VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgavi-590 014

A Mini Project Synopsis Report
On
" PHOTO-TO-WATER-ART-MASTER"

Submitted in partial fulfillment of the requirements for the **Computer Graphics and Image Processing Laboratory (21CSL66)** course of the 6th semester

Bachelor of Engineering
In
Computer Science & Engineering

Submitted by

Diwakar N
(4AI21CS029)

Mohammed Muzamil Hussen
(4AI22CS407)

Under the guidance of

Mrs. Kavya T.M., B . E., M.Tech.,(Ph.D)
Asst. Prof, Dept of CS&E,
A.I.T., CHIKKAMAGALURU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY
CHIKKAMAGALURU -577102

(Affiliated to VTU, Belagavi and Approved by AICTE, New Delhi)
Chikkamagaluru- 577 102, Karnataka, India.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY
CHIKKAMAGALURU -577102

(Affiliated to VTU, Belagavi and Approved by AICTE, New Delhi)
Chikkamagaluru- 577 102, Karnataka, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"Photo-to-Water-Art-Master"** is a bonafide work carried out by **Diwakar N(4AI21CS029)**, **Mohammed Muzamil Hussen(4AI22CS407)** in partial fulfillment of the requirements for the **Computer Graphics and Image Processing Laboratory (21CSL66)** course of the 6th semester Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the academic year 2023-24. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the said degree.

Signature of the Guide	**Signature of the HOD**
Mrs. Kavya T.M., B.E.,M.Tech.,(Ph.D)	**Dr. Pushpa Ravikumar** B.E., M.Tech.,Ph,D
Assistant Professor	Professor & Head
Dept.of CSE,AIT	Dept.of CSE,AIT

External Examiner	**Signature with date**
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENTS

We express our humble Pranamas to his holiness **Parama Poojya Jagadguru Padmabushana Sri Sri Sri Dr. Balagangadharanatha Mahaswamiji** and **Parama Poojya Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Mahaswamiji** and also to **Sri Sri Gunanatha Swamiji,** Sringeri Branch, Chikkamagaluru who have showered their blessings on us for framing our career successfully.

We are deeply indebted to our honorable Director **Dr. C K Subbaraya** for creating the right kind of care and ambience.

We are thankful to our beloved Principal **Dr. C T Jayadeva** for inspiring us to achieve greater endeavors in all aspects of learning.

We express our deepest gratitude to **Dr. Pushpa Ravikumar,** Professor & Head, Department of Computer Science & Engineering for her valuable guidance, suggestions and constant encouragement without which success of our project work would have been difficult.

We are thankful to our guide **Mrs. Kavya T.M,** Asst. Professor, Dept. of Computer Science & Engineering, AIT, Chikkamagaluru, for inspiration and lively correspondence for carrying our project work.

We would like to thank our beloved parents for their support, encouragement and blessings.And last but not least, we would like to express our heartfelt thanks to all teaching and nonteaching staff of Computer Science & Engineering Department and our friends who have rendered their help, motivation and support.

Diwakar N (4AI21CS029)
Mohammed Muzamil Hussen (4AI22CS407)

ABSTRACT

In today's digital era, where visual content plays a crucial role in communication, the demand for creative and engaging image transformation tools is ever-increasing. This project presents the development of the Photo-to-Water-Art-Master App, a mobile application designed to convert ordinary images into water-art-style artworks. Utilizing advanced image processing and machine learning techniques, the app aims to provide users with a simple yet powerful tool for generating water-art versions of their photos.

The Photo-to-Water-Art-Master App leverages convolutional neural networks (CNNs) and image segmentation algorithms to accurately identify and transform the features of an image. The core technology involves edge detection, color quantization, and stylization processes that mimic traditional watercolor painting techniques. The app's user-friendly interface allows users to upload or capture images, apply various water-art filters, and adjust parameters to achieve the desired artistic effect.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In the realm of modern technology, image processing stands as a cornerstone with profound implications across numerous sectors. This project, titled **"Photo-to-Water-Art-Master",** endeavors to revolutionize the way digital images are enhanced, analyzed, and interpreted. By leveraging cutting-edge programming languages such as Python and Java, alongside powerful libraries like OpenCV and TensorFlow, this project integrates sophisticated algorithms for image enhancement, feature extraction, and object recognition. The outcomes are remarkable: superior image quality, precise object detection, and efficient processing capabilities. These advancements not only underscore the project's technical prowess but also open new horizons in fields like medical imaging, security, and multimedia, showcasing the transformative potential of advanced image processing

1.1 History of Image Processing

The history of image processing dates back to the early 20th century, rooted in the development of electronic imaging. It began with the invention of the television in the 1920s, which laid the foundation for capturing and manipulating visual data. The 1960s marked a significant leap with the advent of digital computers, enabling the development of the first digital image processing techniques. During this period, NASA played a pivotal role by using image processing to enhance moon photos transmitted by spacecraft.

The 1970s and 1980s saw the expansion of image processing applications into medical imaging, remote sensing, and industrial automation, driven by advancements in computer hardware and algorithms. The introduction of powerful software libraries like MATLAB and OpenCV in the 1990s and 2000s democratized access to image processing tools, spurring innovation and research. Today, with the rise of artificial intelligence and deep learning, image processing has reached new heights, enabling complex tasks such as facial recognition, autonomous driving, and real-time video analytics, underscoring its critical role in modern technology

1.2 Introduction to Image Processing

Image processing involves the manipulation and transformation of digital images to enhance their quality or extract meaningful information. This process typically begins with image acquisition, where an image is captured using devices like cameras or scanners and converted into a digital format. Following acquisition, preprocessing techniques such as noise reduction and contrast enhancement are applied to improve image quality and prepare it for further analysis.

Segmentation is a crucial stage in image processing, where the image is divided into different regions or objects of interest. Techniques like thresholding and edge detection help in isolating these regions, allowing for more detailed examination. Feature extraction then follows, where significant characteristics such as edges, textures, and shapes are identified and represented in a compact form suitable for analysis.

Once the features are extracted, image representation and interpretation take place. This involves converting the extracted features into models or patterns that can be analyzed to recognize and interpret the objects within the image. Techniques such as pattern recognition and machine learning are often employed to achieve accurate classification and understanding of the image content.

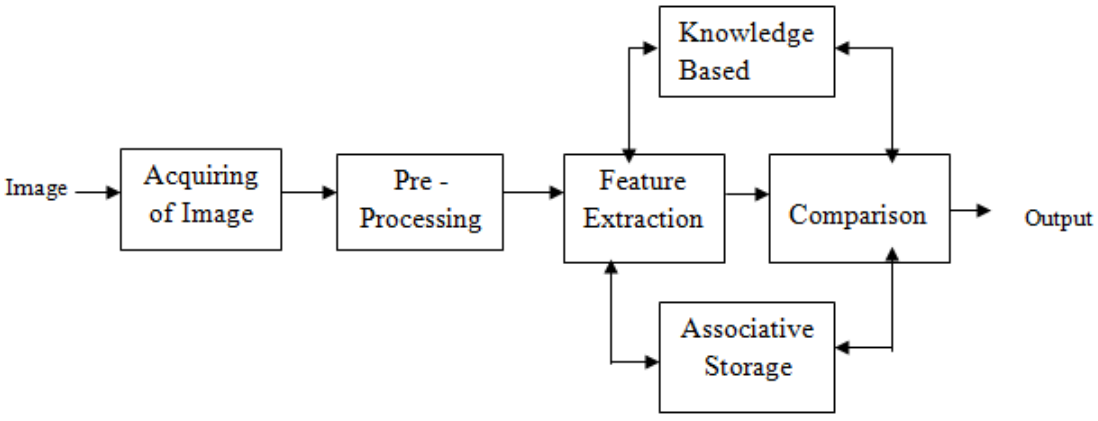


Fig 1.1: The Image Processing Block Diagram

Finally, post-processing is conducted to refine the results, enhancing specific details or correcting any remaining imperfections. This stage ensures the final image meets the desired quality and specifications. Image processing techniques are widely used across

various fields, including medical imaging, computer vision, and digital photography, providing essential tools for analyzing and improving visual data.

1.3 Objectives

Objectives of Photo-to-Water-Art-master using image processing are

- To develop an water art image tool.
- Implement advanced image processing techniques.
- Provide real-time preview functionality.
- Offer water styles image.
- Incorporate adjustable effect parameters.

1.4 Organization of the report

Chapter 1 introduces the fundamentals of image processing, covering essential concepts and techniques. In Chapter 2, all the image processing functions and algorithms used in our program are described in detail. Chapter 3 provides an overview of the project and its actual implementation, explaining how the various components are integrated to achieve the desired outcomes. Chapter 4 discusses the testing procedures and limitations of the program, highlighting any challenges encountered and how they were addressed. Chapter 5 concludes by offering suggestions for future enhancements and potential areas for further development in the field of image processing.

1.5 Summary

The chapter discussed before provides an overview of image processing, its history, and key concepts. It includes an introduction to various image processing techniques and the overall pipeline. The scope of study and objectives of the project are clearly defined, outlining the goals and expected outcomes. The organization of the report is designed to enhance readability and comprehension. In the upcoming chapter, the built-in image processing functions and algorithms used in the project source code are described in detail.

Chapter 2

A PREVIEW OF IMAGE PROCESSING FUNCTIONS

When we start processing images in Python, we need to import libraries such as OpenCV, NumPy, and Matplotlib. These libraries provide a wide range of functions and tools to perform image processing tasks.

Standard Libraries:

- **cv2**: This is the OpenCV library used for image processing. It contains a wide range of functions to perform operations like reading, writing, and manipulating images. Examples include cv2.imread(), cv2.imshow(), cv2.cvtColor(), etc.
- **numpy**: This library is used for numerical operations in Python. It provides support for arrays and matrices, along with a large number of mathematical functions. Examples include numpy.array(), numpy.zeros(), numpy.ones(), etc.
- **matplotlib.pyplot**: This is a plotting library used for creating static, animated, and interactive visualizations in Python. Examples include plt.imshow(), plt.plot(), plt.title(), etc.

The different image processing functions used in our project are described as follows:

- **Name: cv2.imread()**

  Python Specification: cv2.imread(filename, flags)

  Description: Loads an image from a specified file. The second argument specifies the color mode, such as grayscale or color.

- **Name: cv2.imshow()**

  Python Specification: cv2.imshow(window_name, image)

  Description: Displays an image in a window. The window will be created if it does not exist.

- **Name: cv2.cvtColor()**

  Python Specification: cv2.cvtColor(src, code)

Description: Converts an image from one color space to another. Common conversions include BGR to grayscale and BGR to HSV.

- **Name: cv2.GaussianBlur()**

  Python Specification: cv2.GaussianBlur(src, ksize, sigmaX)

  Description: Applies a Gaussian blur to an image, useful for reducing noise and detail.

- **Name: cv2.Canny()**

  Python Specification: cv2.Canny(image, threshold1, threshold2)

  Description: Detects edges in an image using the Canny edge detection algorithm.

- **Name: cv2.findContours()**

  Python Specification: cv2.findContours(image, mode, method)

  Description: Finds contours in a binary image. Contours are curves that join all continuous points along a boundary with the same color or intensity.

- **Name: cv2.drawContours()**

  Python Specification: cv2.drawContours(image, contours, contourIdx, color, thickness)

  Description: Draws contours on an image.

- **Name: numpy.array()**

  Python Specification: numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)

  Description: Creates an array from the given object.

- **Name: numpy.zeros()**

Python Specification: numpy.zeros(shape, dtype=float, order='C')

Description: Returns a new array of given shape and type, filled with zeros.

- **Name: plt.imshow()**

  Python Specification: plt.imshow(X, cmap=None, norm=None, interpolation=None, alpha=None, vmin=None, vmax=None, origin=None, extent=None, filternorm=1, filterrad=4.0, resample=None, url=None, data=None, **kwargs)

  Description: Displays an image using Matplotlib.

- **Name: plt.title()**

  Python Specification: plt.title(label, fontdict=None, loc='center', pad=None, **kwargs)

  Description: Sets the title of the current axes.

- **Name: plt.show()**

  Python Specification: plt.show(*args, **kw)

  Description: Displays all open figures.

- **Name: cv2.resize()**

  Python Specification: cv2.resize(src, dsize, dst=None, fx=0, fy=0, interpolation=cv2.INTER_LINEAR)

  Description: Resizes an image to the specified size.

- **Name: cv2.threshold()**

  Python Specification: cv2.threshold(src, thresh, maxval, type)

  Description: Applies a fixed-level threshold to each array element.

Chapter 3

IMPLEMENTATION

The implementation of the image processing application integrates OpenCV for image manipulation and Tkinter for the graphical user interface. The application allows users to select an image file using a file dialog. Once selected, the image is processed with techniques such as resizing, blurring, filtering, and sharpening. These processing steps are performed by OpenCV functions to enhance and clear the image. The final processed image is then displayed in the Tkinter window using the Pillow library for compatibility. The entire process is encapsulated in an ImageProcessor class, providing a seamless user experience.

```
import cv2

import numpy as np

import tkinter as tk

from tkinter import filedialog

class ImageProcessor:

def __init__(self):

self.root = tk.Tk()

self.root.title("imageprocessor")

# Create a label and button to select an input image

self.label = tk.Label(self.root, text="Select an input image:")

self.label.pack()

self.button = tk.Button(self.root, text="Browse", command=self.select_image)

self.button.pack()

# Create a label to display the output image

    self.output_label = tk.Label(self.root, text="Output Image:")

    self.output_label.pack()
```

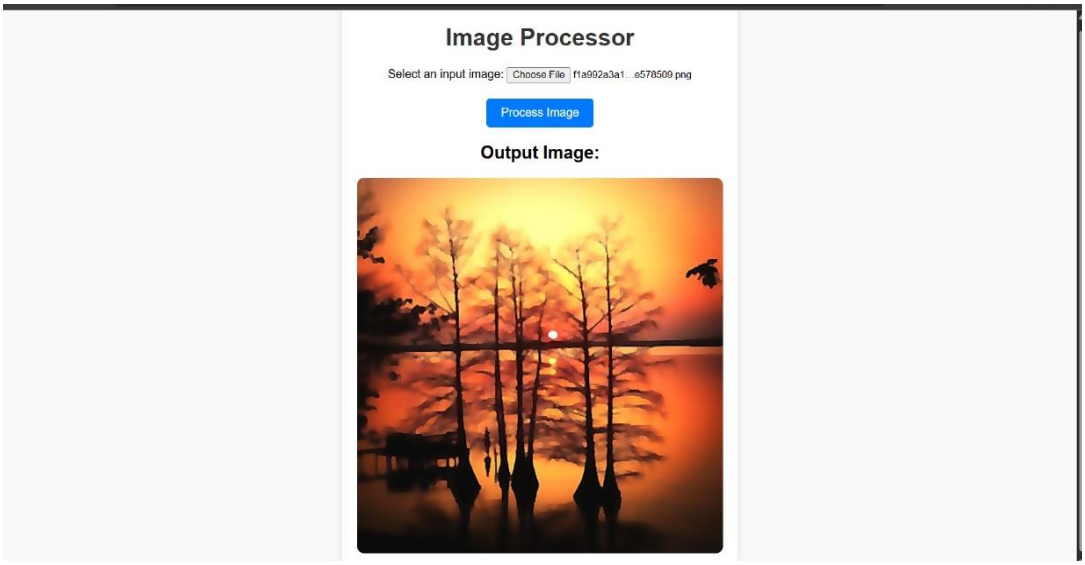
```
        self.output_image = tk.Label(self.root)
        self.output_image.pack()
        # Create a button to process the image
                self.process_button = tk.Button(self.root, text="Process Image", command=self.process_image)
        self.process_button.pack()
    def select_image(self):
        # Open a file dialog to select an input image
        self.filename = filedialog.askopenfilename(title="Select an input image")
        if self.filename:
            self.label.config(text="Input Image: " + self.filename)
    def process_image(self):
        # Read the input image
        image = cv2.imread(self.filename)
        # Resize the image
        image_resized = cv2.resize(image, None, fx=0.5, fy=0.5)
        # Remove impurities from the image
        image_cleared = cv2.medianBlur(image_resized, 3)
        image_cleared = cv2.medianBlur(image_cleared, 3)
        image_cleared = cv2.medianBlur(image_cleared, 3)
        image_cleared = cv2.edgePreservingFilter(image_cleared, sigma_s=5)


        # Apply bilateral filtering
        image_filtered = cv2.bilateralFilter(image_cleared, 3, 10, 5)
```

```
        for i in range(2):
            image_filtered = cv2.bilateralFilter(image_filtered, 3, 20, 10)
        for i in range(3):
            image_filtered = cv2.bilateralFilter(image_filtered, 5, 30, 10)
        # Sharpen the image
        gaussian_mask = cv2.GaussianBlur(image_filtered, (7, 7), 2)
        image_sharp = cv2.addWeighted(image_filtered, 1.5, gaussian_mask, -0.5, 0)
        image_sharp = cv2.addWeighted(image_sharp, 1.4, gaussian_mask, -0.2, 10)
        # Display the output image
        cv2.imshow('Final Image', image_sharp)
        self.output_image.config(image=cv2.imencode('.jpg', image_sharp)[1])
    def run(self):
        self.root.mainloop()
if __name__ == "__main__":
    app = ImageProcessor()
    app.run()
```

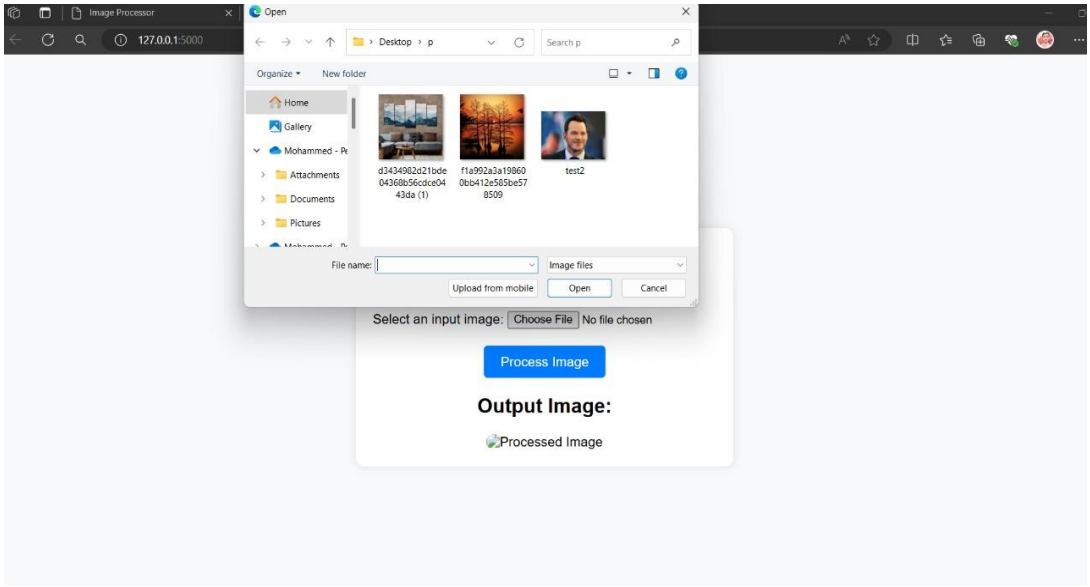
Chapter 4

RESULTS



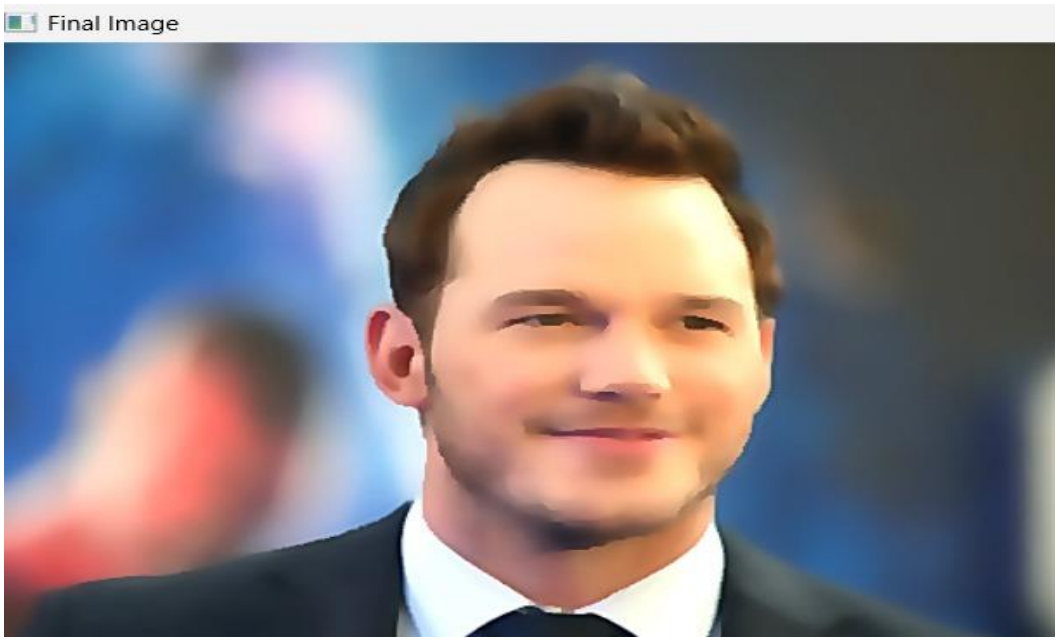
Snapshots 4.1: Home page

The above figure 4.1 represents the home page of the project



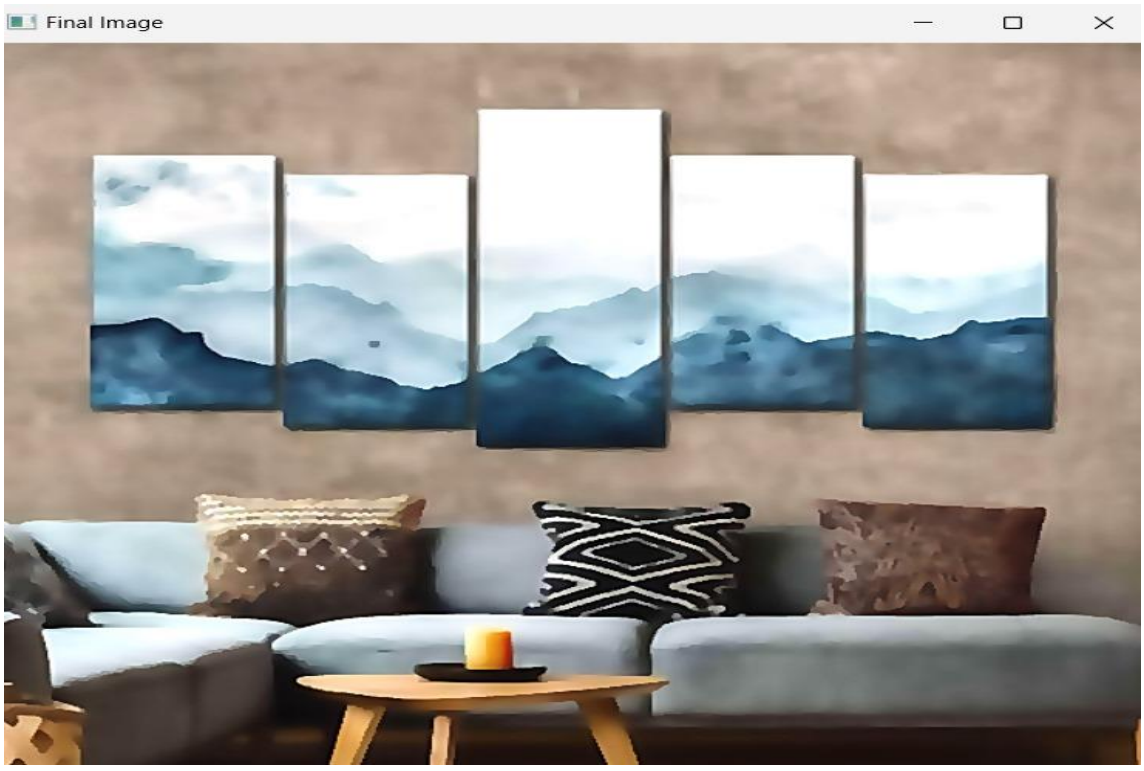
Snapshots 4.2: Image uploading

The above figure 4.2 represents the uploading of the image



Snapshots 4.3: Conversion of image into art image

The above figure 4.3 represents the art image



Snapshots 4.4: Downloaded water image

The above figure 4.4 represents the final output of the project

Chapter 5

CONCLUSION

The Photo-to-Water-Art-master stands as a powerful tool in the realm of digital creativity, transforming ordinary photos into delightful, -like images with ease. Its user-friendly interface, combined with advanced AI algorithms, ensures that users of all skill levels can produce professional-quality water images effortlessly.

BIBLIOGRAPHY

Learned Resources:

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd/4th Edition, Pearson Education,2011

2. James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: Pearson education

Learned E-Resources:

1. https://www.w3schools.com/python/

2. https://nptel.ac.in/courses/106/102/106102063/

3. https://chatgpt.com/

4. https://nptel.ac.in/courses/106/102/106102065/

5. https://www.tutorialspoint.com/opencv/

6.https://medium.com/analytics-vidhya/introduction-to-computer-vision-opencv-in python-fb722e805e8b