PROJECT REPORT

on

Cartoonify Image using OpenCV

Submitted in partial fulfillment of the requirements
for the award of degree

MASTER OF COMPUTER APPLICATIONS

of

KLE TECHNOLOGICAL UNIVERSITY

by

Mr. Karthik S Modak
(SRN: 01FE21MCA036)

Mr. Steven Manchala
(SRN: 01FE21MCA054)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
KLE TECHNOLOGICAL UNIVERSITY
Vidyanagar, Hubballi - 580031 Karnataka.
April- 2023

A Project Report on

Cartoonify Image using OpenCV

Submitted in partial fulfillment of the requirements
for the award of degree

MASTER OF COMPUTER APPLICATIONS

of

KLE TECHNOLOGICAL UNIVERSITY

by

Mr. Karthik S Modak
(SRN: 01FE21MCA036)

Mr. Steven Manchala
(SRN: 01FE21MCA054)

under the guidance of

Internal Guide:	**Internal Guide:**
Mrs. Deepa C Mulimani	Mr. Praveenkumar S.M
Asst. Professor	Asst. Professor
MCA Dept.	MCA Dept.

KLE TECH.

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
KLE TECHNOLOGICAL UNIVERSITY
Vidyanagar, Hubballi - 580031 Karnataka.
April- 2023

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
KLE TECHNOLOGICAL UNIVERSITY

CERTIFICATE

This is to certify that the project work entitled
"Cartoonify Image using OpenCV"

Submitted in partial fulfillment of the requirements
for the award of degree of
Master of Computer Applications
of
KLE Technological University, Hubballi, Karnataka
is a result of the bonafide work carried out by

Mr. Karthik S Modak
(SRN: 01FE21MCA036)

Mr. Steven Manchala
(SRN: 01FE21MCA054)

during the academic year 2022-2023

Mrs. Deepa C Mulimani	**Mr. Praveenkumar S.M**	**Dr. P. R. Patil**
Project Guide, Asst Professor MCA Dept.	Project Guide, Asst Professor, MCA Dept.	Professor, Head of Department, MCA Dept.

Viva-Voce Examination

Name of the Examiners	Signature with Date
1. ______________________________	:____________________
2. ______________________________	:____________________

ACKNOWLEDGEMENT

Every successful completion of any undertaking would be complete only after we remember and thank the almighty, the parents, the teachers, and the personalities, who directly or indirectly helped and guided during the execution of that work. The success of this work is equally attributed to all well-wishers who have encouraged and guided throughout the execution.

I express my deepest gratitude to **Mrs. Deepa C Mulimani** and **Mr. Praveenkumar S.M,** External guide for their guidance and assistance throughout the project with great interest.

I avail this opportunity to express deepest gratitude to **Dr. P. R. Patil, Professor & HoD** for encouraging us with his valuable tips.

I am grateful to **Prof. B. L. Desai,** Registrar for his blessings. And we are grateful to **KLE Technological University Hubballi** which has given us a bright future. I would like to thank all the people for their guidance and valuable suggestions throughout the project.

I also thank all teaching and non-teaching staff members of the MCA department for their invaluable cooperation.

Finally, I would like to express our sincere thanks to our **Parents and Friends** for their enormous encouragement and all other who have extended their helping hands towards completion of our project.

Mr. Karthik S Modak
Mr. Steven Manchala

Contents

ABSTRACT

Cartoonifying an image using OpenCV and Python involves utilizing the OpenCV library, an open-source computer vision and image processing software, and the Python programming language to transform a regular photograph into a cartoon-like representation. The process typically involves converting the image to grayscale, applying various image filters to extract edges and reduce noise, and then adding color back to the image. This can be achieved through a combination of techniques, like bilateral filtering, adaptive thresholding, and color quantization. Additionally, Python's NumPy and Matplotlib libraries can be used to manipulate and display the resulting image. Overall, cartoonifying images using OpenCV and Python is a powerful and customizable approach that allows for a wide range of creative possibilities. Cartoonifying an image using OpenCV and Python is a powerful and flexible approach that can be used for a variety of purposes beyond creating cartoon-like representations. The technique can be used for image segmentation, object detection, and feature extraction, among other applications.

Chapter 1

INTRODUCTION

Cartoons are illustrations of characters that can be imaginary or are based on a personality. They are semi-realistic or non-realistic, which are popular nowadays to depict a situation or a happening in a funny, satirical way. One of the earliest examples of traditionally animated movies, Fantasmagorie (1908), in which all the movie frames were hand-drawn, and till years the animation culture with hand-drawn frames. Walt Disney caught the race with their remarkable cartoon series and took the animation cartoons to the next level. Earlier, cartoonists used to draw these cartoons by hand, and when 'Anime' started catching the light, it became difficult for them to attract each movie frame by hand, which took a lot of time and was not reversible if met with a mistake.

In the present world a lot of applications are being built for applying different effects, filters and style to images and videos. So, due to this applications are being built based on current trends on the social media applications like Facebook, Instagram, Snapchat and others. Automation technology is booming nowadays. Human effort is reduced to the minimum; with a set of programs doing our job for us, animation technology has taken a considerable leap. Our job is to make those programs. So, this is the project which converts regular camera clicked images into cartoonized form, created using Python language and relative libraries. In times like today, where technology is taking giant leaps with great minds improving them day by day, the animation industry's main problem is the quality, with 4K, IMAX, and all other super high-definition movies catching popularity. Their camera being different exclusively, the animation of these movies while maintaining quality is a huge task. Toy Story 4, The Garden of Words being an excellent example for being one of the most detailed animated movies ever made. Anime movies take much longer than regular movies in times of VFX effects and animations. The workforce, skillset, and resources in terms of equipment and investment are more than classic movies, so time plays an essential role in creating these movies. For saving time, automation is the best way to go; with the improvement in machine learning and artificial intelligence, every production process has improved significantly and improves the quality of the product.

There are numerous ways to convert a specific photo to a cartoon one, but the best way is to do manual with software like Adobe Photoshop, and not everyone has their hands on tools like these, so the users there are limited. Although the output obtained through the manual process is the best, it is time-consuming, and doing it for a large number of images is not efficient at all. Therefore, systems like these are a go-go for every field as it provides output in seconds with upgrades and everything, the scale will surely improve with even better results.

With increasing technology, numerous software came into the market to digitally draw the illustrations, thus reducing human effort and were more efficient than being hand-drawn by artists. Here we will build one interesting application that will cartoonify the image provided to it. To build this cartoonifier application we will use Python and OpenCV. This is one of the exciting and thrilling applications. While building this application we will also see how to use libraries like easygui, Tkinter, matplotlib, numpy and others. Here you have to select the image and then the application will convert that image into its cartoon form. Mainly, we build this application using OpenCV and python as the programming languages.

Chapter 2

LITERATURE SURVEY

This uses a technique cartoonifying images using OpenCV and Python, which involves converting the image to grayscale, applying various filters to extract edges and reduce noise, and then adding color back to the image. The authors also suggest the use of adaptive thresholding and color quantization to achieve the desired effect.[1] This presents a real-time system for cartoonifying video streams using OpenCV and Python, which involves applying bilateral filtering and edge detection to the video frames, and then adding color back to the image. The authors also propose the use of machine learning techniques to automatically adjust the parameters based on the input video.[2] This compares different techniques for cartoonifying images using OpenCV and Python, including bilateral filtering, adaptive thresholding, and color quantization. The authors evaluate the techniques based on factors such as computational complexity, image quality, and processing time.[3] The methodologies used are The Algorithms, Identifying The Edges and Colour to the RGB Image. Accuracy is to provide the high accuracy model as compare with existing models. The main aim here is not to simply build another tool for users to turn their favorite images into cartoon drawings but to draw their attention towards the coding side of it and how easily one can make their own program after learning the basics.[4]

2.1 Proposed System

The goal of this project is to provide an application named “Cartoonify Image using OpenCV” to convert given image to a cartoonified image .

1. User Interface -The user interface will allow the user to select an image file and view the resulting cartoon image.
2. Image Saving- The system will allow the user to save the resulting cartoon image as a file in a variety of formats, including JPEG and PNG.
3. Support for different image formats- the Application works very well with many Image File Formats (tiff, png, jpg, jpeg and others).

Overall, the proposed system will provide a comprehensive solution for cartoonifying images using OpenCV and Python, offering a range of customization options and features to enhance the user experience.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

3.1 Introduction

The Software Requirement Specification is developed to address the user requirement for the development of an application. This document provides functional and non-functional requirements for “Cartoonify Image using OpenCV” application.

3.2 Purpose

The purpose of the "Cartoonify Image using OpenCV" project includes the development of an application that can transform an input image into a cartoon-like image using OpenCV in Python.

The application should have a user-friendly interface that allows users to select an input image, apply the cartoonifying effect, and save the output image. The application should also handle errors gracefully and provide informative error messages to the user.

3.3 Scope

The project “Cartoonify Image using OpenCV” is a python application, which is capable of selecting image from the computer and applying different cartoon filters and getting the output.

3.4 Software Requirements

- Operating system: Windows 7 or above
- IDE: Pycharm IDE, Visual Studio Code

- Programming language: Python 3.6
- Python Libraries: OpenCV, Tkinter, easygui, imageio, matpltlib, numpy

3.5 Hardware Requirements

- Processor: Core i3 and above
- RAM: 4GB and above
- Hard Disk Space: 40GB

3.6 Functional Requirements

- Load the input image in Python using OpenCV.
- Convert the input image to grayscale.
- Apply a median blur to the grayscale image.
- Apply an adaptive threshold to the blurred image.
- Apply a bilateral filter to the original image.
- Combine the bilateral image with the thresholded image to create a cartoon-like effect.
- Save the cartoonified image to a file.

3.7 Non-Functional Requirements

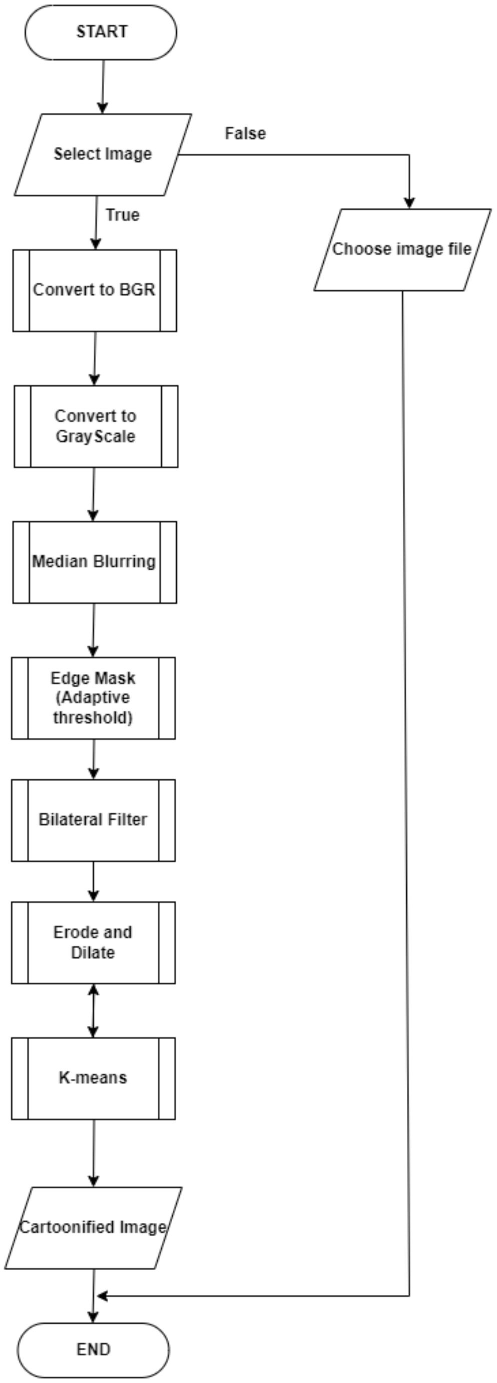
- **Performance**: The algorithm should be optimized for fast execution and should be able to handle images of different sizes and formats.
- **Accuracy**: The cartoonified image should retain the important features of the original image and should be visually appealing.
- **User interface**: The application should have a user-friendly interface with easy-to-use controls for selecting input images and saving output images.
- **Compatibility**: The application should be compatible with different operating systems and versions of Python and OpenCV.
- **Error handling**: The application should be able to handle errors gracefully and provide informative error messages to the user.
- **Security**: The application should not compromise the security of the user's system or data.
- **Maintainability**: The code should be well-structured and documented for easy maintenance and future updates.

Chapter 4

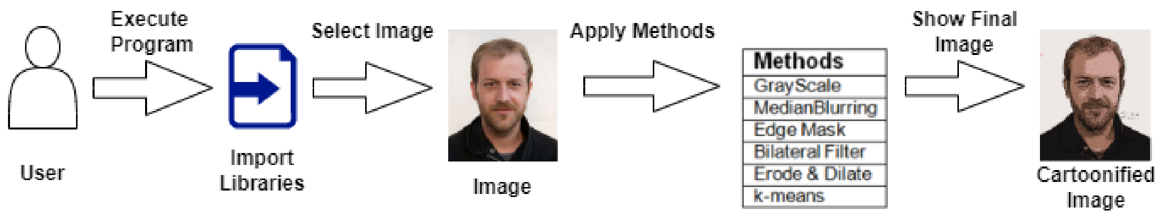
SYSTEM DESIGN

The process of defining the architecture, interfaces, and data for a system that satisfies specific requirements

4.1 Process Diagram



4.2 Architecture Diagram



Chapter 5

IMPLEMENTATION

This Phase explains the implementation of this project, software tools used and several issues. This illustrates the correctness of the project, which means whether it is satisfying the user requirements and does it holds the requirement specification as mentioned. It enables us to correct, if any error occurs.

Python is chosen for building this application is because it has a large number of libraries built for various purposes.

5.1 Technologies used

Python: python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). We use Python 3.6.4 as a programming language for building this application.

OpenCV: OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. It is used to perform different operations on images that transform them using different techniques.

Numpy: Mainly NumPy is used for dealing with arrays. Here the images that we use are stored in the form of arrays. So for that, we use NumPy.

easygui: easygui is a module used for GUI programming in python. In our application easygui is used to open the file box to upload images from the local system.

Imageio: Imageio is a python library that reads and writes the images.

Matplotlib: Matplotlib is used for visualization purposes. Here we plot the images using matplotlib.

OS: Here in our application os is used for dealing with paths like reading images from the path and saving the image to the path.

Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

5.2 Module Explanation

Step 1 — Installing OpenCV

pip install opencv-python

Step 2 — Importing necessary libraries

OpenCV
easygui
numpy
matplotlib.pyplot
imageio

Step 3 — Reading an image

Now for input, we select an image. Image can be selected from either your system or can be uploaded directly from the internet. For this, we use of easygui library.

The image is stored in img. In case no file is selected or the uploaded file format is not selected then a message as “Can not find any image. Choose appropriate file” will be displayed.

The image is by default read in BGR color palette. We need to convert it in RGB format.

Changing color palette
Next, we have to convert the image to grayscale

Converting the image to grayscale

Step 4 — Median Blurring

Here, we perform median blurring on the image. The central element of the image is replaced by the median of all the pixels in the kernel area. This operation processes the edges while removing the noise.

Step 5 — Edge Mask

The next step is to create the edge mask of the image. To do so, we make use of the adaptive threshold function.

Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions.

Step 6 — Removing Noise

We need to get rid of the unwanted details of the image or the "noise". In imaging, noise emerges as an artifact in the image that appears as a grainy structure covering the image.

To do so, we perform bilateral blurring.

Step 7 — Eroding and Dilating

Erosion —

- Erodes away the boundaries of the foreground object
- Used to diminish the features of an image.

Dilation —

- **Increases the object area**
- **Used to accentuate features**

Step 8 — Cartoonification of an image

Now, we perform stylization on the image using k-means clustering image.

Step 9 — Close button

It is used to close the application

5.3 Source Code

GUI

```
root=Tk()

root.geometry('600x550')

root.title('Cartoonify Your Image !')

root.configure(background='white')

root.iconbitmap('images\s.ico')

lbl=Label(root, text="Cartoonify Image Using OpenCV", fg='#364156', font=("Helvetica", 20))
```

```
lbl.place(x=90, y=30)
```

Reading the image

```
def upload():
    ImagePath = easygui.fileopenbox()
    cartoonify(ImagePath)

def cartoonify(ImagePath):
    global img1
    img1 = cv2.imread(ImagePath)
    if img1 is None:
        print("Can not find any image. Choose appropriate file")
        sys.exit()
    img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
    root.after(1000,root.destroy())
```

Displaying all the images in a sequence

```
    plt.imshow(img1)
    plt.axis("off")
    plt.title("ORIGINAL IMAGE")
    timer.start()
    plt.show()
```

Converting the color space from RGB to Grayscale

```
    global img1g
    img1g = cv2.cvtColor(img1, cv2.COLOR_RGB2GRAY)
    plt.imshow(img1g, cmap='gray')
    plt.axis("off")
    plt.title("GRAYSCALE")
    plt.show()
```

Median Blurring

```
    global img1b
    img1b = cv2.medianBlur(img1g, 3)
    plt.imshow(img1b, cmap='gray')
    plt.axis("off")
    plt.title(" MEDIAN BLURRING")
    plt.show()
```

Creating edge mask

```
    global edges
    edges = cv2.adaptiveThreshold(img1b, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
      dcv2.THRESH_BINARY, 3, 3)
    plt.imshow(edges, cmap='gray')
    plt.axis("off")
    plt.title("EDGE MASK")
    plt.show()
```

Removing noise

```
    global img1bb
    img1bb = cv2.bilateralFilter(img1b, 15, 75, 75)
    plt.imshow(img1bb, cmap='gray')
    plt.axis("off")
    plt.title("BILATERAL BLURRING")
    plt.show()
```

Eroding and Dilating

```
    global kernel,img1e,img1d
```

```
kernel = np.ones((1, 1), np.uint8)
img1e = cv2.erode(img1bb, kernel, iterations=3)
img1d = cv2.dilate(img1e, kernel, iterations=3)
plt.imshow(img1d, cmap='gray')
plt.axis("off")
plt.title(" ERODING AND DILATING")
plt.show()
```

COLOR QUANTIZATION

Color Quantization is implemented by using clustering algorithms.

1. Exclusive Clustering

Here we use exclusive clustering. K-MEANS technique is used.

```
global imgf
imgf = np.float32(img1).reshape(-1, 3)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
compactness, label, center = cv2.kmeans(imgf, 5, None, criteria, 10,
   cv2.KMEANS_RANDOM_CENTERS)
center = np.uint8(center)
final_img = center[label.flatten()]
final_img = final_img.reshape(img1.shape)

global final
final = cv2.bitwise_and(final_img, final_img, mask=edges)
plt.imshow(final, cmap='gray')
plt.axis("off")
plt.title("CARTOONIFIED IMAGE")
plt.show()
```

Chapter 6

TESTING

Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

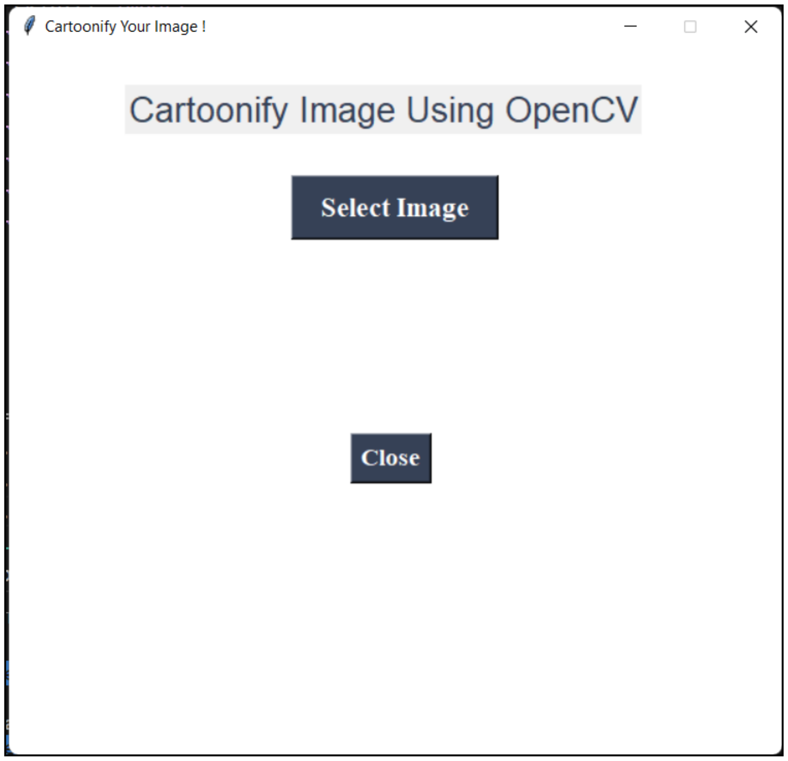
Test Steps:

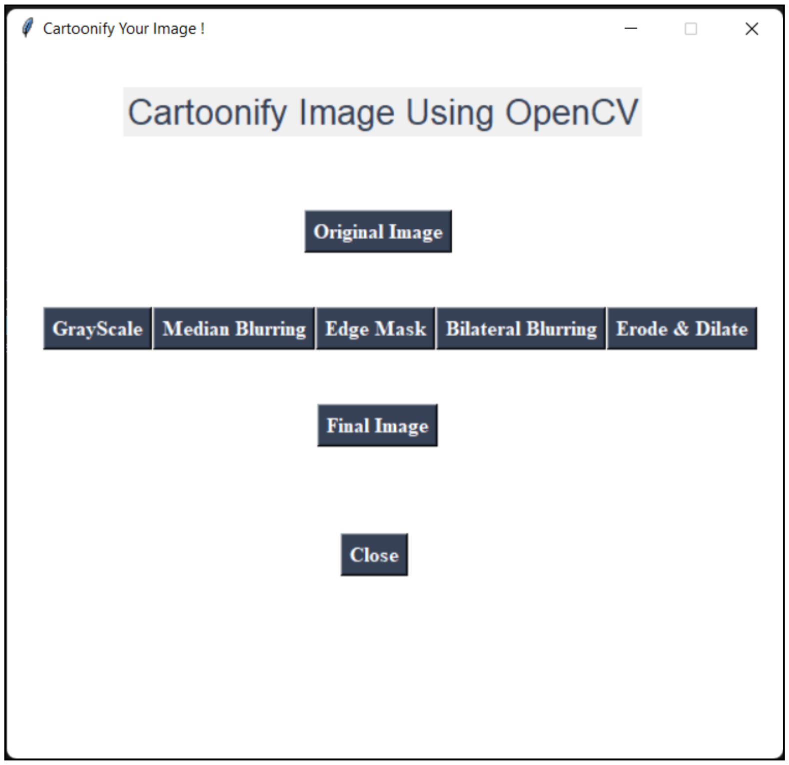
1. Load the original image.
2. Verify that the image has been loaded correctly.
3. Call function to cartoonify image.
4. Verify that cartoonified image has been created and saved correctly.
5. Load the cartoonified image.
6. Verify that the cartoonified image has been loaded correctly.
7. Compare pixel values of original and cartoonified images.
8. Verify that the pixel values of cartoonified image are different from the original image.
9. Verify that the cartoonified image has cartoon-like effect.

Expected results:

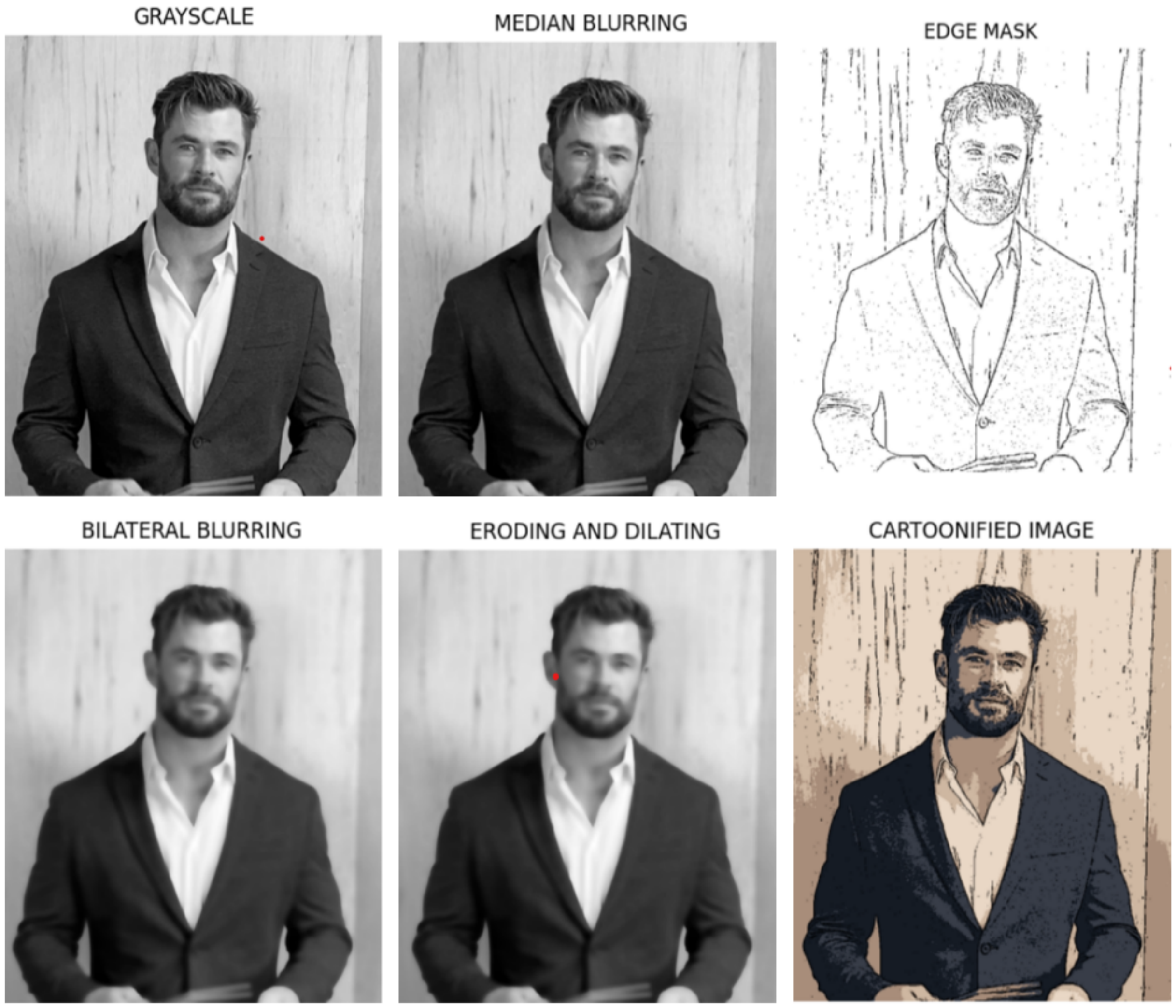
1. The original image should be loaded without any errors.
2. The loaded image should be the same as original image.
3. The cartoonification process should be executed without any errors.
4. The cartoonified image should be saved without errors.
5. The cartoonified image should be loaded without errors.
6. The loaded image should be same as cartoonified image.
7. The pixel values of the original and cartoonified images should be different.
8. The pixel values of cartoonified image should be significantly different from the original image.
9. The cartoonified image should have a cartoon-like effect.

User Interface



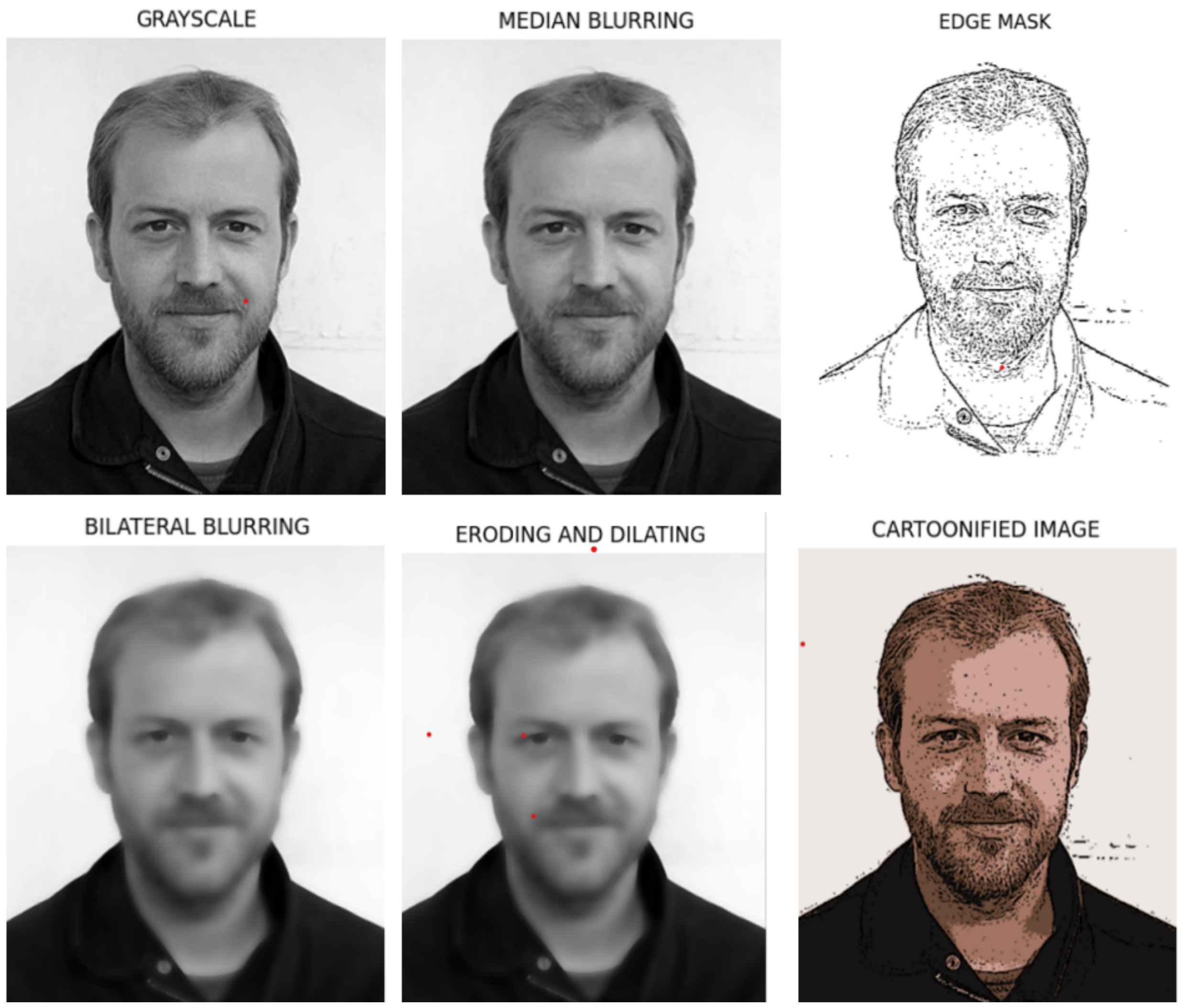


Input



Output

Input



Output

Chapter 7

CONCLUSION

The project “Cartoonify Image using OpenCV” is a python application, which is capable of selecting image from the computer and applying different cartoon features and getting the results. It contains all the 6 transitions of the image. And the final image is the cartoonified image. This technique can be used in variety of applications such as digital art and design, creating creative images. Overall, Cartoonifying image using OpenCV is a straightforward process that can yield impressive results with a little experimentation and practice. For future research many different filters can be used along with the existing methods.

Chapter 8

BIBLIOGRAPHY

1. "Cartooning an Image using OpenCV and Python" by A. J. Amsavalli and G. Ravindran.

2. "Real-time Cartoonifying of Video Streams using OpenCV and Python" by H. B. Dharmendra and R. C. Ranjitha.

3. "A Comparative Study of Cartoonification Techniques using OpenCV and Python" by N. R. Arunkumar and K. R. Rajan.

4. Cartoonize an Image using OpenCV –Aditi Rastogi