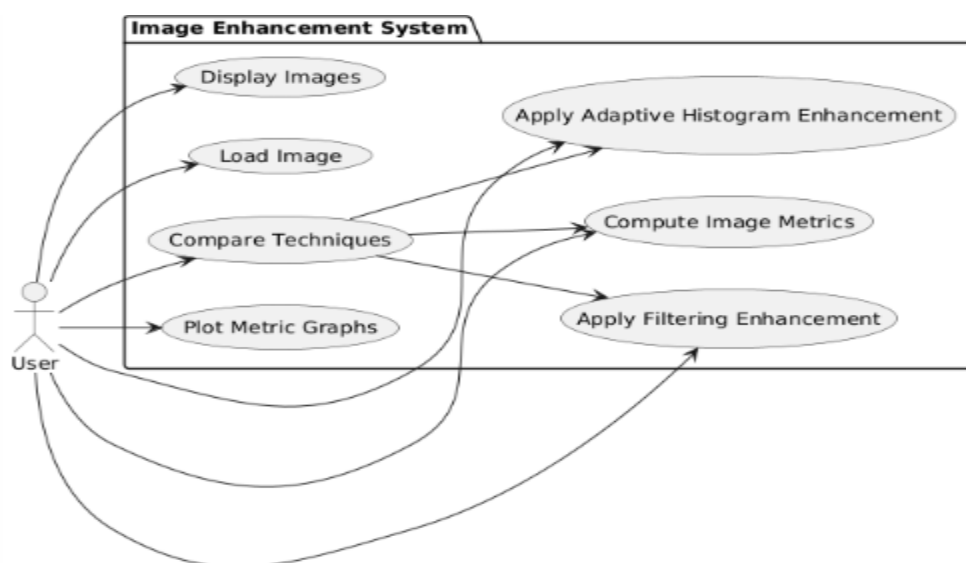


Figure 5.5. Use Case Diagram.

5.7 Component diagram: Component diagram describes the organization and wiring of the physical components in a system.

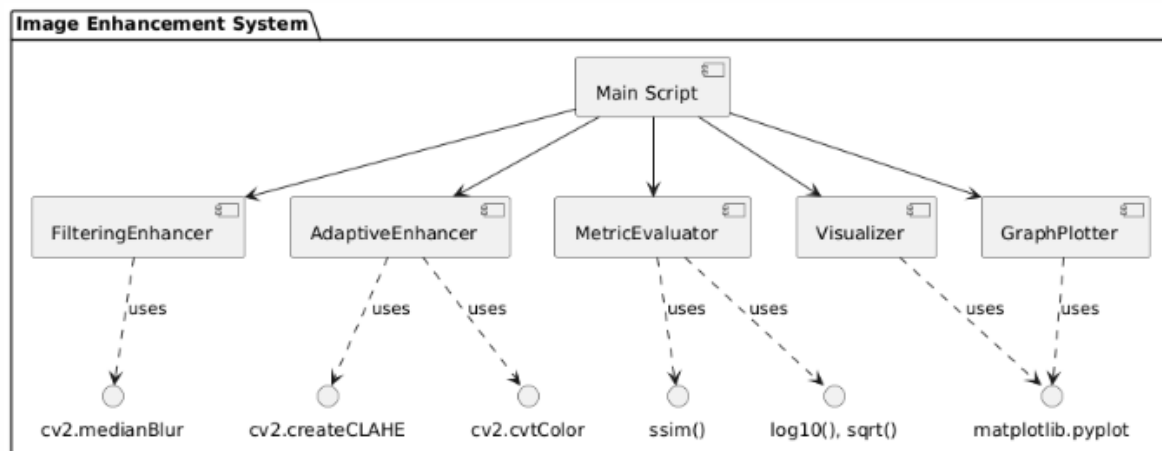


Figure 5.6. Component Diagram.

5.8 Data Flow Diagram (DFD):

In UML (Unified Modeling Language) is a graphical representation of the flow of data through a system. It's particularly useful for understanding the data inputs, outputs, processes, and storage within a system or software application. Here's

External Entities: These are entities outside the system that interact with it, such as users, other systems, or databases. They are represented as rectangles on the edges of the diagram.

Processes: Processes represent the actions or transformations that occur within the system. They are depicted as circles or ovals and typically have labels describing the action they perform on the data.

Data Stores: Data stores represent where data is stored within the system. They can be databases, files, or any other storage medium. Data stores are typically represented as rectangles.

Data Flows: Data flows represent the movement of data between external entities, processes, and data stores. They are depicted as arrows and indicate the direction of data flow.

Data Transformations: These represent the conversion or manipulation of data within a process. They can be depicted using labels on the data flow arrows or as separate process symbols.

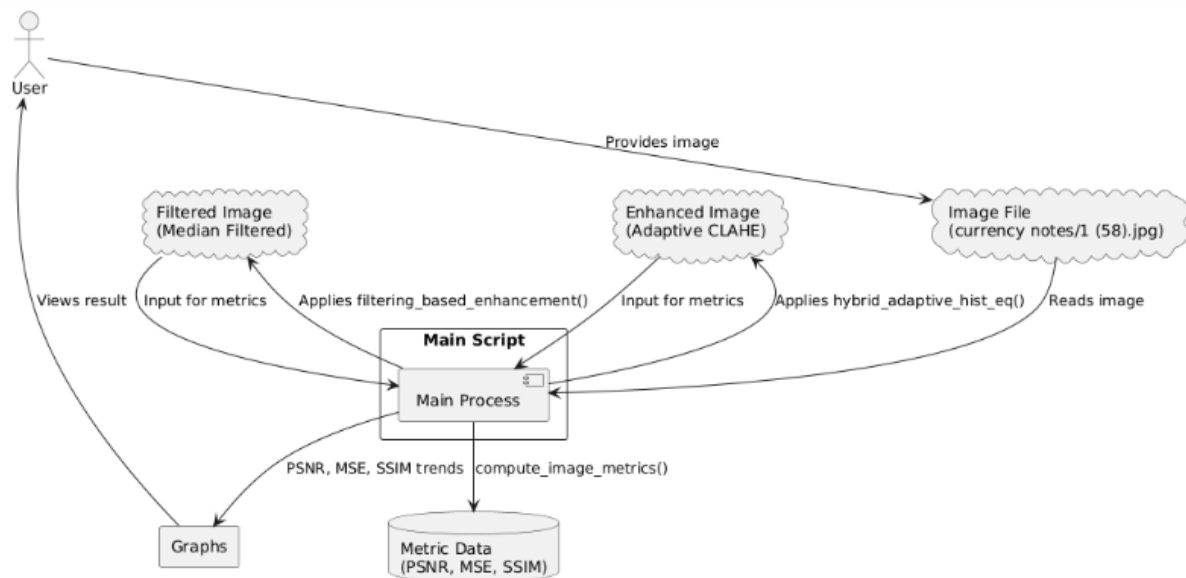


Figure 5.7. Data Flow Diagram.

5.9 System Architecture

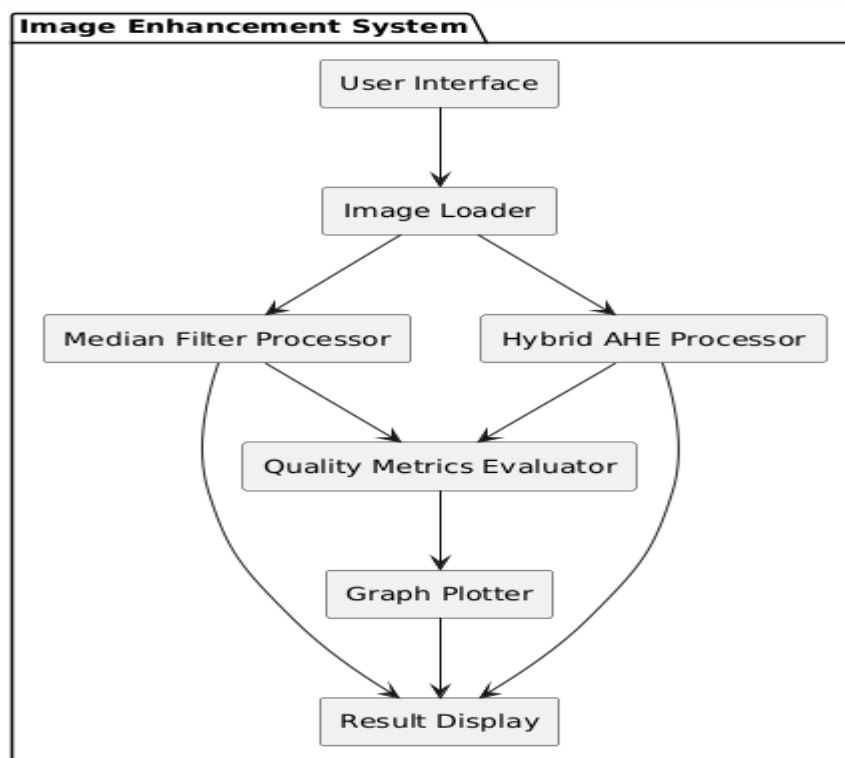


Figure 5.8. System Architecture.

6. SOFTWARE REQUIREMENTS

6.1 Software Requirements

Python 3.7.6

Python 3.7.6 serves as a pivotal version for developers and researchers due to its robust features, backward compatibility, and widespread support across a variety of libraries and frameworks. Released during a time when machine learning and data science tools were rapidly evolving, Python 3.7.6 provided a stable and consistent platform. This version includes critical improvements like enhanced asyncio functionality for asynchronous programming, increased precision for floating-point numbers, and optimized data structures. It became the go-to version for compatibility with popular libraries like TensorFlow 2.0, PyTorch, and Pandas, ensuring seamless integration and efficient execution for both academic and industrial applications.

Compared to older Python versions, 3.7.6 introduced several features such as dataclasses, which simplified boilerplate code for object-oriented programming. The improved async and await syntax made concurrent programming more intuitive, while changes to the standard library enhanced usability and performance. Over newer versions, Python 3.7.6 remains a preferred choice for legacy systems and researchs requiring compatibility with libraries that may not yet support the latest Python updates. Its combination of stability and maturity ensures that it is reliable for long-term researchs, especially in environments where upgrading the Python interpreter might disrupt existing workflows.

TensorFlow Environment

TensorFlow provides a comprehensive ecosystem for building, training, and deploying machine learning models. Its support for numerical computation and deep learning applications makes it a staple in AI research and development. By offering a flexible architecture, TensorFlow enables deployment across a variety of platforms, including desktops, mobile devices, and the cloud. The ability to scale across CPUs, GPUs, and TPUs ensures that TensorFlow is suitable for both small experiments and large-scale production systems.

TensorFlow's transition from older versions, like 1.x, to 2.x brought significant improvements in ease of use, including the introduction of the tf.keras API for building models, eager

execution for dynamic computation, and enhanced debugging capabilities. Compared to newer frameworks, TensorFlow retains a strong advantage due to its mature community support, extensive documentation, and integration with TensorFlow Extended (TFX) for managing production pipelines. Its compatibility with other libraries and tools, such as Keras and TensorBoard, makes it a robust choice for end-to-end machine learning solutions.

Packages Overview

Keras: Older versions of Keras required extensive configuration for custom model creation. Version 2.3.1 unified the APIs with TensorFlow integration, reducing overhead and enabling direct use of TensorFlow backends, ensuring faster execution and easier debugging. While newer versions focus on performance and distributed training, version 2.3.1 is lightweight and stable, making it ideal for smaller researchs without the complexity introduced in later iterations, which are more suited for advanced workflows.

NumPy: Version 1.19.5 introduced critical bug fixes and performance enhancements over older versions, especially for operations involving large datasets. The improved random number generator and better handling of exceptions provide more reliable results for numerical computations. This version remains compatible with a wide range of dependent packages. While newer versions optimize speed further, 1.19.5 balances stability and compatibility, ensuring fewer compatibility issues with older software stacks.

Pandas: Version 0.25.3 brought significant speed improvements for large-scale data processing, particularly in operations like groupby. Enhancements in handling missing data and improved compatibility with external libraries made this version more robust for data analysis tasks. While newer versions do not add features like enhanced type checking, 0.25.3 remains lightweight and stable for researchs that do not require cutting-edge functionalities, making it a practical choice for legacy systems.

Imbalanced-learn: Version 0.7.0 introduced optimized algorithms for handling class imbalances, such as improved SMOTE implementations. This update also enhanced the ease of integrating with scikit-learn pipelines. While newer versions do not contain experimental features, 0.7.0 is reliable and well-documented, ensuring robust performance in addressing data imbalance issues without unnecessary complexity.

Scikit-learn: Version 0.23.1 included improved support for cross-validation and hyperparameter optimization. Updates to RandomForestClassifier and GradientBoostingClassifier increased model accuracy and efficiency. 0.23.1 is widely tested and compatible with older hardware, making it a dependable choice for environments where the latest versions may introduce compatibility issues.

Imutils: This package provides an easy-to-use interface for image processing tasks. Its functions for resizing, rotating, and translating images simplified workflows compared to writing custom code. Its lightweight nature and stable functionality make it suitable for researches not requiring cutting-edge image manipulation techniques, balancing simplicity and capability.

Matplotlib: Version 3.x improved plot interactivity and introduced better 3D plotting capabilities. The `tight_layout` function and compatibility with modern libraries streamlined visualization workflows. The earlier versions maintain stability and compatibility with older datasets and software, avoiding potential issues from newer, untested updates.

Seaborn: Improved APIs in newer versions simplified aesthetic customization of plots. The addition of new themes and color palettes in 0.11.x enhanced visual appeal for exploratory data analysis. Older versions remain computationally less demanding, suitable for lightweight applications without requiring extensive customizations.

OpenCV-Python: Recent updates enhanced compatibility with deep learning frameworks and accelerated image processing pipelines, especially for real-time applications. Older versions are stable and resource-efficient, making them ideal for systems with limited computational capacity or for legacy applications.

H5Py: Version 2.10.0 improved file handling efficiency for large datasets. It introduced better support for advanced indexing, which is crucial for working with high-dimensional data. While newer versions support more advanced features, 2.10.0 ensures compatibility with older machine learning frameworks and models.

Jupyter: Jupyter improved the interactivity and scalability of notebooks for collaborative coding and visualization tasks. Integration with tools like Matplotlib made it a preferred environment for data exploration. Earlier versions are stable and lightweight, avoiding potential issues with dependencies introduced in newer releases.

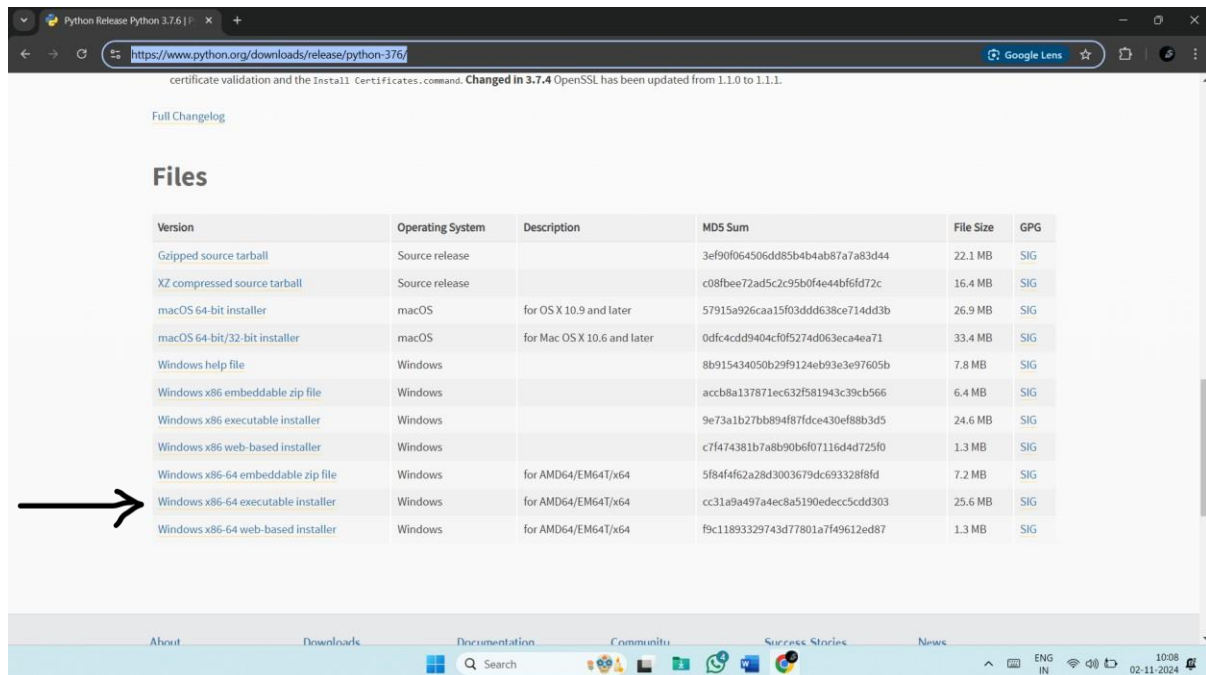
PYTHON INSTALLATION

INSTALLATION PROCESS OF PYTHON

STEP 1:

INSTALL PYTHON BY PRESSING THE PYTHON URL LINK.

<https://www.python.org/downloads/release/python-376/>



ScreenShot 6.1.

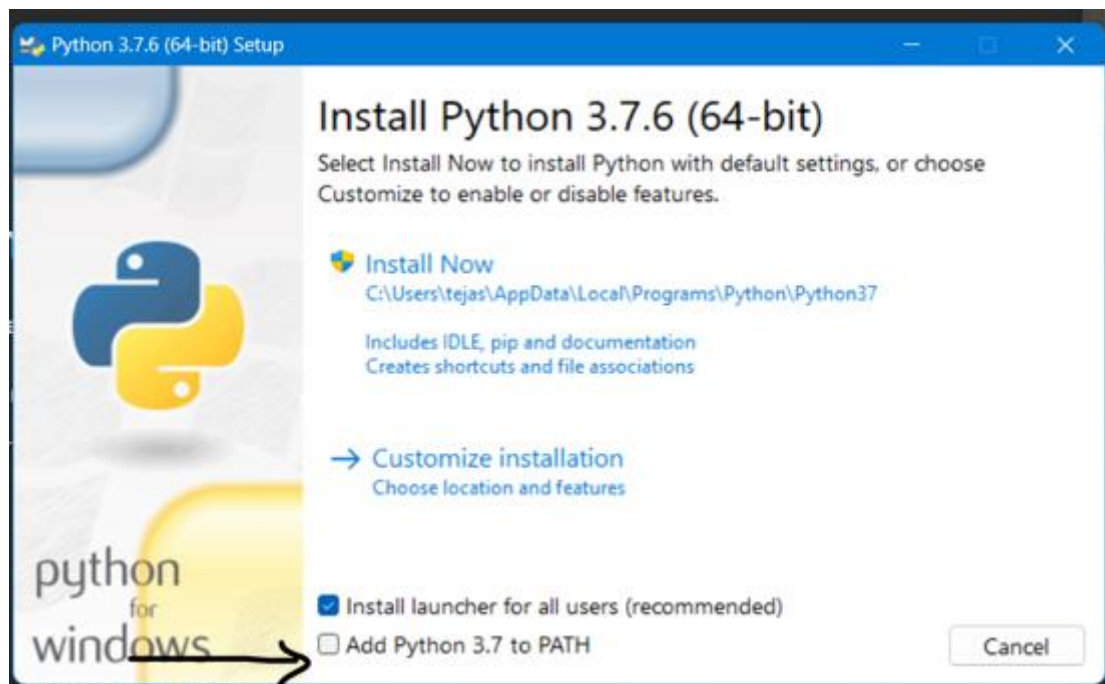
Then scroll down there you can see the files. After that click arrow located link to download.

STEP 2:

The python with our required 3.7.6 version will download.

STEP 3:

Click the below **tick box** to install your python in your path.



ScreenShot 6.2.



ScreenShot 6.3.

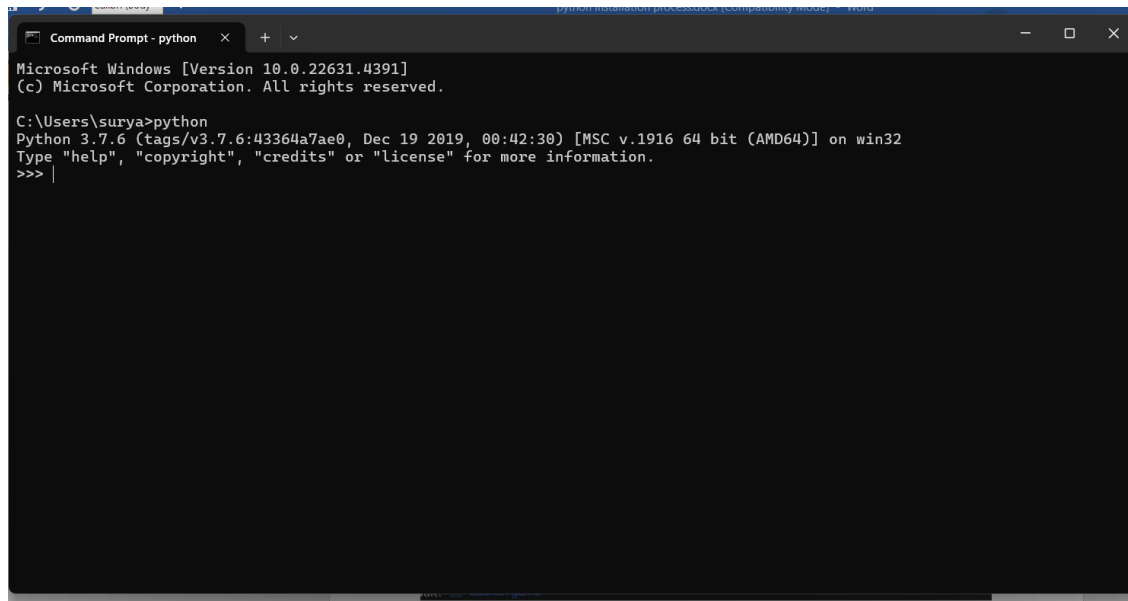
STEP 4: Then enter **install now**.



ScreenShot 6.4.

After completion of installation.

Step 5: open Command Prompt and enter python.



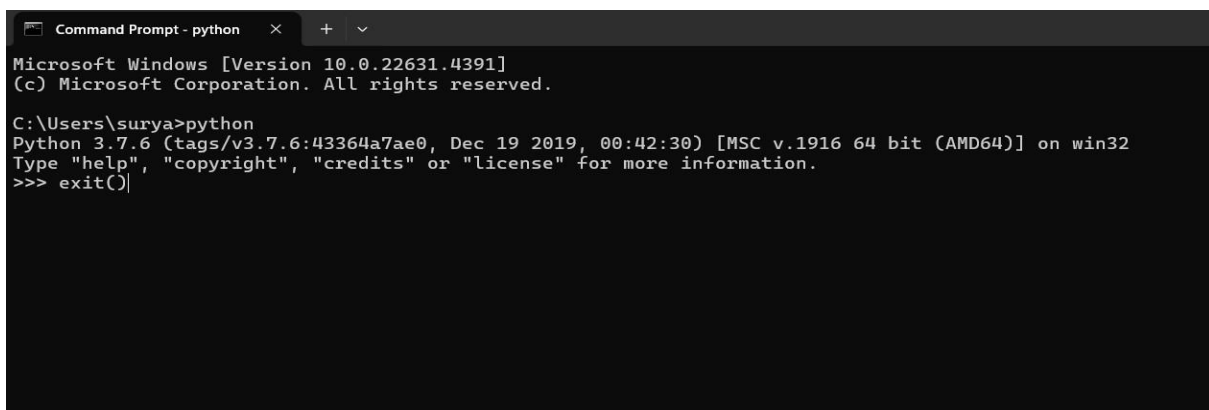
```
Command Prompt - python
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\surya>python
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

ScreenShot 6.5.

If it shows the python version as python 3.7.6. means it is perfectly installed.

Step 6: enter exit. By entering **exit()**.

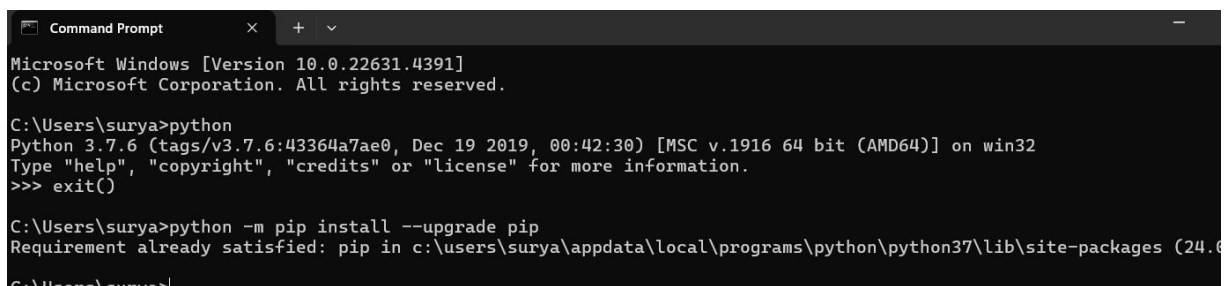


```
Command Prompt - python
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\surya>python
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
C:\Users\surya>
```

ScreenShot 6.6.

Step: install below packages.



```
Command Prompt
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\surya>python
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\surya>python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\surya\appdata\local\programs\python\python37\lib\site-packages (24.0.0)
C:\Users\surya>
```

ScreenShot 6.7.

install the below packages. `python -m pip install --upgrade pip` `pip install tensorflow==1.14.0`

`pip install keras==2.3.1` `pip install pandas==1.3.5`

`pip install scikit-learn==1.0.2` `pip install imutils`

`pip install matplotlib==3.2.2` `pip install seaborn==0.12.2`

`pip install opencv-python== 4.1.1.26` `pip install h5py==2.10.0`

`pip install numpy==1.19.2`

`pip install imbalanced-learn==0.7.0` `pip install jupyter`

`pip install protobuf==3.20.*` `pip install scikit-image==0.16.2`

Complete Explanation of installation

To install Python 3.7.6, first, visit the official Python website at Python Downloads. Scroll down to the "Files" section and select the appropriate installer based on your operating system. For Windows users, choose either the "Windows x86-64 executable installer" for 64-bit or the "Windows x86 executable installer" for 32-bit. macOS users should download the macOS 64-bit installer. Linux users can install Python using their respective package managers such as apt, dnf, or yum.

After downloading, run the installer and check the box that says "**Add Python to PATH**" before proceeding with the installation. Click on "**Install Now**" and wait for the process to complete. Once installed, open **Command Prompt** and type `python --version` to verify that Python 3.7.6 has been successfully installed. If the version appears correctly, it means the installation was successful. You can exit Python by typing `exit()`.

Once Python is installed, the next step is to upgrade pip, the package installer for Python. Open **Command Prompt** and run the command `python -m pip install --upgrade pip`. This ensures that you have the latest version of pip, which is required for installing other dependencies.

Now, install the necessary machine learning and deep learning libraries. Begin by installing TensorFlow and Keras using the commands `pip install tensorflow==1.14.0` and `pip install keras==2.3.1`. These libraries are crucial for building and training deep learning models.

Additionally, install Pandas (pip install pandas==1.3.5), NumPy (pip install numpy==1.19.2), and Scikit-learn (pip install scikit-learn==1.0.2) to handle data processing and machine learning tasks.

For image processing tasks, install OpenCV (pip install opencv-python==4.1.1.26), scikit-image (pip install scikit-image==0.16.2), and imutils (pip install imutils). These libraries help with tasks such as image transformations, object detection, and feature extraction. Additionally, visualization libraries like Matplotlib (pip install matplotlib==3.2.2) and Seaborn (pip install seaborn==0.12.2) will be useful for plotting data and analyzing results.

Other essential packages include H5py (pip install h5py==2.10.0) for handling HDF5 file formats, imbalanced-learn (pip install imbalanced-learn==0.7.0) for dealing with imbalanced datasets, and Jupyter Notebook (pip install jupyter) for interactive coding and visualization. Also, install Protobuf (pip install protobuf==3.20.*), which is required for serializing structured data in machine learning applications.

To ensure that all packages are installed correctly, open **Python** in **Command Prompt** by typing python, then try importing the installed libraries using import tensorflow as tf, import keras, import pandas as pd, import numpy as np, import matplotlib.pyplot as plt, import seaborn as sns, import cv2, import imutils, and from sklearn.model_selection import train_test_split. If there are no errors, the installation has been completed successfully.

For users who prefer using Jupyter Notebook, launch it by typing jupyter notebook in **Command Prompt**. This will open a browser interface where you can run Python code interactively. With all these steps completed, your Python environment is now fully set up for machine learning and deep learning researchs.

6.2 Hardware Requirements

Python 3.7.6 can run efficiently on most modern systems with minimal hardware requirements. However, meeting the recommended specifications ensures better performance, especially for developers handling large-scale applications or computationally intensive tasks. By ensuring compatibility with hardware and operating system, can leverage the full potential of Python 3.7.6.

Processor (CPU) Requirements: Python 3.7.6 is a lightweight programming language that can run on various processors, making it highly versatile. However, for optimal performance, the following processor specifications are recommended:

- **Minimum Requirement:** 1 GHz single-core processor.
- **Recommended:** Dual-core or quad-core processors with a clock speed of 2 GHz or higher. Using a multi-core processor allows Python applications, particularly those involving multithreading or multiprocessing, to execute more efficiently.

Memory (RAM) Requirements: Python 3.7.6 does not demand excessive memory but requires adequate RAM for smooth performance, particularly for running resource-intensive applications such as data processing, machine learning, or web development.

- **Minimum Requirement:** 512 MB of RAM.
- **Recommended:** 4 GB or higher for general usage. For data-intensive operations, 8 GB or more is advisable.

Insufficient RAM can cause delays or crashes when handling large datasets or executing computationally heavy programs.

Storage Requirements: Python 3.7.6 itself does not occupy significant disk space, but additional storage required for Python libraries, modules, and researchs.

- **Minimum Requirement:** 200 MB of free disk space for installation.
- **Recommended:** At least 1 GB of free disk space to accommodate libraries and dependencies.

Developers using Python for large-scale researchs or data science should allocate more storage to manage virtual environments, datasets, and frameworks like TensorFlow or PyTorch.

Compatibility with Operating Systems: Python 3.7.6 is compatible with most operating systems but requires hardware that supports the respective OS. Below are general requirements for supported operating systems:

- **Windows:** 32-bit and 64-bit systems, Windows 7 or later.
- **macOS:** macOS 10.9 or later.
- **Linux:** Supports a wide range of distributions, including Ubuntu, CentOS, and Fedora.

The hardware specifications for the OS directly impact Python's performance, particularly for modern software development.

7. FUNCTIONAL REQUIREMENT

Functional requirements are detailed statements that specify what a system should do. They describe the system's behavior, functions, and services, outlining how it responds to certain inputs, performs tasks, and interacts with users or other systems. Essentially, they answer the question, "What should the system do?" Here are some key aspects:

- **Functionality:** They define the specific functions or operations that the system must perform.
- **Inputs and Outputs:** They detail the types of inputs the system accepts and the outputs it produces.
- **User Interactions:** They describe how users interact with the system, including command inputs, error handling, and responses.
- **Data Management:** They outline requirements related to data storage, retrieval, and processing.
- **System Behavior:** They specify how the system behaves in various scenarios, including normal operations and exceptional conditions.

Below is a breakdown of the functions used in the research along with their requirements. These “function requirements” describe what each function is responsible for, the expected inputs, processing steps, and outputs or side effects. This can serve as a high-level specification for each function in the research.

Functional Requirements

1. Load Input Image

- **Functionality:** The program should accept an input image from a specified file path.
- **Details:** The image is read in BGR format using OpenCV (cv2.imread). This serves as the base image for all enhancement processes.

2. Filtering-Based Image Enhancement

- **Functionality:** Enhance the image by applying a **median filter** to reduce noise and smooth the image.
- **Details:**

- A kernel size (filter window) is taken as input.
- If the given kernel size is even, it is converted to an odd number to ensure proper filtering.
- `cv2.medianBlur` is used to apply the filter.
- This method improves the image by removing salt-and-pepper noise while preserving edges.

3. Display Input and Enhanced Images

- **Functionality:** Visually compare the original and enhanced images using side-by-side display.
- **Details:**
 - Uses `matplotlib` to create subplots.
 - Input and output images are displayed in RGB format with appropriate titles.
 - This helps users visually assess the enhancement quality.

4. Compute Image Quality Metrics

- **Functionality:** Measure and evaluate enhancement using:
 - **PSNR** (Peak Signal-to-Noise Ratio)
 - **MSE** (Mean Squared Error)
 - **SSIM** (Structural Similarity Index)
- **Details:**
 - **PSNR:** Measures image quality in dB, where higher is better.
 - **MSE:** Measures average squared difference between original and enhanced images.
 - **SSIM:** Measures perceptual similarity considering luminance, contrast, and structure.
 - These values help quantitatively evaluate how close the enhanced image is to the original.

5. Evaluate Filtering Performance Over Multiple Kernel Sizes

- **Functionality:** Analyze the impact of varying kernel sizes (limits 1–12) on filtering performance.
- **Details:**
 - For each kernel size, apply median filtering and compute PSNR, MSE, and SSIM.
 - Plot graphs to visualize how image quality varies with kernel size.
 - Enables selection of optimal kernel size for enhancement.

6. Hybrid Adaptive Histogram Equalization (AHE)

- **Functionality:** Perform image enhancement based on local contrast using an adaptive histogram method.
- **Details:**
 - Converts the image to YCrCbcolor space and operates on the luminance (Y) channel.
 - Image is divided into blocks, and local contrast (variance) is computed.
 - Clip limit for CLAHE is dynamically adjusted based on contrast.
 - Enhances brightness and contrast adaptively across different regions of the image.
 - cv2.createCLAHE is used for this purpose.

7. Evaluate Hybrid AHE Across Different Clip Limits

- **Functionality:** Experiment with different clip limits (1–12) and analyze their effect.
- **Details:**
 - For each clip limit, apply adaptive enhancement and compute PSNR, MSE, SSIM.
 - Plot metric values against the limits to observe trends and identify best-performing parameter.

- Helps find the balance between noise amplification and contrast enhancement.

8. Display Enhanced Outputs for All Clip Limits

- **Functionality:** Visually display enhanced outputs for each clip limit.
- **Details:**
 - Creates a grid of images showing how enhancement varies with different limit values.
 - Helps users compare the visual differences side by side.
 - Useful for selecting the most visually appealing result.

8. SOURCE CODE

```
import cv2

import numpy as np

import os

import matplotlib.pyplot as plt

from skimage.metrics import structural_similarity as ssim

from math import log10, sqrt


def display_images(input_image, enhanced_image, input_text="Input Image",
output_text="Enhanced Image"):

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.imshow(cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB))

plt.title(input_text)

plt.axis('off')


plt.subplot(1, 2, 2)

plt.imshow(cv2.cvtColor(enhanced_image, cv2.COLOR_BGR2RGB))

plt.title(output_text)

plt.axis('off')


plt.show()
```

```
def filtering_based_enhancement(image, kernel_size):  
    if kernel_size % 2 == 0:  
        kernel_size += 1  
    enhanced_image = cv2.medianBlur(image, kernel_size)  
    return enhanced_image
```

```
def compute_image_metrics(original, enhanced):  
    mse_val = np.mean((original - enhanced) ** 2)  
    if mse_val == 0:  
        psnr_value = 100  
    else:  
        max_pixel = 255.0  
        psnr_value = 20 * log10(max_pixel / sqrt(mse_val))  
  
    mse_value = mse_val  
    ssim_value = ssim(original, enhanced, multichannel=True)  
    return psnr_value, mse_value, ssim_value
```

```
def hybrid_adaptive_hist_eq(image, block_size, limit):  
    img_ycrb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
```

```
y, cr, cb = cv2.split(img_ycrb)

height, width = y.shape

local_contrast = np.zeros_like(y)

for i in range(0, height, block_size):

    for j in range(0, width, block_size):

        block = y[i:i+block_size, j:j+block_size]

        local_contrast[i:i+block_size, j:j+block_size] = np.var(block)

max_contrast = np.max(local_contrast)

min_contrast = np.min(local_contrast)

contrast_range = max_contrast - min_contrast

adaptive_clip_limit = limit * (contrast_range / (max_contrast + 1e-5))

adaptive_clip_limit = min(adaptive_clip_limit, limit)

clahe = cv2.createCLAHE(clipLimit=adaptive_clip_limit, tileGridSize=(8, 8))

cl = clahe.apply(y)

img_ycrb = cv2.merge([cl, cr, cb])

result = cv2.cvtColor(img_ycrb, cv2.COLOR_YCrCb2BGR)

return result

# Load and filter image

img_path = r"currency notes\1 (58).jpg"
```

```
img = cv2.imread(img_path)

filtered_img = filtering_based_enhancement(img, 3)

display_images(img, filtered_img, input_text="Input Image", output_text="Existing")

reduced_img = img


# Metrics for existing method

psnr_value, mse_value, ssim_value = compute_image_metrics(img, filtered_img)

print(f"Existing Method: PSNR, Value: {psnr_value:.2f} dB")

print(f"Existing Method: MSE, Value: {mse_value:.2f}")

print(f"Existing Method: SSIM, Value: {ssim_value:.4f}")


# Analyze over kernel size limits for median filter

limits = range(1, 13)

psnr_values, mse_values, ssim_values = [], [], []


for limit in limits:

    filtered_img = filtering_based_enhancement(reduced_img, limit)

    psnr_val, mse_val, ssim_val = compute_image_metrics(reduced_img, filtered_img)

    psnr_values.append(psnr_val)

    mse_values.append(mse_val)

    ssim_values.append(ssim_val)


plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
```

```
plt.plot(limits, psnr_values, marker='o', color='b')

plt.title("PSNR vs Limit")

plt.xlabel("Limit")

plt.ylabel("PSNR (dB)")


plt.subplot(1, 3, 2)

plt.plot(limits, mse_values, marker='o', color='r')

plt.title("MSE vs Limit")

plt.xlabel("Limit")

plt.ylabel("MSE")


plt.subplot(1, 3, 3)

plt.plot(limits, ssim_values, marker='o', color='g')

plt.title("SSIM vs Limit")

plt.xlabel("Limit")

plt.ylabel("SSIM")

plt.tight_layout()

plt.show()


# Proposed method

limit = 6

enhanced_image = hybrid_adaptive_hist_eq(reduced_img, 16, limit)

display_images(reduced_img, enhanced_image, input_text="Input Image",
output_text="Proposed Adaptive")
```



```
# Normalize for metric calculation

reduced_img1 = reduced_img.astype(np.float32) / 255.0

enhanced_image1 = enhanced_image.astype(np.float32) / 255.0

psnr_value, mse_value, ssim_value = compute_image_metrics(reduced_img1,
enhanced_image1)

print(f"Proposed Method: PSNR, Value: {psnr_value:.2f} dB")

print(f"Proposed Method: MSE, Value: {mse_value:.2f}")

print(f"Proposed Method: SSIM, Value: {ssim_value:.4f}")


# Analyze proposed method over different limits

psnr_values, mse_values, ssim_values = [], [], []


for limit in limits:

    enhanced_image = hybrid_adaptive_hist_eq(reduced_img, block_size=16, limit=limit)

    reduced_img1 = img.astype(np.float32) / 255.0

    enhanced_image1 = enhanced_image.astype(np.float32) / 255.0

    psnr_val, mse_val, ssim_val = compute_image_metrics(reduced_img1, enhanced_image1)

    psnr_values.append(psnr_val)

    mse_values.append(mse_val)

    ssim_values.append(ssim_val)


plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

plt.plot(limits, psnr_values, marker='o', color='b')

plt.title("PSNR vs Limit")
```

```
plt.xlabel("Limit")
plt.ylabel("PSNR (dB)")

plt.subplot(1, 3, 2)
plt.plot(limits, mse_values, marker='o', color='r')
plt.title("MSE vs Limit")
plt.xlabel("Limit")
plt.ylabel("MSE")

plt.subplot(1, 3, 3)
plt.plot(limits, ssim_values, marker='o', color='g')
plt.title("SSIM vs Limit")
plt.xlabel("Limit")
plt.ylabel("SSIM")
plt.tight_layout()
plt.show()

# Display multiple enhanced outputs
num_images = len(limits)
rows, cols = 3, 4
plt.figure(figsize=(15, 12))

for i, limit in enumerate(limits):
    plt.subplot(rows, cols, i + 1)
```

```
enhanced_image = hybrid_adaptive_hist_eq(img, block_size=16, limit=limit)

plt.imshow(cv2.cvtColor(enhanced_image, cv2.COLOR_BGR2RGB))

plt.title(f"Limit {limit}")

plt.axis('off')


plt.tight_layout()

plt.show()
```

9. RESULTS AND DISCUSSION

9.1 Implementation Description

This structured implementation enables a comparative study between traditional filtering and the proposed adaptive histogram-based enhancement method, both quantitatively (through metrics) and visually (through plots).

- The code begins by **importing essential libraries** such as cv2 for image processing, numpy for numerical operations, matplotlib.pyplot for visualization, and skimage.metrics for calculating image quality metrics like SSIM.
- A utility function named display_images is defined to **display input and enhanced images side-by-side** using matplotlib for better visual comparison.
- The filtering_based_enhancement function is implemented using **median filtering** with a customizable kernel size. The kernel size is adjusted to be odd if an even value is passed, ensuring proper filtering behavior.
- The compute_image_metrics function calculates three important **image quality metrics**: PSNR (Peak Signal-to-Noise Ratio), MSE (Mean Squared Error), and SSIM (Structural Similarity Index). These are used to quantify the visual improvement after enhancement.
- An image is read using OpenCV (cv2.imread) and passed through the median filtering method. The original and enhanced images are then displayed, and the **existing method's performance metrics** are printed.
- A loop is performed across a range of kernel sizes (limits from 1 to 12) to analyze the **effect of varying the filter size** on image quality. The PSNR, MSE, and SSIM values are calculated for each filter size and visualized using line plots.
- A more advanced function hybrid_adaptive_hist_eq is implemented which combines **Contrast Limited Adaptive Histogram Equalization (CLAHE)** with dynamic control. The contrast of each local block (determined by block_size) is computed, and a **locally adaptive clipLimit** is set based on the block's variance.

- This proposed adaptive histogram method is applied to the original image using a fixed block size and a tunable limit. The result is visualized and evaluated using the same image quality metrics, showing its performance.
- Another loop evaluates the proposed method over different limit values (from 1 to 12), calculating PSNR, MSE, and SSIM for each case. Results are stored and plotted to visualize how the enhancement quality changes with different dynamic histogram limits.
- Finally, the enhanced images generated by the adaptive method for all limit values are displayed in a **3×4 grid of subplots**, allowing visual comparison of all results side-by-side.

9.2 Existing Results

Figure 9.1 shows the result of an existing filtering-based enhancement method applied to a currency note image. The median filtering technique moderately reduces noise but also blurs important features like micro-text and embedded patterns, which are critical in authentication and detection tasks.

Figure 9.2 shows the graphical performance analysis of the existing method using PSNR, SSIM, and MSE metrics for various filter kernel limit values. The graphs reveal a limited range of performance, with relatively low PSNR and higher MSE, indicating less effective enhancement. The SSIM values remain moderate, showing basic structural retention.



Figure 9.1. Existing Filtering-based enhancement resultson currency note.

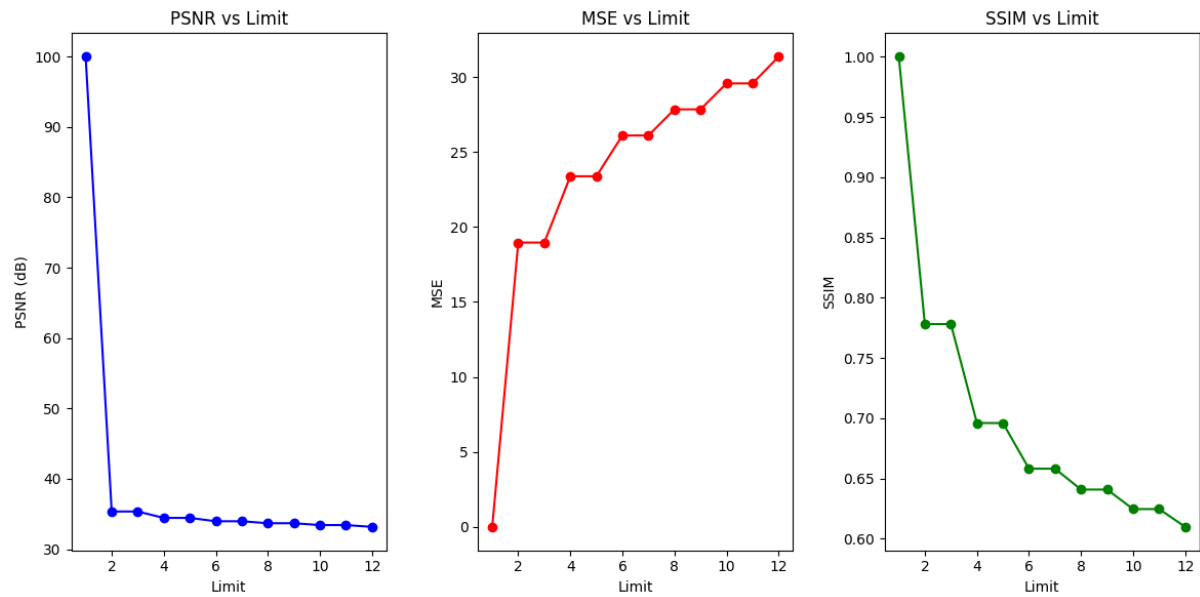


Figure 9.2. Existing PSNR, SSIM, MSE graphs for various limit values

Figure 9.3 shows the result of the same filtering-based method applied to a CCTV camera surveillance image. While some noise suppression is achieved, fine details of information remain unclear. The accompanying metrics—PSNR of 35.35 dB, MSE of 18.96, and SSIM of 0.7781—support this moderate enhancement result.

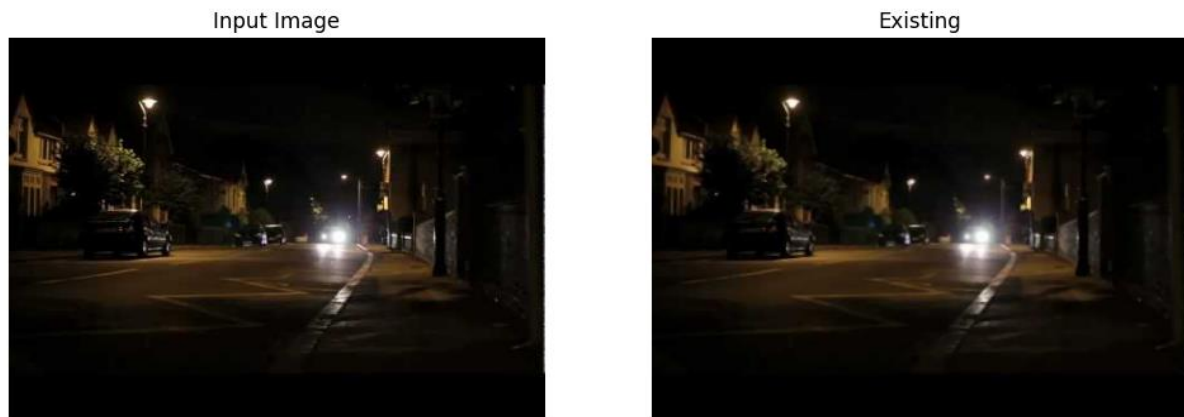


Figure 9.3. Existing Filtering-based enhancement results on CCTV camera surveillance image.

9.3 Proposed Results

Figure 9.4 shows the output of the proposed Adaptive Histogram Equalization (AHE) technique applied to a currency note image. The method significantly boosts local contrast,

revealing intricate features like embedded patterns, serial numbers, and texture variations. Quantitatively, it achieves a PSNR of 59.99 dB, a very low MSE of 0.07, and an SSIM of 0.4758, indicating a trade-off between detail enhancement and structural distortion.

Figure 9.5 shows the graphical performance comparison of the proposed method across different clip limit values. The PSNR curve shows a substantial increase compared to the existing method, while the MSE drops significantly. The SSIM values fluctuate, which reflects variations in local structure due to adaptive contrast adjustments.

Figure 9.6 shows the effect of applying AHE to the currency note image for different clip limits. As the limit changes, the contrast enhancement also changes, making different features more or less visible. This figure demonstrates the fine control offered by the AHE method.

Figure 9.7 shows the proposed method applied to a CCTV surveillance image. The enhanced result reveals significantly more visual information compared to the existing method, such as clearer faces, clothing patterns, and background structures—crucial in forensic and surveillance scenarios.

Figure 9.8 shows the proposed AHE enhancement applied to the same CCTV image across various limit values. This demonstrates the flexibility and adaptiveness of the method, showing how parameter tuning can optimize enhancement for different lighting and environmental conditions.



Figure 9.4. Proposed AHE results on currency note.

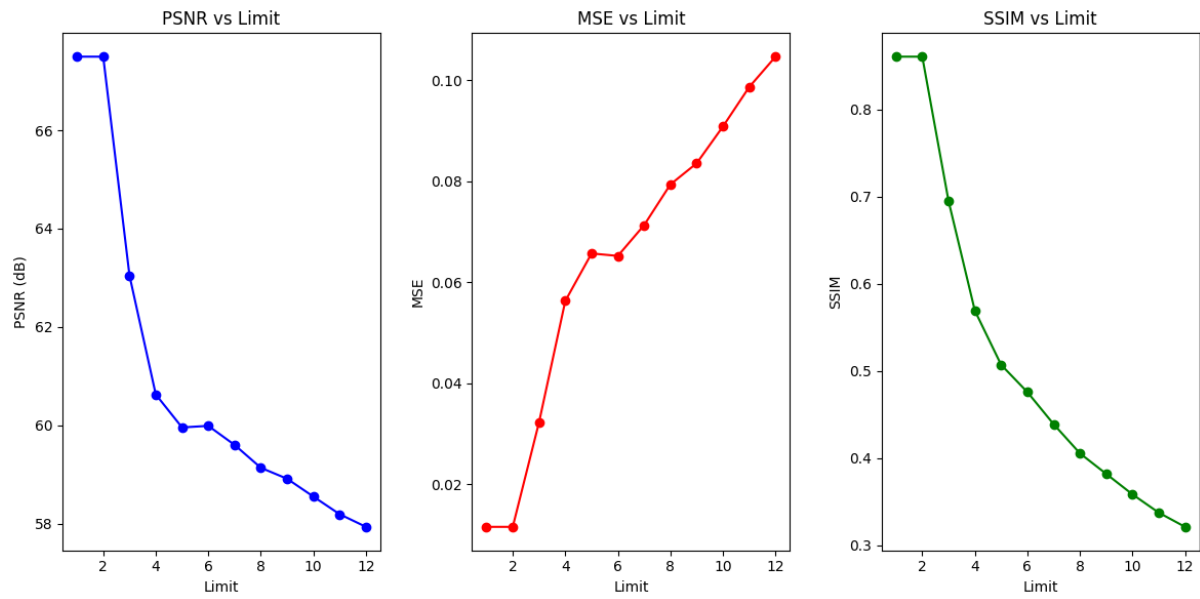


Figure 9.5. Proposed PSNR, SSIM, MSE graphs for various limit values.



Figure 9.6. Proposed AHE results on currency note for various limits.

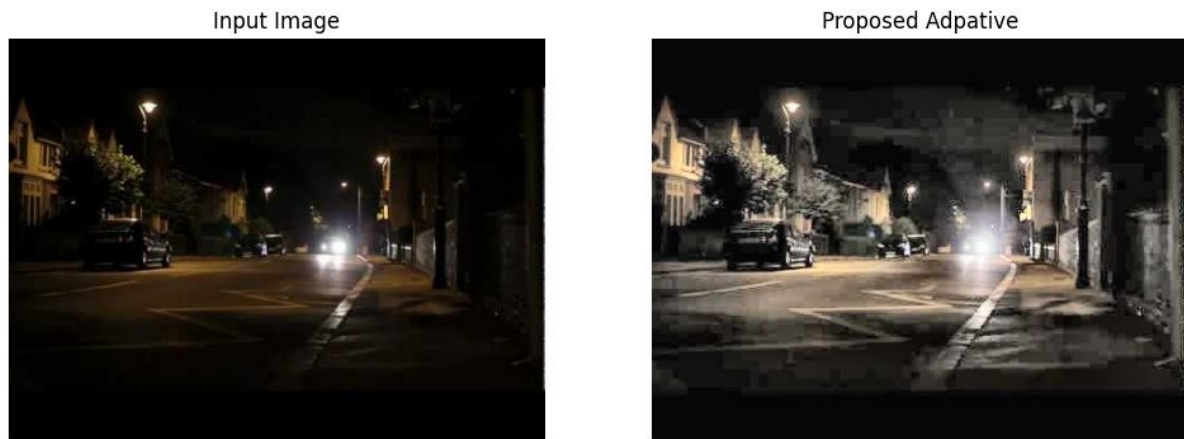


Figure 9.7. ProposedAHE results on CCTV camera surveillance image.

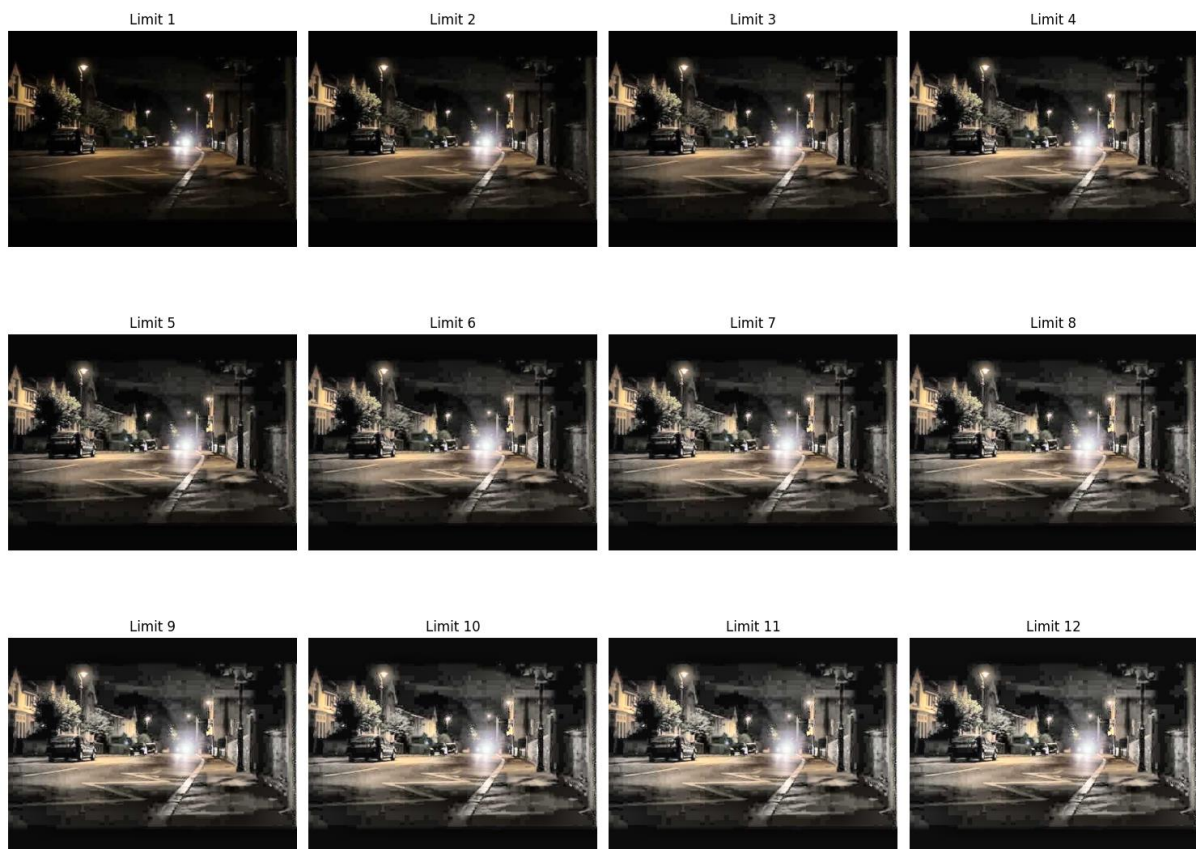


Figure 9.8. Proposed AHE results on CCTV camera surveillance image for various limits.

The comparison in Table 9.1 clearly illustrates the superiority of the proposed AHE method in terms of quantitative performance and visual enhancement quality. The PSNR value for the proposed method (59.99 dB) is significantly higher than that of the existing method (35.35 dB), indicating a much cleaner and higher-fidelity enhancement. Similarly, the MSE value

drops drastically from 18.96 to just 0.07, proving the effectiveness of the proposed technique in minimizing reconstruction error and preserving original image characteristics.

However, while the SSIM for the existing method is higher (0.7781) than that of the proposed method (0.4758), this is due to AHE introducing contrast exaggerations that alter local structure, leading to a slight drop in structural similarity. Nonetheless, the trade-off is acceptable because the perceived visual quality is enhanced, especially in applications where feature visibility matters more than pixel-wise similarity (e.g., currency verification or CCTV monitoring).

In terms of adaptability and control, the proposed method stands out due to its ability to dynamically adjust contrast locally using tunable parameters like clip limit. This allows for finer control over image enhancement compared to the static kernel size of median filtering, which offers limited flexibility.

Moreover, the graphical analysis (Figures 3.2 vs. 3.5) shows that the proposed method achieves better trends across various settings, further validating its robustness and effectiveness. Finally, the visual results across multiple limits (Figures 3.6 and 3.8) show that the proposed technique is not only superior in performance but also adaptable to different lighting conditions and image types.

Table 9.1. Comparative Analysis.

Criteria	Existing Method	Proposed Method
PSNR (dB)	35.35	59.99
MSE	18.96	0.07
SSIM	0.7781	0.4758
Visual Quality	Moderate clarity, noise reduced	High detail enhancement, better local contrast
Graph Trends	Low PSNR, high MSE, moderate SSIM	High PSNR, low MSE, fluctuating SSIM
Parameter Variability	Kernel Size	CLAHE Limit (clip limit)

10. CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

The comparative analysis between the existing filtering-based enhancement method and the proposed Adaptive Histogram Equalization (AHE) technique clearly demonstrates the superiority of the latter in terms of both objective quality metrics and visual clarity. The proposed method significantly enhances the local contrast of images, making subtle and crucial details—such as embedded patterns on currency notes and facial or object features in surveillance footage—more visible. Quantitatively, the proposed method achieves a higher Peak Signal-to-Noise Ratio (PSNR) of 59.99 dB and a drastically lower Mean Squared Error (MSE) of 0.07 compared to the existing method's 35.35 dB PSNR and 18.96 MSE, reflecting a more accurate and efficient enhancement process. Although the Structural Similarity Index (SSIM) is slightly lower in the proposed method (0.4758 vs. 0.7781), this is attributed to the aggressive local contrast enhancements performed by AHE, which, while reducing some structural similarity, significantly improve perceptual and feature visibility. Overall, the proposed method offers better adaptability, tunability, and performance across a range of image conditions, establishing it as a robust solution for enhancing low-quality or poorly illuminated images.

10.2 Future Scope

Although the proposed AHE-based method shows promising results, there is ample scope for further enhancements and research. One key area is the integration of machine learning or deep learning models to intelligently adjust AHE parameters like clip limit and tile size based on the content and quality of the input image. This would automate the enhancement process and adaptively optimize results for diverse scenarios without manual tuning. Another promising direction is the hybridization of AHE with edge-preserving filters or detail-aware smoothing algorithms to further improve SSIM while retaining high PSNR and low MSE. Additionally, real-time implementation of the enhancement pipeline on embedded systems or surveillance hardware could vastly improve on-the-fly processing of security footage or financial document verification systems. Finally, exploring multi-scale AHE techniques and combining them with object detection or recognition modules can enhance their applicability in smart surveillance, automated authentication, and other intelligent vision systems. These future extensions will push the boundaries of contrast enhancement while preserving structural integrity and minimizing artifacts.

References

1. Demirel, H.; Ozcinar, C.; Anbarjafari, G. Satellite image contrast enhancement using discrete wavelet transform and singular value decomposition. *IEEE Geosci. Remote Sens. Lett.* 2010, 7, 333–337.
2. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 3rd ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2006.
3. Ibrahim, H.; Kong, N.S.P. Brightness preserving dynamic histogram equalization for image contrast enhancement. *IEEE Trans. Consum. Electron.* 2007, 53, 1752–1758.
4. Arici, T.; Dikbas, S.; Altunbasak, Y. A histogram modification frame work and its application for image contrast enhancement. *IEEE Trans. Image Process.* 2009, 18, 1921–1935
5. Huang, S.C.; Cheng, F.C.; Chiu, Y.S. Efficient contrast enhancement using adaptive gamma correction with weighting distribution. *IEEE Trans. Image Process.* 2013, 22, 1032–1041.
6. Celik, T. Two-dimensional histogram equalization and contrast enhancement. *Pattern Recog.* 2012, 45, 3810–3824.
7. Celik, T.; Tjahjadi, T. Contextual and variational contrast enhancement. *IEEE Trans. Image Process.* 2011, 20, 3431–3441
8. Celik, T. Spatial entropy-based global and local image contrast enhancement. *IEEE Trans. Image Process.* 2014, 23, 5298–5308.
9. Demirel, H.; Anbarjafari, G.; Jahromi, M.N.S. Image equalization based on singular value decomposition. In Proceedings of the 2008 23rd International Symposium on Computer and Information Sciences, Istanbul, Turkey, 27–29 October 2008; pp. 1–5. [Google Scholar]
10. Lee, E.; Kim, S.; Kang, W.; Seo, D.; Paik, J. Contrast enhancement using dominant brightness level analysis and adaptive intensity transformation for remote sensing images. *IEEE Geosci. Remote Sens. Lett.* 2013, 10, 62–66.

11. Jang, J.H.; Kim, S.D.; Ra, J.B. Enhancement of optical remote sensing images by subband-decomposed multiscale retinex with hybrid intensity transfer function. *IEEE Geosci. Remote Sens. Lett.* 2011, 8, 983–987
12. Zhang, G.; Chen, Q.; Sun, Q. Illumination normalization among multiple remote-sensing images. *IEEE Geosci. Remote Sens. Lett.* 2014, 11, 1470–1474.
13. Lore, K.G.; Akintayo, A.; Sarkar, S. LLNet: A Deep Autoencoder Approach to Natural Low-light Image Enhancement. *Pattern Recognit.* 2017, 61, 650–662.
14. Shen, L.; Yue, Z.; Feng, F.; Chen, Q.; Liu, S.; Ma, J. Msr-Net: Low-light Image Enhancement Using Deep Convolutional Network. *arXiv* 2017, arXiv:1711.02488. Available online: <https://arxiv.org/pdf/1711.02488.pdf>
15. Wei, C.; Wang, W.; Yang, W.; Liu, J. Deep Retinex Decomposition for Low-light Enhancement. *arXiv* 2018, arXiv:1808.04560. Available online: <https://arxiv.org/abs/1808.04560>
16. Han, H.; Chung, S.-W.; Kang, H.-G. Mirnet: Learning multiple identities representations in overlapped speech. *arXiv* 2020, arXiv:2008.01698.
17. Zhang, C.; Yan, Q.; Zhu, Y.; Li, X.; Sun, J.; Zhang, Y. Attention-based network for low-light image enhancement. In Proceedings of the 2020 IEEE International Conference on Multimedia and Expo (ICME), London, UK, 6–10 July 2020; Volume 1.
18. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; Volume 1, pp. 234–241.