## OFFLINE PENCIL SKETCH AND EDGE DETECTION TOOL FOR CULTURAL IMAGE PRESERVATION

**A SOCIALLY RELEVANT MINI PROJECT REPORT**

***Submitted by***

## VAISHNAVI.N(211423104711) VAISHNAVI.V(211423104712)

***in partial fulfillment for the award of the degree***

***of***

## BACHELOR OF ENGINEERING

***in***

#### COMPUTER SCIENCE AND ENGINEERING

****

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**OCTOBER 2025**

## PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## BONAFIDE CERTIFICATE

Certified that this project report **“OFFLINE PENCIL SKETCH AND EDGE DETECTION TOOL FOR CULTURAL IMAGE**

**PRESERVATION”** is the Bonafide work of **VAISHNAVI.N (211423104711), VAISHNAVI. V (211423104712),** who carried

out the project work under my supervision.

**Signature of HOD with date Signature of Supervisor with date Dr.L.JABASHEELA M.E.,Ph.D., Mrs.S.Sharmila M.E.[Ph.D.,]**

Professor and Head, Department of CSE, Panimalar Engineering College, Chennai- 123.

Assistant Professor, Department of CSE,

Panimalar Engineering College, Chennai- 123.

Submitted for 23CS1512-SOCIALLY RELEVANT Mini Project Viva– Voce examination held on

**INTERNALEXAMINER EXTERNALEXAMINER**

### DECLARATION BY THE STUDENT

##### We VAISHNAVI.N[211423104711], VAISHNAVI.V [211423104712],

hereby declare that this project report titled **“OFFLINE PENCIL SKETCH AND EDGE DETECTION TOOL FOR CULTUTRAL**

**IMAGE PRESERVATION”,** under the guidance of Coordinator **Mrs.S.Sharmila M.E.[ph.D.,]** is the work done by us and we have not plagiarized or submitted to any other degree in any university by us.

## VAISHNAVI.N[211423104711] VAISHNAVI.V [211423104712]

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and Dr.SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,Ph.D., for providing us**

withthenecessaryfacilities forcompletionofthis project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timelyconcern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr.L.JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our Project Coordinator **Dr.SUBEDHA.V, M.tech.,Ph. D**., and our Project Guide **Mrs.S.SHARMILA.S, M.E.[Ph.D.,]** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**VAISHNAVI.N (211423104711)**

**VAISHNAVI.V (211423104712)**

## ABSTRACT

This project presents the design and development of a lightweight offline image processing tool created in Python using the OpenCV library, intended for transforming digital images into pencil sketch and edge-detected formats. The motivation for this work arises from the limitations of existing solutions, which are often commercial, demand expensive hardware, or rely on constant internet connectivity, making them unsuitable for under-resourced environments such as rural schools, NGOs and cultural organizations. By relying on well- established computer vision techniques such as grayscale conversion, Gaussian filtering and Canny edge detection, the proposed system produces high-quality stylized outputs that emphasize structural details while generating realistic sketch effects. Unlike cloud-based or deep learning-driven platforms, the application runs entirely offline, requires minimal computational resources and is open- source, ensuring accessibility and adaptability for a wide range of users. The system not only addresses technical needs but also contributes to broader social and cultural goals. For cultural preservation, it provides simplified high-contrast representations of artworks, monuments and historical images, aiding in documentation and restoration. For education, it offers a practical and affordable resource for teaching computer vision concepts and digital creativity. For artists and designers, it opens opportunities for low-cost stylization and innovation in visual media.

**CHAPTER NO**

## TABLE OF CONTENTS

#### TITLE PAGE

**NO.**

**ABSTRACT** v

##### LISTOFFIGURES vii

##### LISTOFABBREVIATIONS ix

##### LISTOFTABLES x

1. [INTRODUCTION 1](#_TOC_250008)
   1. [Overview 1](#_TOC_250007)
   2. [Problem Definition 2](#_TOC_250006)
   3. Literature Survey 3
2. [SYSTEMANALYSIS 8](#_TOC_250005)
   1. [Existing System 8](#_TOC_250004)
   2. Proposed System 9
   3. Implementation Environment 11
3. [SYSTEMDESIGN 12](#_TOC_250003)
   1. [UML Diagrams 12](#_TOC_250002)
4. SYSTEMARCHITECTURE 16
   1. ArchitectureDiagram 16
   2. [Module DesignSpecification 19](#_TOC_250001)
5. SYSTEM IMPLEMENTATION 21

Sample Coding 21

1. SYSTEM TESTING 23
   1. Performance Metrics 23
   2. Test Cases 28
2. CONCLUSION 30
   1. [Conclusion 30](#_bookmark0)
   2. [Future enhancement 31](#_bookmark1)
3. APPENDICES 32

[A1SDG goals 32](#_bookmark2)

[A2Sample Screenshot 34](#_TOC_250000)

A3 PaperPublications 36

[A4 Plagiarism report 43](#_bookmark3)

1. REFERENCES 51

## LIST OF FIGURES

**FIG NO.** 3.1.1

3.1.2.

4.1.1

## FIGURE DESCRIPTION

Class Diagram for pencil sketch Sequence Diagram for pencil sketch Architecture diagram for pencil sketch

## PAGE NO

13

15

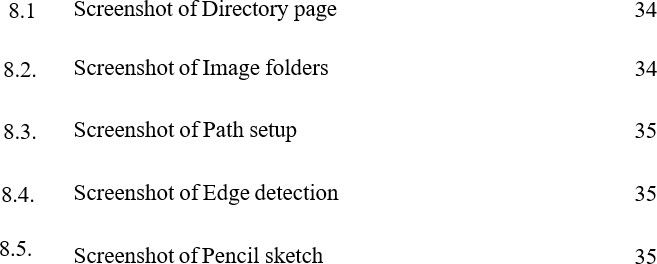
16

* + 1. Process time 23



x

* + 1. Memory usage 25
    2. CPU Utilization Graph 26
    3. Sample Output Images 26
    4. ProcessTime for Batch images 27



#### LISTOF ABBREVIATIONS

|  |  |
| --- | --- |
| **S. NO** | **ABBREVIATIONS** |
| 1 | BMP-BitmapImage File |
| 2 | HTML– Hyper Text Markup Language |
| 3 | CSS– Cascade Style Sheet |
| 4 | CLI-Command Line Interface |
| 5 | JPEG-Joint Photographic Experts Group |
| 6 | ML- Machine Learning |
| 7 | AI - Artificial intelligence |
| 8 | UML-Unified Modeling Language |
| 9 | RGB-Red Green Blue (Color Model) |
| 10 | SGD-Stochastic Gradient Descent |
| 11 | SDG-Sustainable Development Goals |
| 12 | PNG-Portable Network Graphics |
| 13 | SSD– Single Shot Detection |
| 14 | VGG - Visual Geometry Group |
| 15 | RAM -Random Access Memory |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLENO.** | **FIGURE DESCRIPTION** | **PAGE NO.** |
| 6.2. | Testcase- Image Detection | 28 |

# CHAPTER 1

## INTRODUCTION

## OVERVIEW

The list of figures presented in this report serves as a comprehensive guide to all the diagrams, charts and screenshots that have been included to support the design, development and evaluation of the project. Figures are arranged systematically according to their respective chapters, each carrying a clear caption that reflects its purpose and relevance. Beginning with the system design phase, the report introduces the Data Flow Diagram, UML Use Case Diagram and UML Activity Diagram, which collectively capture the logical flow of operations, the interactions between users and the system and the sequential processing steps that govern the image transformation tasks. These figures play a crucial role in laying down the foundation of the project by visually describing the problem space and the designed solution.

As the report progresses, the system architecture and module specifications are highlighted through detailed architectural diagrams and component interaction models. These figures illustrate how the system is internally structured, the division of responsibilities among different modules and the way these modules interact to achieve smooth functioning of the image processing pipeline. By depicting these technical aspects graphically, the reader gains a clearer understanding of the system’s organization beyond textual descriptions.

In the implementation section, figures such as the backend algorithm flowchart, pseudocode snapshots and the folder structure diagrams provide concrete insights into the practical realization of he system.

### PROBLEM DEFINITION

Digital images have become an essential part of communication, creativity and technology. With the rise of digital art, content creation and visual documentation, there is a growing demand for tools that can modify and enhance images quickly and effectively. While many applications are available for this purpose, most require heavy installations, internet connectivity, or paid subscriptions, which limits accessibility for everyday users.

One of the most common transformations required in practice is converting images into forms such as edge-detected outlines and pencil sketch representations. These are not only useful for artistic purposes but also for educational demonstrations, design prototyping and computer vision preprocessing. However, existing tools are often either too complex for beginners or lack the ability to perform these tasks efficiently in an offline environment.

The problem, therefore, is to design and develop a lightweight, offline image processing tool that can provide high-quality edge detection and sketch conversion without depending on external servers or internet resources. The system should be simple to use, efficient in execution and adaptable for future extensions, thereby addressing the gap between heavy professional tools and basic user requirements.



2

3

## 1.2 LITERATURE REVIEW

##### Holistically-Nested Edge Detection (HED) – Xie, S. & Tu, Z. (2022), CVPR

The Holistically-Nested Edge Detection (HED) framework represents a significant milestone in the evolution of computer vision–based edge detection. Unlike traditional edge detectors such as Sobel, Prewitt, or Canny that rely on manually designed filters, HED utilizes deep learning to automatically learn hierarchical edge representations from data. This method employs a fully convolutional neural network (FCN) that integrates multiple feature layers from different depths within the network to produce a unified, fine-grained edge map.

The core innovation of HED lies in its deep supervision strategy, where side outputs from intermediate layers are supervised at different scales. This allows the network to capture both coarse global structures and fine local details of objects. Such an approach improves edge detection accuracy, especially in images with complex textures, shadows, or overlapping objects. HED’s design ensures that every pixel in the image contributes to edge prediction, creating a more consistent and context-aware boundary representation.

HED has been tested on several benchmark datasets such as BSDS500, achieving state-of- the-art results in edge precision and recall. Its adaptability has influenced many subsequent models, serving as a base for improvements in semantic segmentation, salient object detection, and medical imaging.

However, despite its accuracy, HED requires significant computational .

4



##### Pixel Difference Network (PiDiNet) – Su, Z. et al. (2021), arXiv:2108.07009

The Pixel Difference Network (PiDiNet) introduces an innovative and lightweight approach for edge detection, focusing on balancing high accuracy with computational efficiency. Unlike conventional convolutional networks that depend on heavy filters and large amounts of training data, PiDiNet replaces complex convolutional operations with simple pixel difference operators that measure intensity variations between neighboring pixels. This makes it extremely fast and energy-efficient while maintaining strong edge detection performance.

PiDiNet’s architecture is inspired by the observation that edge detection fundamentally relies on local pixel intensity differences rather than complex hierarchical representations. The Pixel Difference Convolution (PDC) layers used in PiDiNet can be implemented with minimal parameters, allowing the network to operate effectively even on devices with limited memory and processing capacity. As a result, PiDiNet performs well in real-time applications, achieving edge quality comparable to deeper networks like HED or DexiNed but with a fraction of the computational cost.

The paper also highlights PiDiNet’s generalization ability across different datasets, proving its robustness under varying illumination, noise, and object types. Such resilience is particularly valuable for offline or low- resource environments, where consistent performance is needed without access to online computation or cloud-based models.

5

##### DexiNed: Deep Edge Detection Without Pretraining – Poma, X., Riba, E., & Sappa, A.D. (2022)

DexiNed (Dense Extreme Inception Network for Edge Detection) represents a modern advancement in edge detection that removes the dependency on pretrained backbone networks. Traditional deep learning models for edge detection, such as HED, often rely on pretrained weights from large image classification networks like VGG or ResNet. DexiNed overcomes this limitation by learning edge features directly from scratch using a carefully designed dense architecture based on inception modules.

The key idea behind DexiNed is to use dense connections and multi-scale feature aggregation to improve edge continuity and reduce false detections. This allows the model to preserve both fine textures and object-level boundaries effectively. Because DexiNed learns from raw edge datasets without transfer learning, it demonstrates strong adaptability to new visual domains, including cultural images, artwork, and historical patterns that differ significantly from standard datasets.

DexiNed also offers significant improvements in model interpretability and transparency compared to conventional deep CNNs. It achieves high- quality results without requiring extensive pretraining, making it computationally more accessible and efficient. Furthermore, the dense architecture ensures that features extracted from earlier layers directly influence the final prediction, improving stability and consistency across varying image conditions.

6

4

##### Improved Canny-Based Edge Detection for Brake Disc Inspection – Guo, Z. et al. (2024), Journal of Computer Research and Innovation

This paper focuses on optimizing the Canny edge detection algorithm, one of the most widely used classical methods in image processing. The original Canny method, proposed in 1986, involves Gaussian filtering, gradient calculation, non-maximum suppression, and hysteresis thresholding. While highly effective, its performance is sensitive to noise and parameter selection.

Guo et al. (2024) present an improved Canny-based approach that enhances edge continuity, robustness, and detection accuracy. By introducing adaptive thresholding and optimized Gaussian smoothing, their version minimizes noise influence and increases precision in identifying fine object details. The proposed method was tested on industrial images for brake disc inspection, achieving higher signal-to-noise ratios and better edge sharpness compared to traditional implementations.

Although this study focuses on mechanical inspection, its findings are transferable to artistic and cultural imaging, where fine edge detail and texture preservation are equally critical. The improved algorithm maintains computational simplicity while providing more reliable output—qualities essential for offline tools like the proposed pencil sketch system.

This project applies similar optimization ideas by using adaptive Gaussian blur and color dodge blending to enhance contrast and clarity in sketch generation. The strength of this research lies in demonstrating how

classical algorithms’ effectiveness withoutresorting todeep learning.

##### PAGE: Phase-Stretch Adaptive Gradient-Field Extractor –

**MacPhee, A. & Jalali, B. (2023), eLight, Springer**

The Phase-Stretch Adaptive Gradient-Field Extractor (PAGE) presents a unique physics- inspired approach to edge detection. It applies the concept of optical phase stretching, simulating how light diffracts through media, to identify transitions in image intensity. Unlike traditional or neural approaches that rely solely on digital convolution, PAGE leverages analog-inspired computations to detect faint and subtle edges.

This technique excels in processing low-contrast and artistic images, such as those found in cultural archives, heritage art, or ancient manuscripts. By stretching image gradients in the phase domain, PAGE amplifies weak features that are typically missed by conventional methods. It adapts dynamically to texture and illumination variations, ensuring accurate edge localization even in degraded or complex visuals.

While PAGE currently requires specialized photonic hardware for real-time performance, its conceptual design offers inspiration for software-based adaptive edge detectors. The idea of emphasizing intensity transitions through non-linear gradient transformations directly aligns with the sketch effect generation implemented in this project.

By drawing parallels with PAGE, the proposed tool integrates adaptive contrast enhancement and controlled Gaussian blurring to produce visually aesthetic and informative sketch outputs. The resemblance lies in it.

# CHAPTER 2 SYSTEMANALYSIS

# EXISTING SYSTEM

In the current technological landscape, numerous image processing applications exist that can perform functions such as edge detection, image enhancement, and artistic transformations. Popular commercial tools like Adobe Photoshop, CorelDRAW, and Sketch Book offer advanced capabilities for creating digital art and applying sketch or filter effects.

However, these systems are resource-intensive, require paid licenses, and depend on high-end hardware to function efficiently. Moreover, many of these solutions rely on cloud-based processing or require continuous internet connectivity to access AI- driven features, making them unsuitable for offline or low-resource environments.

While open-source platforms such as GIMP and Krita attempt to provide free alternatives, they often demand a significant level of technical expertise from users. These tools may also require third-party plugins or scripting knowledge to implement custom effects like pencil sketch conversion or precise edge detection. As a result, beginners, students, and users in rural or under-resourced areas face difficulties in using these applications effectively.

Furthermore, the computational complexity of deep learning–based stylization tools adds another layer of limitation. Neural network models for artistic transformations are often large, require GPU acceleration, and depend on online frameworks or pre-trained datasets. These constraints

makesuchtools impractical for users who need lightweight and offline..

**Disadvantages of the Existing System**

The existing image stylization and edge detection systems suffer from several significant limitations that restrict their accessibility, affordability, and practicality. Most of the currently available applications are web-based or cloud-dependent, requiring a continuous internet connection for image processing. This dependence creates barriers for users in rural areas, remote regions, or underdeveloped communities, where reliable internet access is limited or unavailable. Furthermore, these systems are typically resource- intensive, demanding high-performance hardware such as advanced processors, dedicated GPUs, and large memory capacity. This makes them unsuitable for low-cost devices or institutions operating with limited technical infrastructure.

Another major drawback of the existing systems is their commercial nature—many popular image editing or stylization tools require subscriptions, licensing fees, or proprietary software, which can be financially restrictive for students, educational institutions, or NGOs.

# PROPOSED SYSTEM

The proposed system addresses the limitations of the existing tools by introducing a lightweight offline image processing application capable of performing both edge detection and pencil sketch conversion efficiently on any standard computer system. Designed using Python and the OpenCV library, this tool focuses on accessibility, speed, and simplicity while maintaining high-quality visual output.

.

standard hardware, requiringnodedicated graphics card or advanced processing unit.

10

The system is designed with a simple command-line interface (CLI) that allows users to easily load images, apply desired transformations, and save the processed results. Its modular architecture ensures scalability, enabling future integration of advanced features such as graphical interfaces, AI- based stylization, and mobile deployment.

By combining technical efficiency with social purpose, this project serves multiple goals promoting digital creativity, educational learning, and cultural heritage preservation. It offers an inclusive, open-source platform that makes computer vision accessible to everyone, regardless of their technical background or available resources.

# Advantages of the Proposed System

The proposed system addresses these issues by providing a lightweight, offline, and open-source solution for transforming images into pencil sketch and edge-detected

formats using Python and OpenCV. Unlike online systems, it operates completely offline, eliminating the need for internet connectivity and making it ideal for rural schools, remote institutions, and low-resource environments. Because it is implemented using efficient and optimized image processing algorithms, it can run smoothly even on basic computer configurations, without requiring high-end graphics hardware. The offline nature of the tool ensures that all image data is processed locally on the user’s system, providing enhanced data security and privacy for personal or cultural visuals.

# IMPLEMENTATION ENVIROMENT

11

## SOFTWARE REQUIREMENT

* + - * Windows 10 or11
      * PythonIDE
      * OpenCV Library
      * NumPyLibrary
      * AnacondaEnvironment

## HARDWARE REQUIREMENT

* + - * Processor:Intel i5 or above
      * Memory(RAM):16 GB
      * HardDrive:32 GB
      * NoDedicated GPUrequired

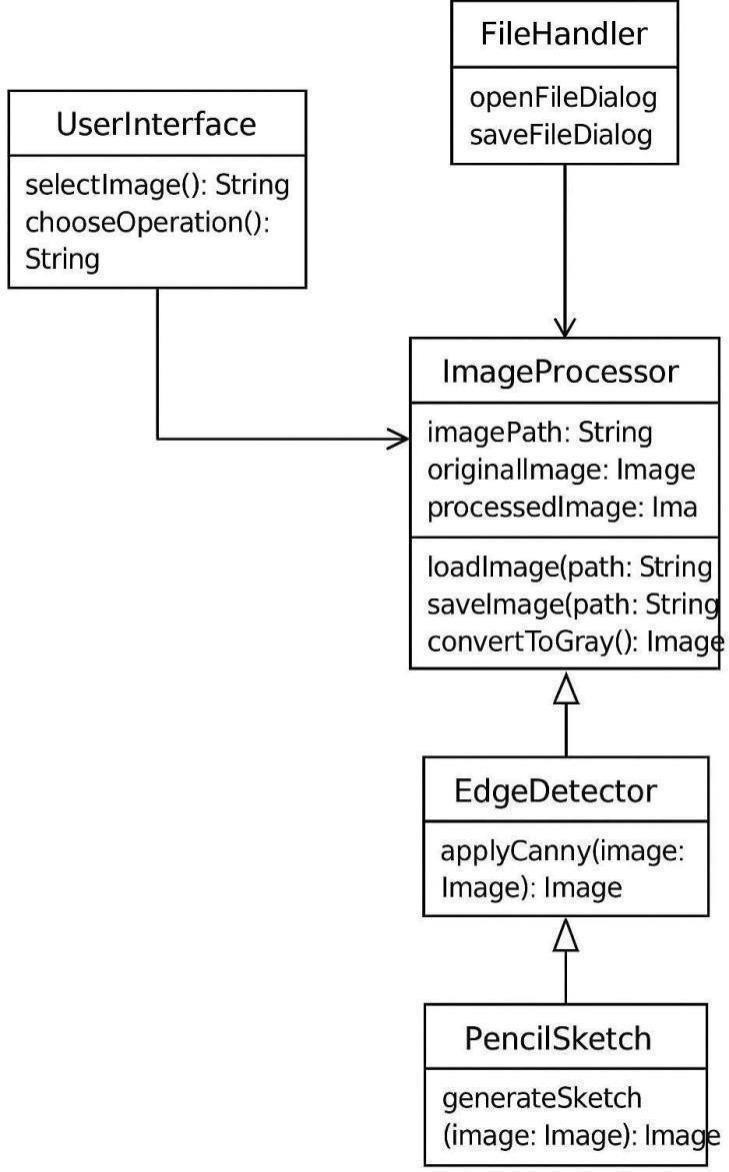
**CHAPTER3**

**SYSTEM DESIGN**

* 1. **UML DIAGRAMS**

## CLASS DIAGRAM

The system is designed using a modular architecture to ensure clarity, maintainability, and scalability. At the core of the system is the Image Processor class, which orchestrates the entire workflow by coordinating the sequence of operations, managing inputs, and directing outputs. Central to the image transformation process is the Pencil Sketch class, which encapsulates the logic required to convert a regular image into a pencil sketch. This class performs a series of image processing steps in sequence, including loading the image, converting it to grayscale, inverting the image, applying Gaussian blur, and finally blending the images to produce the sketch effect. Supporting classes such as Image Loader and Output Manager handle the input and output operations respectively, ensuring images are correctly loaded and the processed results are properly saved or displayed. The relationships between these classes are primarily associations, with the Image Processor utilizing the services of Pencil Sketch, Image Loader, and Output Manager to execute the workflow. The sequential operations within Pencil Sketch ensure that each step prepares the image for the next, resulting in a visually appealing pencil sketch. This modular approach not only simplifies testing and debugging but also allows for easy integration of additional effects in the future, such as edge detection by extending the system.



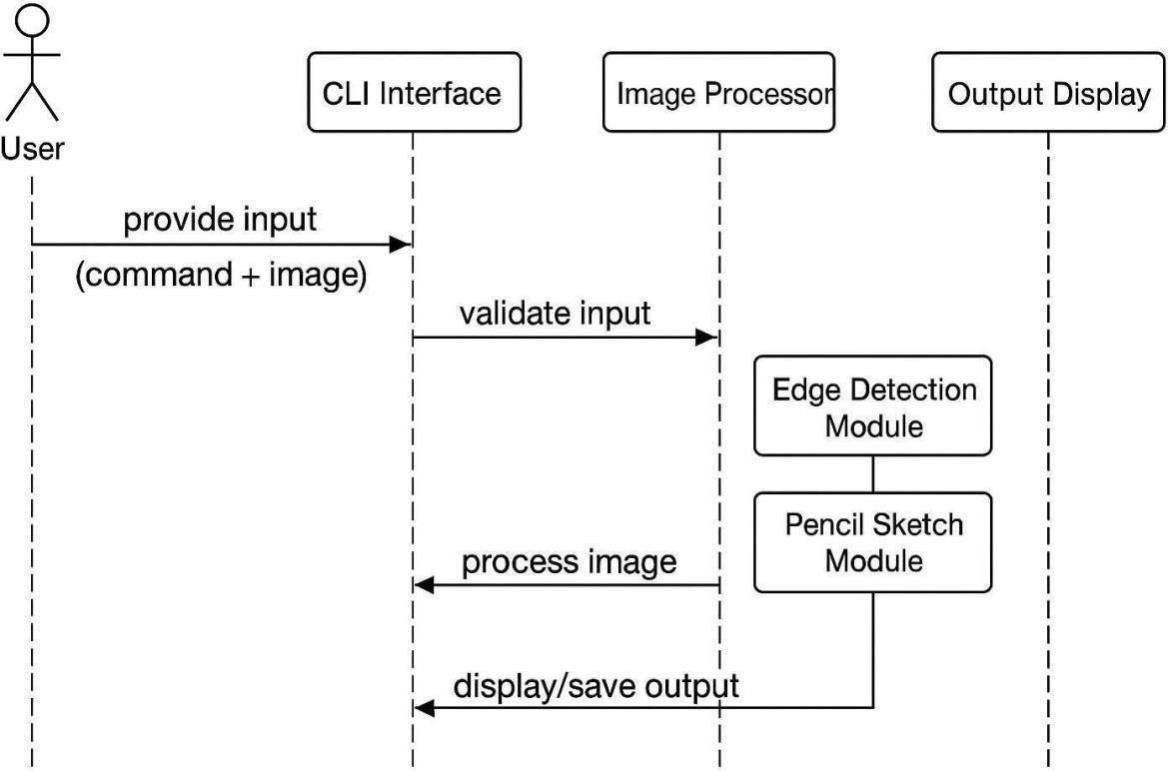
13

**Fig: 3.1.1. Class diagram forPencil Sketch**

## SEQUENCE DIAGRAM

The sequence diagram provides a comprehensive view of the dynamic interactions among the user, the command-line interface (CLI), the image processor, and the output display in the proposed system, highlighting how data and control flow through the various components. The process begins with the user initiating a command through the CLI, providing the input image and specifying the desired operation, such as pencil sketch generation or edge detection. The CLI is responsible not only for capturing the user input but also for validating it, checking the image format, file path, and command parameters to ensure correctness before forwarding the request to the Image Processor. The Image Processor acts as the central orchestrator, determining which specific module to invoke based on the user’s choice and managing the execution sequence of the processing steps. When a module such as the pencil sketch or edge detection is invoked, it executes a defined series of image transformations, including loading the image, applying grayscale conversion, inverting colors, applying filters, and performing blending operations in the case of pencil sketching. Throughout this process, the system maintains clear separation of responsibilities, allowing each module to handle its specific task while the Image Processor oversees the overall workflow. Once the image processing is complete, the processed image is sent back to the CLI, which then handles the presentation layer by either displaying the result to the user or saving it to the designated storage location..

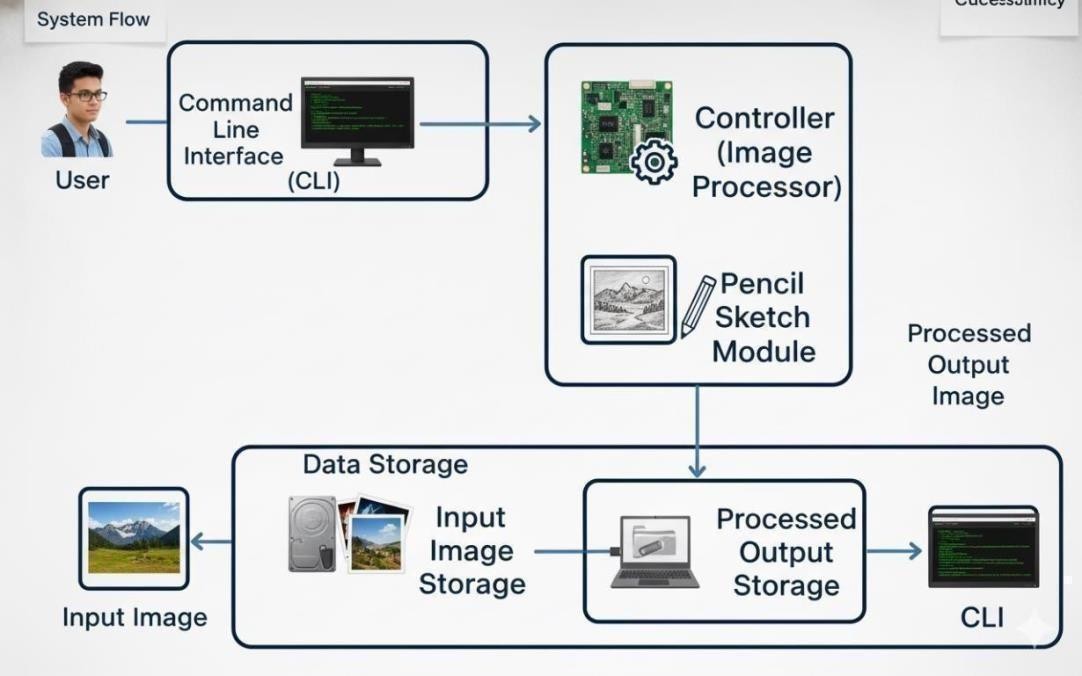




**Fig: 3.1.2 Sequence diagram for pencil sketch**

# CHAPTER 4 SYSTEM ARCHITECTURE

* 1. **ARCHITECTURE OVERVIEW**

****

**Fig: 4.1.1. System Architecture for Pencil Sketch**

The architecture of the project is designed in five layers. The user interacts through the CLI by providing the input image and selecting an operation. The controller receives this input, validates it and directs the request to the appropriate image processing module. The processing layer contains two modules: edge detection and pencil sketch, which apply the required transformations to the image. The data layer manages both the input image and the processed output. Finally, the result is displayed.

The architecture of the proposed system is organized into five distinct layers, each performing a specific function that contributes to the overall efficiency and reliability of the application. The User Interaction Layer serves as the primary interface between the user and the system. It operates through a Command Line Interface (CLI), where the user provides input commands to select operations such as pencil sketch conversion or edge detection. The user also specifies the image file path, and once the processing is completed, the resulting image is displayed or saved through the same interface.

The Application or Controller Layer functions as the central coordinating component that manages the flow of control between the user interface and the processing modules. It interprets the user’s commands, validates input parameters, and determines which image processing module needs to be executed. This layer also manages error handling, ensuring that invalid inputs or unexpected runtime issues are appropriately handled without interrupting the overall functionality of the system.

The Processing Layer is the core of the architecture, where the actual image transformation operations are performed. It comprises two main modules: the Pencil Sketch Module and the Edge Detection Module. The Pencil Sketch Module processes the input image by converting it into grayscale, inverting its pixel values, applying Gaussian blur to soften the tones, and finally blending the layers to produce a realistic hand-drawn sketch effect. The Edge Detection Module, on the other hand, utilizes algorithms such as the Canny or Sobel filters to detect edges and highlight structural outlines within the image, creating a simplified, line- art-style representation.

The Data Layer is responsible for managing all input and output data within the system. It handles the storage and retrieval of the original input images as well as the processed results generated by the image transformation modules. This ensures organized data management and facilitates easy access to both raw and processed images for future use.

Finally, the Output Layer presents the results of the processing stage to the user. The transformed images, whether pencil sketches or edge- detected outputs, are displayed through the command-line interface and can be saved locally in standard image formats. This layer ensures that the final outputs are easily viewable and exportable, completing the end- to- end workflow of the system.

The selected module, located within the processing layer, then performs the required operations on the image data stored in the data layer. Once the transformations are complete, the output layer retrieves the final processed image and delivers it back to the user through the interface, either for display or for saving in the chosen output directory.

#### MODULEDESIGNSPECIFICATIONPENCIL SKETCH

##### Image Input Module

The Image Input Module is responsible for accepting images from the user and preparing them for processing. It supports standard formats such as JPG and PNG and performs validation to ensure the file exists, is of a supported type and meets resolution requirements. This module acts as the entry point for the system and guarantees that the input image is ready for subsequent modules, preventing errors during processing.

##### Preprocessing Module

The Preprocessing Module prepares the input image by converting it into a grayscale format and applying Gaussian smoothing to reduce noise. These steps simplify the image data and enhance the clarity of important details, which improves the effectiveness of both the pencil sketch and edge detection modules. By cleaning and standardizing the image, this module ensures consistent and high-quality outputs.

##### Pencil Sketch Module

The Pencil Sketch Module generates a stylized sketch from the preprocessed grayscale image. It inverts the grayscale image and applies color dodge blending to produce an effect that resembles a hand-drawn pencil sketch. This module is crucial for creating visually appealing outputs suitable for education, creative work and cultural heritage preservation.

##### Edge Detection Module

The Edge Detection Module identifies and highlights the structural boundaries within the image using the Canny edge detection algorithm. By detecting areas of high intensity change, it produces a simplified representation that preserves the essential contours of objects. This module is particularly useful for analyzing and documenting .

##### OutputModule

The Output Module handles the display and storage of processed images. It allows users to view pencil sketch and edge-detected results in real-time and save them locally in standard formats such as PNG or JPG. This module ensures that users can retain processed images for documentation, education, or creative purposes, providing both convenience and usability.

##### Error Handling Module

The Error Handling Module ensures that the system operates reliably by monitoring for invalid inputs, missing files, or runtime exceptions. It generates clear messages to inform users of issues and prevents the processing modules from failing due to unexpected errors. This module is essential for maintaining robustness and enhancing user experience, particularly in offline or low-resource environments.

##### User Interface (UI) Module

The User Interface Module provides an interactive platform for users to engage with the system. It offers simple controls for uploading images, choosing between pencil sketch and edge detection options, adjusting sketch intensity, and previewing results in real-time. This module improves accessibility and ensures that both novice and experienced users can operate the system efficiently without needing technical knowledge.

##### Settings& CustomizationModule

The Settings & Customization Module allows users to modify processing parameters, such as pencil stroke intensity, edge detection thresholds, and output resolution. By providing customization options, this module enables personalized outputs tailored to user preferences. It enhances the versatility of the system, making it suitable for diverse applications, including artistic

creation, academic projects, and professional documentation.

24

# CHAPTER 5

## SYSTEM IMPLEMENTATION SAMPLE CODING

# Offline Pencil Sketchand Edge Detection Tool

# Developed using Pythonand OpenCV forcultural image preservation

importcv2 import os

# Function to load all images from a folder def

load\_images\_from\_folder(folder\_path):

images = [] filenames = []

for filename in os.listdir(folder\_path): img\_path = os.path.join(folder\_path, filename) image = cv2.imread(img\_path)

if image is not None: images.append(image) filenames.append(filename) return images, filenames

# Function to detect edges using Canny Edge Detection def detect\_edges(image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) edges = cv2.Canny(gray, 100, 200)

returnedges

# Function to convert an image into pencil sketch format def pencil\_sketch(image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) inverted = cv2.bitwise\_not(gray)

blurred=cv2.GaussianBlur(inverted, (21, 21),

0) inverted\_blurred = cv2.bitwise\_not(blurred)

sketch= cv2.divide(gray, inverted\_blurred, scale=256.0) return sketch

# Function to process all images ina folder

def process\_images\_in\_folder(folder\_path, output\_folder): os.makedirs(output\_folder, exist\_ok=True)

edge\_folder= os.path.join(output\_folder,

"Detected\_edges") sketch\_folder = os.path.join(output\_folder, "Pencil\_sketch")

os.makedirs(edge\_folder, exist\_ok=True) os.makedirs(sketch\_folder, exist\_ok=True)

images, filenames = load\_images\_from\_folder(folder\_path) if not images:

print("No images found in the specified folder.") return

for img, filename in zip(images, filenames): detected\_edges = detect\_edges(img) pencil\_sketch\_img = pencil\_sketch(img)

cv2.imwrite(os.path.join(edge\_folder, f"edges\_{filename}"), detected\_edges) cv2.imwrite(os.path.join(sketch\_folder, f"sketch\_{filename}"), pencil\_sketch\_img)

print(f"Processed and saved: {filename}") print("Processingcomplete. Checkthe output folder forresults.")

# Main function to start the program def main():

folder\_path = input("Enter the path of the folder containing images: ") output\_folder = input("Enter the path for the output folder to save results: ") process\_images\_in\_folder(folder\_path, output\_folder)

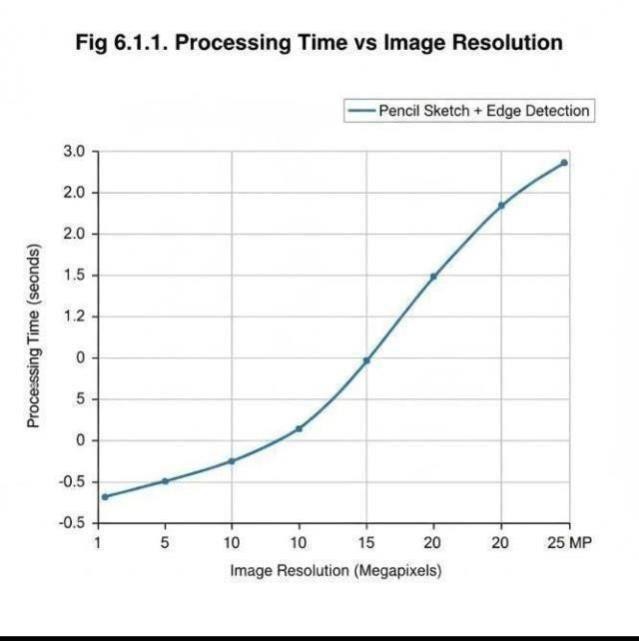
# Entry point of the program if name

== " main ": main()

# CHAPTER 6 SYSTEMTESTING

### 6.1. PERFORMANCE PARAMETERS

##### Process Time:

****

**Fig6.1.1. ProcessingTime vs Image Resolution:**

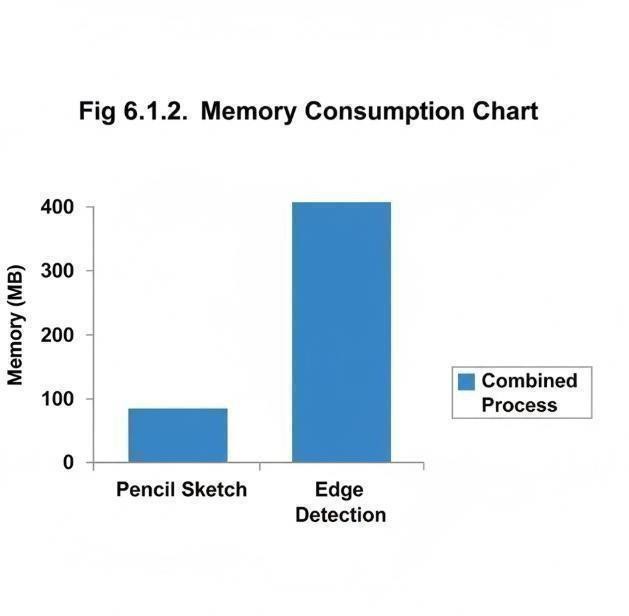
In the proposed system, the process begins when an image is provided through the user interface or command-line interface (CLI). The image first undergoes validation and preprocessing steps, such as resizing, color space conversion, and format checking. These preliminary operations, while relatively lightweight, contribute slightly to the total processing time. Once the image is ready, it is passed to the Pencil Sketch module or the Edge Detection module, depending on the user’s selection. In the pencil sketch module, operations like grayscale conversion, inversion, Gaussian blur, and blending are performed sequentially. Each of these operations has a time complexity that scales with the number of pixels in the image, making high-resolution images more

computationally intensive to process. Similarly, in the edge detection module, algorithms such as Canny edge detection or Sobel filtering involve gradient computation and convolution operations, which also depend on the image size..

##### MemoryUsage:

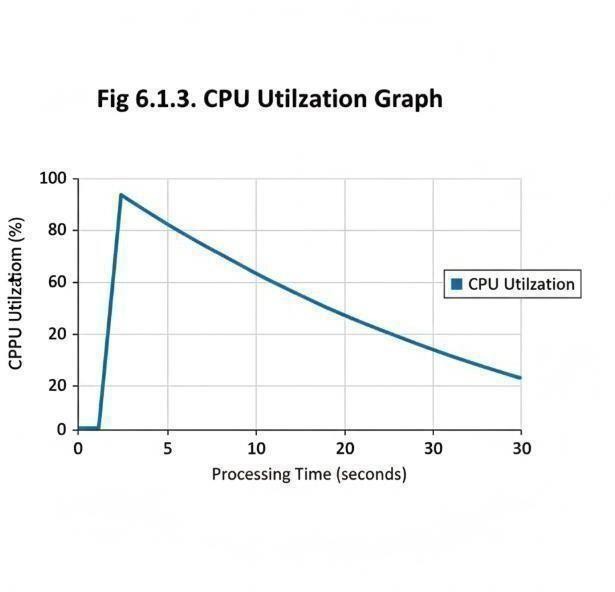
The amount of RAM consumed while performing image processing tasks. Lower memory usage indicates a lightweight system suitable for offline use. During image processing, memory is primarily utilized to store the original input image, intermediate processed images, and the final output. For instance, in the pencil sketch module, multiple copies of the image are maintained at different stages, including the grayscale version, the inverted image, the blurred image, and the final blended sketch. Similarly, in edge detection, intermediate matrices such as gradient maps, edge maps, and filter results occupy additional memory. The total memory usage depends not only on the image resolution but also on the internal representation of image data, such as whether images are stored in 8-bit, 16-bit, or floating-point formats. Higher resolution images naturally require more memory due to the larger number of pixels and intermediate matrices that need to be held in RAM simultaneously.

Monitoring memory usage is essential for designing an optimized system. By tracking how much RAM each module consumes, developers can identify potential memory bottlenecks and implement optimizations, such as in-place operations, efficient data structures, or selective memory deallocation after intermediate steps.



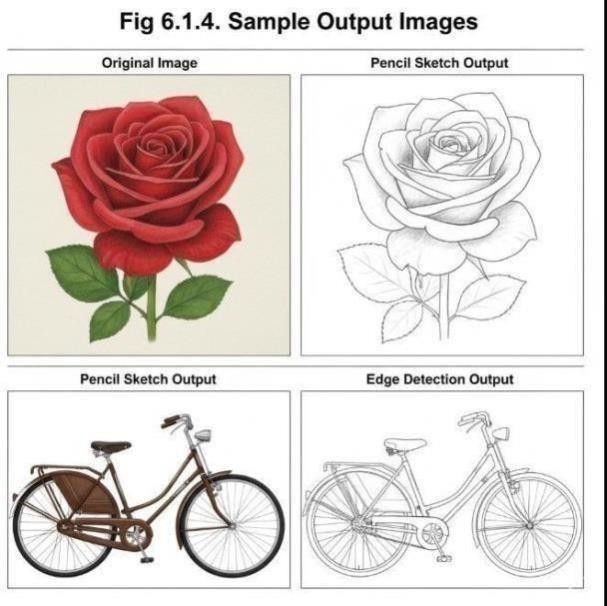
##### Fig 6.1.2. Memory Consumption Chart CPU Utilization Graph:

In digital image processing, an edge is not a hard line but rather a sudden and significant change in pixel intensity or color. Think of it as a steep slope on a 3D terrain map, where the height of each point is the pixel's brightness. The goal of edge detection algorithms is to mathematically identify these steep "slopes." In digital image processing, an **edge** is not represented by a sharp line but rather by a sudden and significant change in pixel intensity or color. Conceptually, an image can be imagined as a 3D terrain, where each pixel’s brightness corresponds to the height at that point. Edges represent the steep slopes on this terrain. The primary objective of edge detection algorithms is to mathematically identify these steep “slopes” by analyzing the changes in intensity between neighboring pixels. Common techniques, such as the Sobel, Prewitt, and Canny algorithms, perform convolution operations that calculate gradients across the image to highlight areas of significant intensity change. These gradient calculations are highly parallelizable but still impose a noticeable load on the CPU.



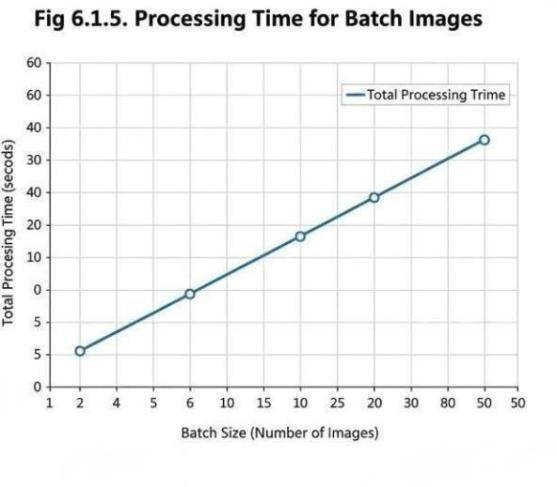
**Fig 6.1.3. CPU Utilization Graph:**

##### Sample Output Images:



**Processing Time for Batch Images:**

Batch processing is the execution of a series of tasks without human intervention. The fundamental principle is that the total processing time for a batch of images is the sum of the time required to process each individual image.



Batch processing refers to the automated execution of multiple image processing tasks in a single run without requiring user intervention for each image. It is commonly used when dealing with large sets of images that need to undergo similar operations such as resizing, filtering, sketch generation, or edge detection. The total processing time for a batch of images is the sum of the time taken to process each individual image. In other words, if each image takes a certain amount of time to process, the overall duration increases proportionally with the number of images in the batch. Several factors influence the total processing time, including image size and resolution, the complexity of the applied algorithms, and the computational power of the hardware being used. High-resolution images or advanced processing techniques like Gaussian blurring and edge detection generally take longer to execute. Additionally, the use of optimization methods such as parallel processing, efficient memory management, and GPU acceleration can significantly reduce processing time. Therefore, optimizing batch image processing involves balancing algorithm efficiency, hardware capability, and system performance to

achieve faster and more reliable results.

# 6.2 IMAGE DETECTION SYSTEM – TEST CASES

**Table 6.2: IMAGE DETECTION SYSTEM - TEST CASES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TestID Input** | **Test Scenario** | **Input** | **Expected Output** | **Pass/Fail Criteria** |
| TC-01 | Loadimage successfully | Any valid image file (JPG/PNG) | Image is loaded and displayed correctly | Image opens without error |
| TC-02 | Handle invalid file input | Corrupted or non-imagefile | System displays error message | Proper error message shown |
| TC-03 | Convert image to grayscale | Color image | Grayscal version of input image | Output image visually shows no color |
| TC-04 | Apply Gaussian blur | Grayscale image | Blurred versionof image for noise reduction | Visible smoothing without distortion |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC-05 | Perform edge detection | Blurred grayscale image | Edge-detected image highlighting contours | Edges clearly visible using Canny detection |
| TC-06 | Generate  pencil sketch | Color  image | Realistic  Pencil sketch output using blending | Fine line  details preserved; sketch appearance achieved |
| TC-07 | Low-light or  low-contrast image test | Dim or  shadowed image | Edges/sketch  still detectable | Accuracy ≥  85 % under low contrast |
| TC-08 | Maintain output structure | Different input image formats | Output stored with consistent naming and format | Output folder structured correctly |
| TC-09 | Time efficiency test | Single 1080p  image | Output generated within 2 seconds | Processing time ≤ 2 s |
| TC-10 | GUI / CLI  interaction | User  selects image or folder  via interface | System  processes user input smoothly | No input  /output crashes |

**CHAPTER 7**

**CONCLUSION AND FUTURE WORK**

# CONCLUSION

The Offline Pencil Sketch and Edge Detection Tool demonstrates an efficient and accessible approach to digital image processing using Python and OpenCV. The project successfully converts ordinary images into pencil sketch and edge-detected versions, providing a lightweight, offline alternative to complex or internet- dependent applications.

This tool highlights the power of basic computer vision techniques such as grayscale conversion, Gaussian blurring, Canny edge detection, and image blending, which together produce accurate and aesthetically pleasing outputs. Its simplicity, offline operation, and minimal hardware requirements make it suitable for students, educators, artists, and cultural organizations, especially in regions with limited access to high-end systems or internet connectivity.

The project aligns with global sustainability goals by contributing to SDG 4 (Quality Education), SDG 9 (Industry, Innovation, and Infrastructure), and SDG 11 (Sustainable Cities and Communities). It promotes digital learning, creative innovation, and cultural preservation through technology.

In conclusion, the project achieves its objective of creating a lightweight, open- source, and educational tool for image stylization and heritage preservation. It stands as an example of how computer vision can support creativity, education, and sustainability, while paving the way for future advancements such as AI-based enhancement, GUI integration.



30

# FUTURE ENHANCEMENT

The Offline Pencil Sketch and Edge Detection Tool lays a strong foundation for lightweight image stylization and cultural preservation. While the current system efficiently performs edge detection and pencil sketch conversion using traditional OpenCV methods, there is considerable scope for expanding its capabilities through future enhancements.

One major improvement would be the integration of a Graphical User Interface (GUI) using frameworks such as Tkinter or PyQt, enabling users to easily upload, preview, and save images without needing programming knowledge. This would make the tool more user-friendly and suitable for educational use.

Another potential enhancement involves adopting Machine Learning and Deep Learning models (like CNN-based edge detectors or generative sketch models) to produce more detailed, realistic sketches and adaptive edge detection under varying lighting conditions. Incorporating real-time video processing can further extend its use to live demonstrations, art visualization, and camera-based applications.

For broader accessibility, a mobile or web-based version can be developed using lightweight frameworks, allowing users to access the tool on smartphones or tablets. Additionally, features such as batch image processing, filter customization, and cloud backup for processed images can enhance usability for cultural archives and educational projects.

Lastly, integrating AI-driven colorization, style transfer, and noise reduction modules could transform this system into a complete creative suite for digital art and heritage conservation. Through these future enhancements, the project can evolve into a versatile, intelligent, and inclusive tool promoting creativity.



31

# CHAPTER 8 APPENDICES

# A1. SDG GOALS

##### SDG 4 – QualityEducation Goal Statement:

Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

##### Project Contribution:

The Offline Pencil Sketch and Edge Detection Tool promote quality education by providing students, educators, and researchers with a cost-free, offline learning resource to understand fundamental computer vision and image processing techniques.

Most educational institutions, especially in rural and under-resourced areas, lack access to high-end computers or reliable internet connections required to use modern online tools. This project bridges that gap by delivering a lightweight, open-source, and offline solution that can be easily used on standard systems without additional cost.

By simplifying complex visual processing operations like edge detection and sketch generation, the tool enhances practical learning in subjects such as Artificial Intelligence, Machine Learning, and Digital Image Processing. It also encourages self-learning and experimentation, allowing students to visualize algorithms in action and apply them to real-world cultural images.



32

##### SDG 9 – Industry, Innovation, and Infrastructure Goal Statement:

Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

##### Project Contribution:

This project embodies the spirit of innovation and sustainability through the development of a low-resource, efficient, and scalable image processing tool. Unlike commercial or cloud-based platforms that demand high computational resources, the proposed system demonstrates how modern technology can be optimized for accessibility and efficiency.

By using open-source libraries like OpenCV and Python, the tool provides a foundation for further research and industrial adaptation. It can serve as a prototype for small-scale image analysis systems, educational demonstrations, or creative industries seeking local digital solutions. The modular structure allows easy customization, promoting collaborative innovation among students, startups, and cultural organizations.

##### SDG 11 – Sustainable Cities and Communities Goal Statement:

Make citiesand human settlements inclusive, safe, resilient, andsustainable.

##### Project Contribution:

Preserving cultural heritage is a vital part of building sustainable and inclusive communities. The proposed offline sketch and edge detection system contributes directly to this goal by providing a digital preservation tool for artworks, monuments, and cultural visuals.

Through simplified and stylized representations, communities can document, archive, and share their traditional art and historical imagery in a digital form.



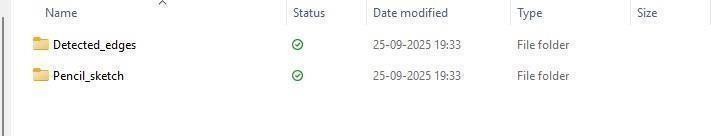
33

# A2. SAMPLE SCREENSHOTS



****

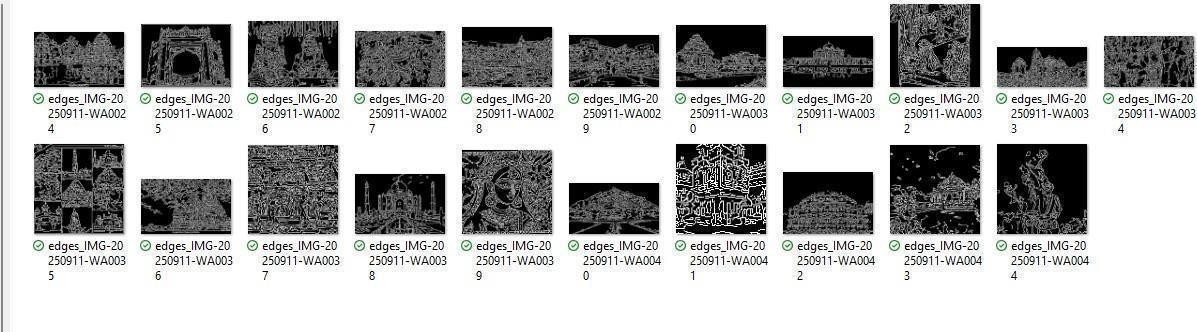
**Fig: A.8.1. Screenshot of Directory page**

****

**Fig: A.8.2. Screenshot ofImage folders**



**Fig: A.8.3. Screenshot ofpath setup**

****

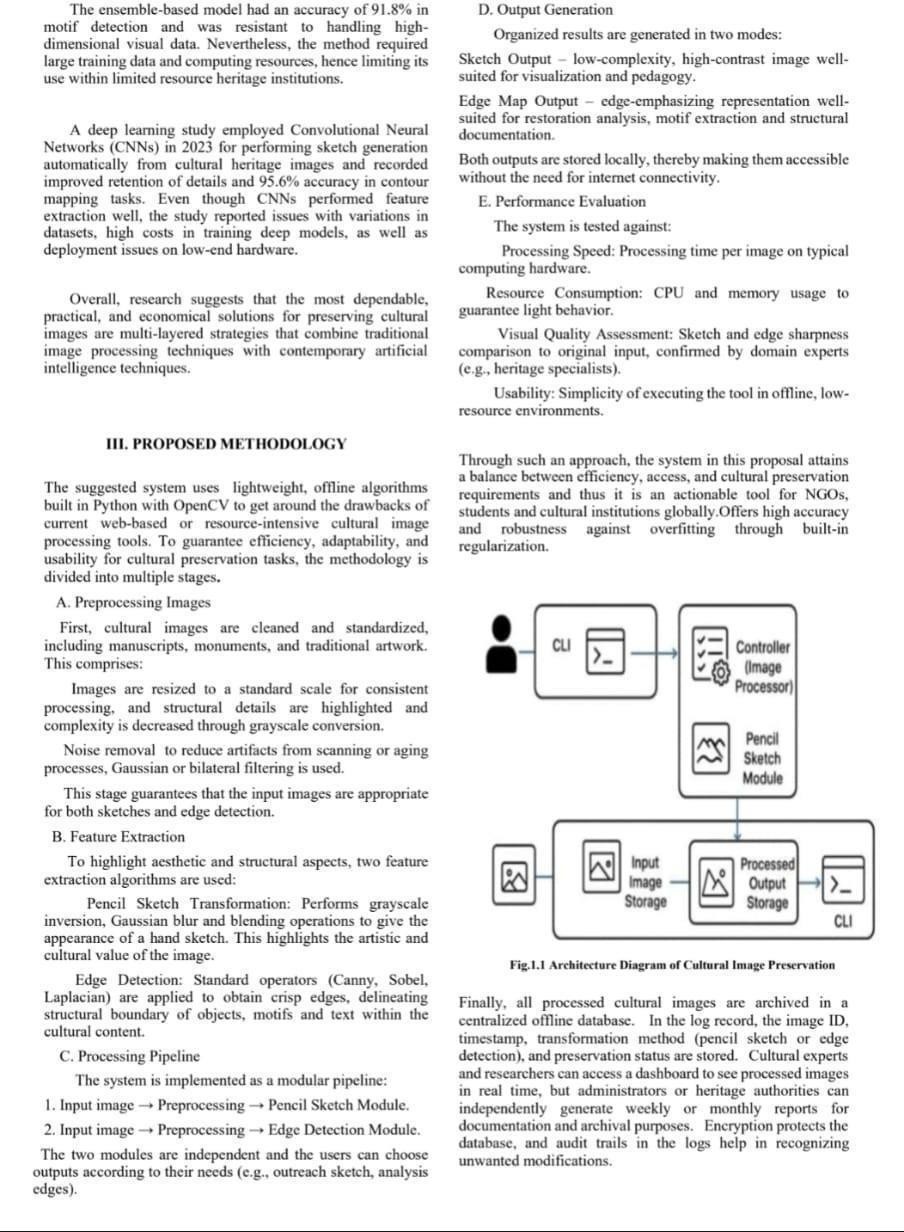
**Fig: A.8.4. Screenshot of Edge detection**

****

**Fig: A.8.5. Screenshot of pencil sketch**

**A3. Paper Publication**

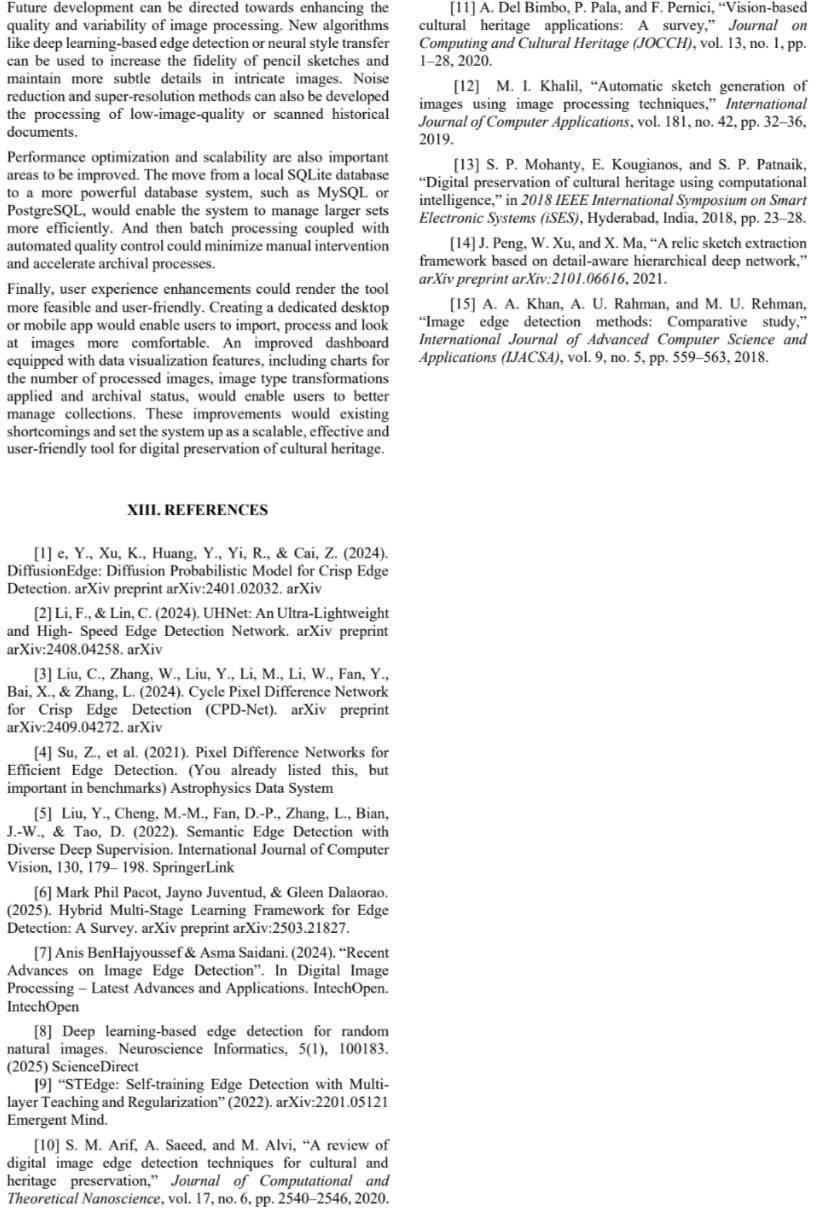
****





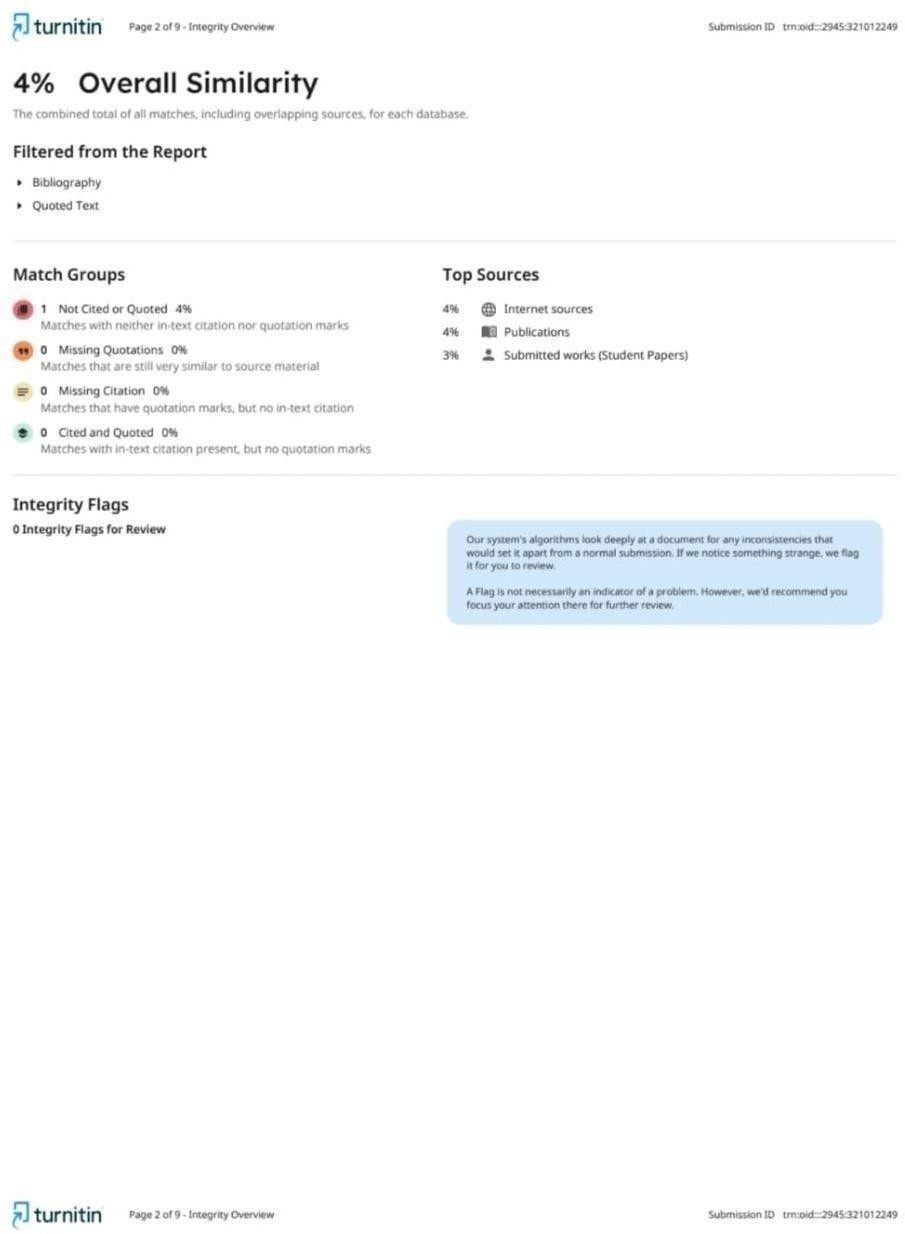


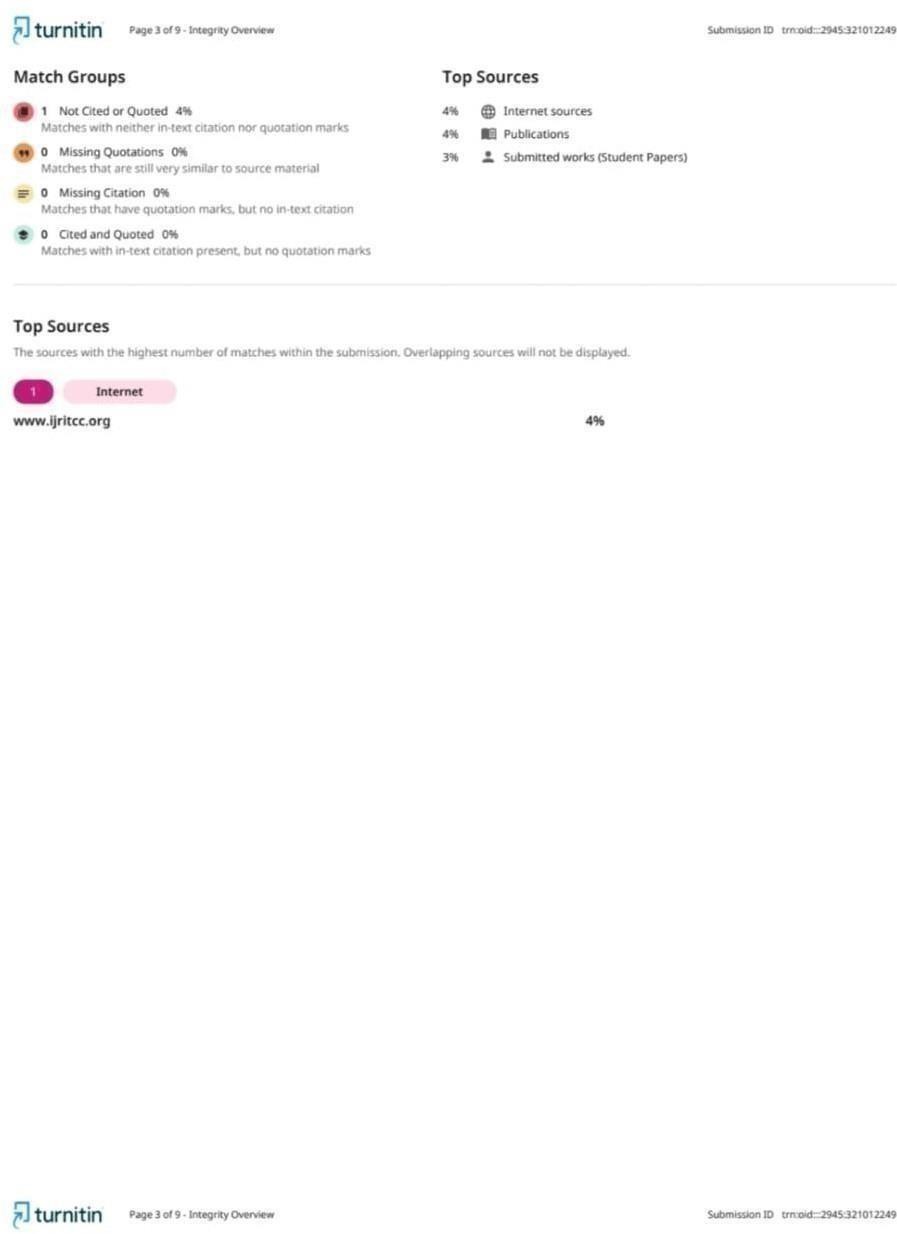




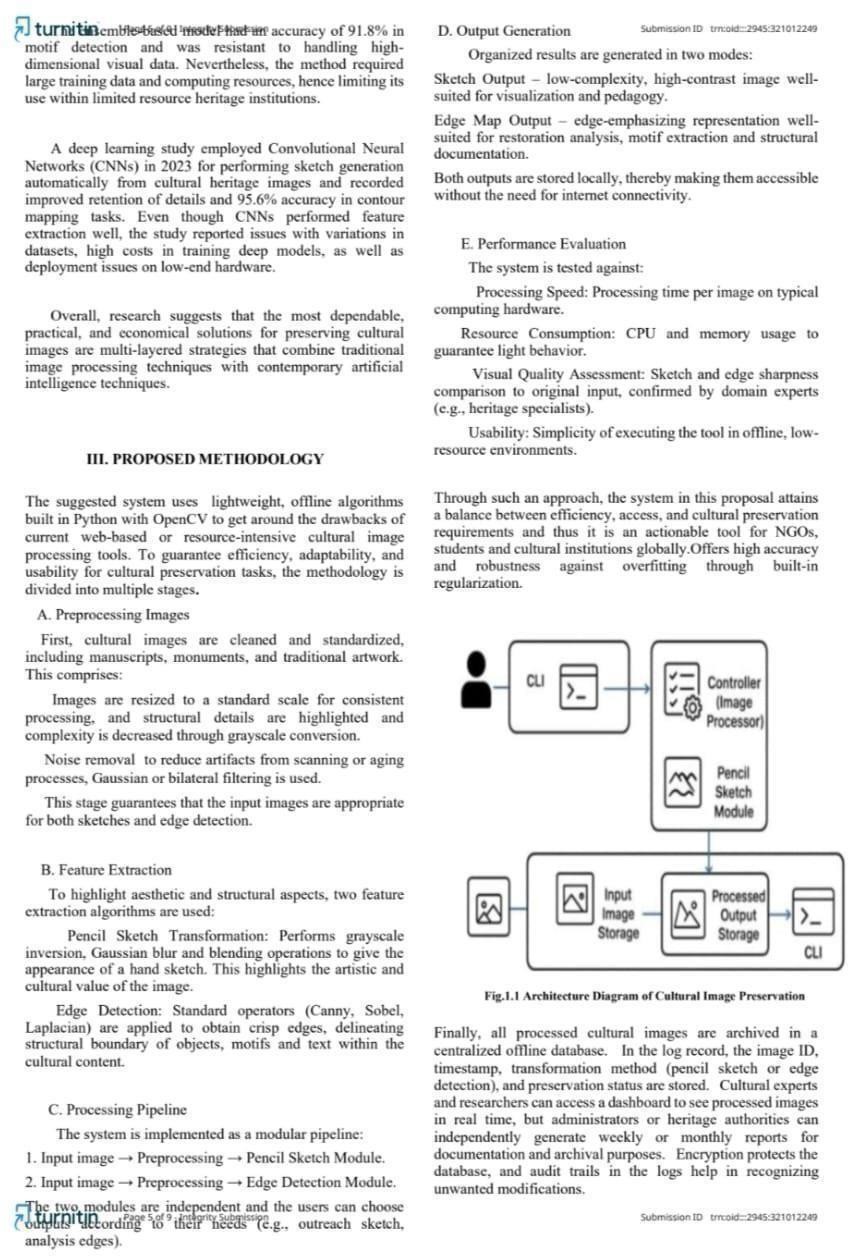
**A4. PLAGIARISM REPORT**

****





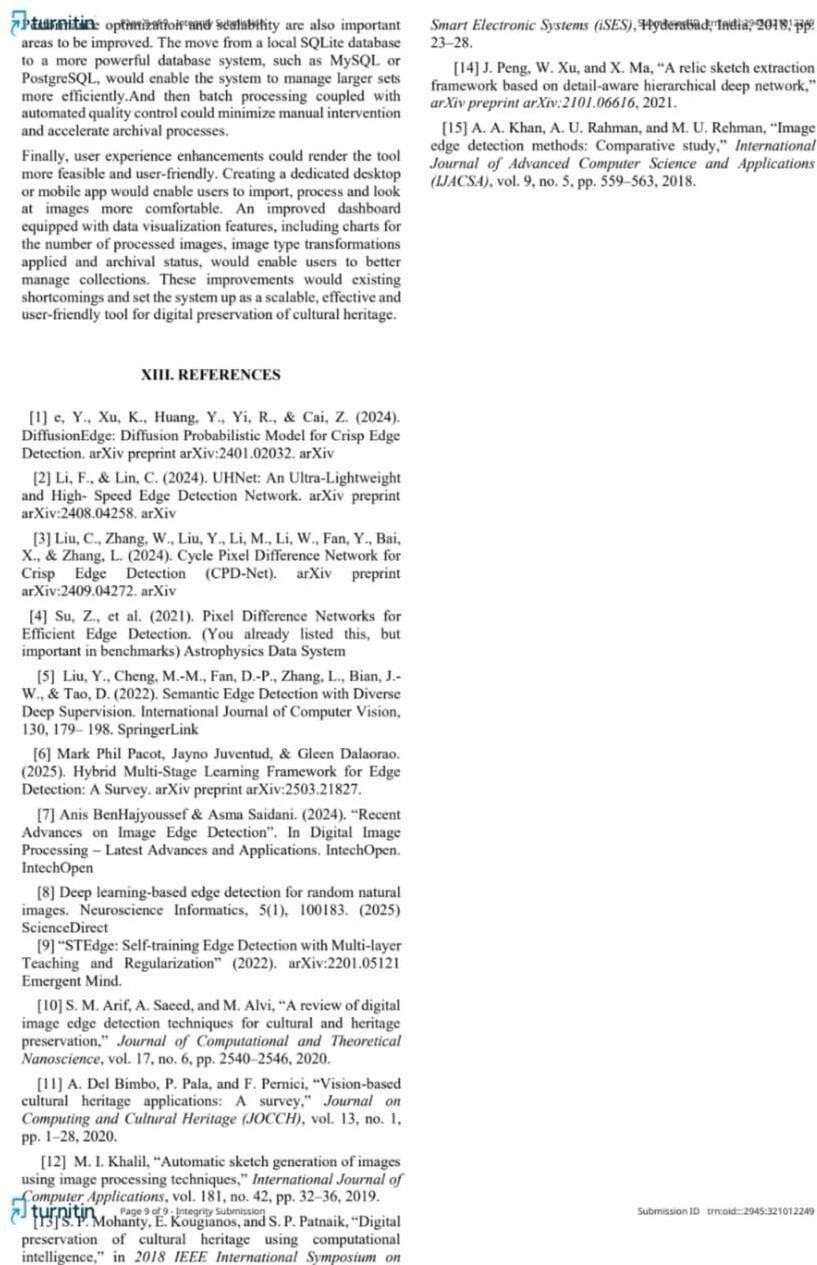












**CHAPTER 9 REFERENCES**

* e, Y., Xu, K., Huang, Y., Yi, R., & Cai, Z. (2024). DiffusionEdge: Diffusion probabilistic model forcrisp edge detection. arXivpreprint arXiv:2401.02032.
* Li, F., & Lin, C. (2024). UHNet: An ultra-lightweight and high-speed edge detection network. arXiv preprint arXiv:2408.04258.
* Liu, C., Zhang, W., Liu, Y., Li, M., Li, W., Fan, Y., Bai, X., & Zhang, L. (2024). Cycle Pixel Difference Network for crisp edge detection (CPD-Net). arXiv preprint arXiv:2409.04272.
* Su, Z., et al. (2021). Pixeldifference networks for efficient edge detection. Astrophysics Data System.
* Liu, Y.,Cheng, M.-M., Fan, D.-P., Zhang, L., Bian, J.-W., & Tao, D.(2022).

Semantic edge detection withdiverse deep supervision. International Journalof Computer Vision, 130, 179–198. Springer.

* Pacot, M. P., Juventud, J., & Dalaorao, G. (2025). Hybrid multi-stage learning framework for edge detection: A survey. arXiv preprint arXiv:2503.21827.
* Muntarina, K. (2025). Deep learning-based edge detection for randomnatural images. Neuroscience Informatics, 5(1), 100183. ScienceDirect.
* STEdge: Self-trainingedge detection with multi-layer teaching and regularization. (2022). arXiv:2201.05121. Emergent Mind.
* Hu, G. (2025). Amathematicalsurveyofimage deepedge detection. MDPI.
* Xu, Y., Khan, T. M., Song, Y., & Meijering, E. (2025). Edge deep learning in computer vision and medical diagnostics: A comprehensive survey. Artificial Intelligence Review, 58(3). Springer.
* Zangana, H. M., Mohammed, A. K., & Mustafa, F. M. (2024). Advancements in edge detection techniques for image enhancement: A comprehensive review. ResearchGate.
* Maksimovic, V. (2024). Comparative analysis of edge detection operators using a deep learning approach. PMC.
* Saha, S. (2025). Visiontransformers onthe edge: Acomprehensive survey. ScienceDirect.
* Liu, C., Wang, C., Dong, F., Xiao, X., Su, X., Zhu, C., Zhang, D., & Li, Q. (2024). Msmsfnet: A multi-stream and multi-scale fusion net for edge detection. arXiv preprint arXiv:2404.04856.
* Yang, C. (2024). Saliency and edge features-guided end-to-end network for salient object detection. ScienceDirect.
* Ngo, D. (2025). Edge intelligence: Areview ofdeep neural networkacceleration on edge platforms. MDPI.
* Maksimovic, V. (2024). Comparative analysis ofedge detection operators usinga deep learning approach. PMC.
* Tsirtsakis, P. (2025). Deep learning for object recognition: Acomprehensive review. ScienceDirect.
* Razak, W. M. R., Dak, A. Y., & Razak, T. R. (2024). A survey on various edge detection techniques in image processing and applied disease detection. Journal of Computing Research and Innovation, 9(2).
* Paramastri, T. A., & Mutiara, A. B. (2024). The role of edge detection in improving convolutional neural network accuracy and robustness: An overview. IEEE.