Image processing in MATLAB

Enhancement, Encryption & Decryption

A
Project
Submitted
In partial fulfillment
For the award of the Degree of
Bachelor of Technology
In Department of Electronics and Communication Engineering

Supervisor: -
Dr. Deepak Jhanwar

Submitted By:
1. Arpit Goyal (18EEAEC011)
2. Gayatri Chauhan (18EEAEC020)
3. Divya Meena (18EEAEC019)

Department of Electronics and Communication Engineering

Engineering College Ajmer

Bikaner Technical University

May 2022

Candidate's Declaration

I hereby declare that the work, which is being presented in the Project, entitled **"Image processing in MATLAB"** in partial fulfillment for the award of Degree of "*Bachelor* of Technology" in the department of Electronics and Communication Engineering, **and submitted to the Department of Electronics and Communication Engineering, Engineering College Ajmer**, Rajasthan Technical University is a record of my investigations carried under the guidance of **Dr. Deepak Jhanwar**, Department of Electronics and Communication Engineering, Engineering College Ajmer**.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

(Name and Signature of Candidates)

Arpit Goyal (18EC11)

Gayatri Chauhan (18EC20)

Divya Meena (18EC19)

Electronics and Communication Engineering,
Engineering College Ajmer,

Counter Signed by

Name of Supervisor:-
Dr. Deepak Jhanwar

Engineering College Ajmer

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the following student of VIII Semester, B. Tech. (Electronics & Communication Engineering) 2021-22 have completed the project titled "**Image processing in MATLAB"** in partial fulfillment for the award of the degree of Bachelor of Technology under Bikaner Technical University, Bikaner.

1. **Mr. Arpit Goyal** (18EEAEC011)
2. **Ms. Gayatri Chauhan** (18EEAEC020)
3. **Ms. Divya Meena** (18EEAEC019)

Date:

Guide name:-	**Project Coordinator name:-**	**HoD name:-**
Dr. Deepak Jhanwar	**Harish Kumar Sharma**	**Anurag Garg**
(Project Guide)	**(Project Coordinator)**	**(Head, ECE)**

ACKNOWLEDGMENT

The project report on "**Image processing in MATLAB**" is the outcome of guidance, moral support, and devotion bestowed on us throughout our work. For this, we acknowledge and express our profound sense of gratitude and thanks to everybody who has been a source of inspiration during the project preparation.

We offer our sincere phrases of thanks to our Project Guide, **Dr. Deepak Jhanwar** without whose support and guidance it would not have been possible for this Project to have materialized and taken a concrete shape.

I would also like to express our gratitude to our Project Coordinator **Harish Sharma** to provide us with a better facility to enhance our practical knowledge.

We are thankful to **Dr. Rekha Mehra** (HOD, ECE), Govt. Engineering College, Ajmer.

Last but not the least, I am thankful to all our colleagues and other staff members, who helped us directly or indirectly in the Project work.

1) **Mr. Arpit Goyal** (18EEAEC011)
2) **Ms. Gayatri Chauhan** (18EEAEC020)
3) **Ms. Divya Meena** (18EEAEC019)

TABLE OF CONTENTS

TABLE OF CONTENTS ……..... v

LIST OF FIGURES …….. vii

ABSTRACT ……..……..... viii

Chapter 1 : Introduction

1.1 Introduction 9

1.2 Objective 9

Chapter 2 : Literature Survey

2.1 MATLAB 10

2.1.1 Basics..... 10

2.1.2 GUIDE..... 10

2.1.3 Debugging.....11

2.2 Image processing 11

2.2.1 About.....12

2.2.2 Tools and libraries..... 12

2.3 Cryptography 13

2.3.1 History..... 13

2.3.2 Algorithms..... 14

2.3.3 Steganography..... 14

2.3.4 Future 15

2.4 Fourier Series..... 15

2.4.1 Intuition.....16

2.4.2 Epicycles.....18

Chapter 3 : Designing Theory

3.1 Enhancement 19

3.1.1 Modifications..... 19

3.1.2 Noise..... 19

3.1.3 Filters..... 19

3.1.4 Operations..... 20

3.1.5 Adjustments..... 20

3.1.6 Menu..... 20

3.1.7 History control..... 20

3.2 Cryptography 21

3.2.1 Encryption..... 21

3.2.1.1 Key generation..... 21

3.2.1.2 Image processing..... 21

3.2.2 Decryption..... 22

3.3 Steganography 22

3.4 Epicycles 23

Chapter 4 : Project Simulation

4.1 Main..... 24

4.2 Enhancement..... 24

4.3 Steganography..... 25

4.4 Encryption..... 26

4.5 Epicycles(One-way)..... 26

Chapter 5: Challenges and Future Improvement

5.1 Challenges Faced 27

5.2 Future Improvement 27

Conclusions **29**

Literature References **30**

Appendix **31**

LIST OF FIGURES

Chapter 2 : Literature Review

2.4.1. Properties of Fourier Transform..... 16

2.4.1.1. Demonstration of rotating vectors using exponential..... 16

2.4.1.2. Rotation of circles at different frequencies..... 17

2.4.1.3. function as a sum of multiple waves..... 17

3.2.1.1.1 Key Generation..... 21

3.2.1.2 Encryption and Decryption in steganography..... 22

4.1. Main menu of the platform..... 24

4.2.1. Enhancement platform demonstrating noise features..... 24

4.2.2. Enhancement platform demonstrating noise features..... 25

4.3.1. Steganography User Interface..... 25

4.4.1. Encryption and Decryption of an Image..... 26

4.5.1 Drawing an image using Epicycles..... 26

ABSTRACT

Photography was invented in 1826 by French inventor Joseph Nicéphore Niépce. Since then, it has revolutionized the modern world. From medical evidence in court & drawing images of industrial machines to personal photos for memory, not a single field is left untouched by photography. Nowadays when there are so many photo editing software, not a single one ensures true privacy and provides open-source code to ensure that no data is being collected. In six months of project making, we have developed an Image processing app that not only provides enhancement tools but also displays the power of steganography, a user-friendly environment to encrypt your enhanced images before sending them forward, and a very beautiful display of Fourier series epicycles which creates an image using Fourier series, an intuitive way to learn and understand the beauty of mathematics. Mathematics could be called the language of science. Our tool proves why it is said so. And what could be a better platform to make all this other than Matlab which not only provides numerous libraries but also is a powerful debugging tool that can display all the things happening inside the code. This app is built using the Guide function in Matlab and has the potential to influence this generation toward data security as well as mathematics.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Enhancing images plays a crucial role in a variety of areas, from research to day-to-day uses. Images whether taken from a microscopic level or of the mighty space could never be analyzed and studied until filtered and enhanced. Enhancing images also applies when detecting defects in products that cannot be identified by the human eyes, such as PCBs. Any circuit faults can be easily identified by computers using image processing and enhancement. All sorts of entertainment and marketing media use image processing. Keeping in mind all the applications, we have built a MATLAB-based GUI platform that supports every basic image processing and filtering function, with additional features of data hiding using steganography, image encryption, and a beautiful demonstration of how any image can be drawn using Fourier series and epicycles. Apart from all the features, we have built a GUI platform making the interfacing easy and fun for the user.

1.2 Objective

The objective of this project is to build a platform that promises every basic function used in image filtering and processing using MATLAB. Not only enhancing the image but hiding crucial and selected information or hiding the whole image is another feature that this platform provides. One of the highlights of this platform is a demonstration of images using the Fourier series. Our goal is to show a new and magnificent way of showing the power of mathematics and the Fourier series.

CHAPTER 2

LITERATURE SURVEY

2.1 MATLAB

MATLAB or Matrix Laboratory is a high-level language for matrix calculation, numeric analysis, and scientific computing. In MATLAB we can type on the command line or use a program file (.m file). The MATLAB commanding language is quite similar to C. It supports control flow, if, for, while, switch, etc but also has many differences like it doesn't need variable declaration or pointers like C requires.

MATLAB has many advantages such as it requires shorter code and has faster compilation. MATLAB supports multiple in-built libraries and toolboxes that help in easy coding. About our project, we found MATLAB extremely helpful since MATLAB can import and export several image formats like BMP, GIF, HDF, JPEG, PNG, and many more.

2.1.1 BASICS

MATLAB stores images as matrices. The image pixels are referenced using (row, col) values with the origin of the coordinate system (1, 1) in the top left corner of the image. By default, MATLAB reads an image in uint8 (unsigned 8-bit integer) format. The values that each pixel holds are in the range[0, 255].

For some functions, we might need to convert the image format into the double format, where the double format has pixel values in the range [0,1]. A simple function 'im2double()' is used to convert any image into the double format.

2.1.2 GUIDE

GUIDE is a UI design environment that provides a set of tools for creating user interfaces and simplifying the process of laying out and programming UIs. GUIDE creates a new platform where we could drag and drop different tools we require and align them as we need. We can change the colors of the tools, input icons instead of names in buttons, and also input a background image.

Every tool that we design using GUIDE can be programmed according to us as to what we wish for the particular item to work as. This is done when we use the callback feature and write our code. In GUDEs code editor we are given with code debugger, code folding, and code analyzer. Apart from this GUIDE supports adding templates, component libraries, and component browsers.

A very interesting feature that GUIDE provides is plotting 2D and 3D in axes (axes, polar axes, Geo axes). We can even pan, zoom or rotate the axes and figure interactions using keyboard and mouse interactions.

2.1.3 DEBUGGING

Debugging is a very intuitive way of resolving errors. It is a multistep cess that helps in isolating the source of the problem and in correcting the problem by determining a way to work around it.

In MATLAB we can diagnose problems in the MATLAB code file by debugging the code interactively in the live editor and editor or programmatically by using the debugging function in the command window.

The different ways with which we can debug our code are:

- Display output by removing semicolons.
- Run the code to a specific line and pause by clicking the Run to Here button.
- Step into functions and scripts while paused by clicking the Step In button.
- Add breakpoints to your file to enable pausing at specific lines when you run your code.

2.2 IMAGE PROCESSING

It deals with all the filter variations and editing one can apply to an image. From removing noise in an image to adding noise in an image. Filters such as gaussian filters, mean/median filters to noise like Gaussian noise, salt and pepper noise, and periodic noise.Images can be controlled into showing different colors as per the user's need. One can convert a colored RGB image into a grayscale image or binary image. We can even apply red, green, blue, and many more color filters to an image.

From rotation to cropping an image and sharpen to flipping an image, all fall under image processing. Many more operations such as Fourier transform of an image, histogram equalization, histogram

displaying, edge detection, etc are a few of many functions that image processing possesses. Image processing has a great application in our day-to-day lives. Everything that we see in marketing/advertisements to entertainment and posters uses image processing to enhance the way their product looks. Even on social media, the media itself has built its image-enhancing interface for users to modify their image before using it publicly.

2.2.1 ABOUT

In this project we have taken image processing to a next level using all the features like filters and noise reduction and using the final product to be able to use for different purposes listed on the home page like image encryption.

We have made histogram equalization, Fourier transform, histogram display, Sobel edge detection, canny edge detection, log transformation, and power transformation inside the advanced operations category. In filters, we made Gaussian, mean, and median filters, whereas in adding noise we made Gaussian noise, periodic noise, and salt and pepper noise.

our image enhancement platform also supports undo and redo functions where we can easily change back any filter or operation we don't like and repeat the operation back if one changes their mind.

The image processing platform also has a feature of controlling the brightness, contrast, and vignette of an image using a slider.

2.2.2 TOOLS AND LIBRARIES

When we log into ThingSpeak, we can use functions from the below toolboxes

- Statistics and Machine learning toolbox.
- Curve fitting toolbox.
- Signal processing toolbox.
- Mapping toolbox.
- System identification toolbox.
- Deep learning toolbox.
- DSP system toolbox.
- Datafeed toolbox.

- FInancial toolbox.
- Image processing toolbox.
- Text analysis toolbox.
- Predictive maintenance toolbox.

We can package MATLAB files to create a toolbox to share with others. These files can include MATLAB code, data, apps, examples, and documentation.

2.3 CRYPTOGRAPHY

Cryptography is a way of communication that is secure. The prefix "crypt" means "hidden" and the suffix "graphy" means "writing". In cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side.

2.3.1 HISTORY

The first known evidence of the use of cryptography was found in an inscription carved around 1900 BC, on the tomb of the nobleman Khnumhotep II, in Egypt.

Evidence of some use of cryptography has been seen in early civilizations. "Arthashastra", a classic work on statecraft written by Kautilya, describes the espionage service in India and mentions giving assignments to spies in "secret writing".

Fast-forwarding to around 100 BC, Julius Caesar was known to use a form of encryption to convey secret messages to his army generals posted on the war front. This substitution cipher, known as Caesar cipher, is perhaps the most mentioned historic cipher in academic literature. (A cipher is an algorithm used for encryption or decryption.) In a substitution cipher, each character of the plain text (plain text is the message which has to be encrypted) is substituted by another character to form the ciphertext (ciphertext is the encrypted message).

On June 27, 1940, the Germans set up two-way radio communication in their newly occupied French territory, employing their most sophisticated encoding/cipher machine, Enigma, to transmit confidential information during the world war. Enigma was so powerful that by the time any person could crack the Enigma code, the Germans changed the code entirely leaving all the efforts made to crack it, to go in vain. Later an English mathematician Alan Turing built a machine that would calculate faster than humanly possible and cracked the Enigma, hence reducing the war for two years shorter and saving millions of lives. The machine is now what we know as a computer.

There is multiple evidence from the past where encoding information was found useful and at times like war, saved many lives as well. Even today, to protect user's privacy and security many social media claim to offer end-to-end encryption and data privacy.

2.3.2 ALGORITHMS

The algorithm that we have used in encryption is the Advanced Encryption Standard or AES. AES by now is the most symmetric algorithm in the world. AES can encrypt all 128 bits of the data path in one round. Each round consists of four layers, byte substitution, shift row, mix column, and key addition.

In AES we have used finite fields or Galois fields. All internal operations of AES are based on finite fields. There are three basic algebraic structures. A group is a set in which you can perform one operation (usually addition or multiplication mod) with some nice properties.

A Field is a set of elements with the following properties:

- All elements of F form an additive group with the group operator"+" and the neutral element 0.
- All elements of F except a form a multiplicative group with the group operator "X" and the neutral element 1
- When the group operations are mixed the distributivity low holds for all a, b, and c belonging to F: a(b+c) = (ab)+(ac)

A ring is a set equipped with two operators called addition and multiplication. A ring is a group under addition and satisfies some of the properties of a group multiplication. For computation in finite sets for use in cryptography, we use modular arithmetic.

2.3.3 STEGANOGRAPHY

Steganography is the hiding of data in an image or other digital artifact. We can hide any digital information in an image. We can for example hide a text or HTML file in an image or an audio file or any file of 0s and 1s. Hiding one image inside another can be achieved by steganography. We can hide selected data instead of a whole image. Steganography is a very important feature in data hiding and ensures privacy. In our platform, we have shown how steganography can hide a particular data of an image and also recover the data back.

2.3.4 FUTURE

The future of cryptography is very bright. It will completely change the whole data privacy system. Enabling cryptography will help users from getting their data leaked while surfing on the internet or during any computational process. Nowadays, trespassers have emerged with several ways in which they could invade our privacy and manipulate data for their purposes. The technology has developed to a very broad extent that one could record the sound of keys pressed on a keyboard while typing passwords or any confidential content and since each key would generate a different sound due to its unique location, they could track what we would have typed. Realizing and understanding all these threats, cryptography and encryption is a promise to the user of protecting their privacy over the internet and building a safe future.

2.4 FOURIER SERIES

A Fourier series is an expansion of a periodic function in terms of an infinite sum of sines and cosines. The Fourier series makes use of the orthogonality relationships of the sine and cosine functions. The computation and study of the Fourier series are known as harmonic analysis and are extremely useful as a way to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical.

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$

where,

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,dx$$

Fourier coefficients are obtained by :

$$c_m = \frac{1}{T}\int_{T} x(t)e^{-jm\omega_o t}dt$$

Property	Function	Fourier Transform
	$f(t)$	$\hat{f}(\omega)$
Inverse	$\hat{f}(t)$	$2\pi\, f(-\omega)$
Convolution	$f_1 * f_2(t)$	$\hat{f}_1(\omega)\, \hat{f}_2(\omega)$
Multiplication	$f_1(t)\, f_2(t)$	$\frac{1}{2\pi}\, \hat{f}_1 * \hat{f}_2(\omega)$
Translation	$f(t - t_0)$	$e^{-it_0\omega}\, \hat{f}(\omega)$
Modulation	$e^{i\omega_0 t}\, f(t)$	$\hat{f}(\omega - \omega_0)$
Scaling	$f(\frac{t}{s})$	$\lvert s \rvert\, \hat{f}(s\,\omega)$
Time derivatives	$f^{(p)}(t)$	$(i\omega)^p\, \hat{f}(\omega)$
Frequency derivatives	$(-it)^p\, f(t)$	$\hat{f}^{(p)}(\omega)$
Complex conjugate	$f^*(t)$	$\hat{f}^*(-\omega)$
Hermitian symmetry	$f(t) \in \mathbb{R}$	$\hat{f}(-\omega) = \hat{f}^*(\omega)$

fig 2.4.1. Properties of Fourier Transform.

2.4.1 INTUITION

When we dive into exponential fourier series, e^{it} forms a rotating vector which has an amplitude of C_n.

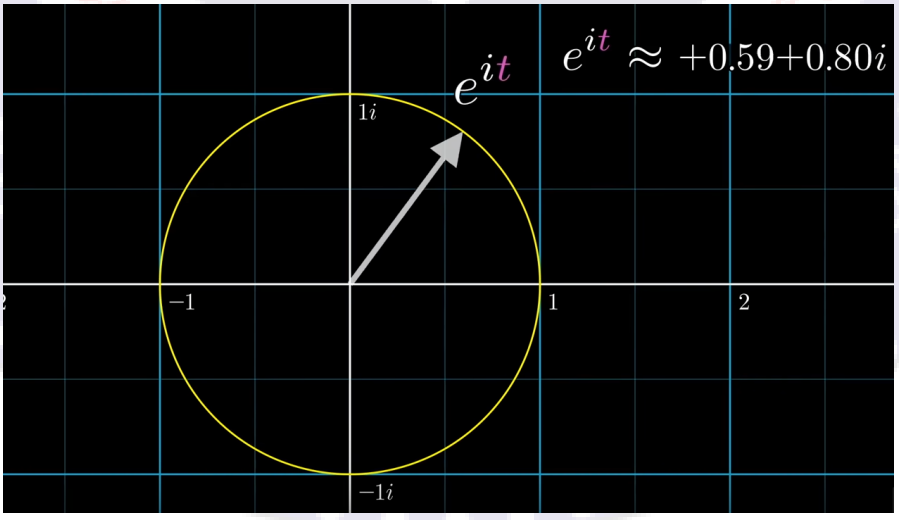

fig 2.4.1.1. Demonstration of rotating vectors using exponential.

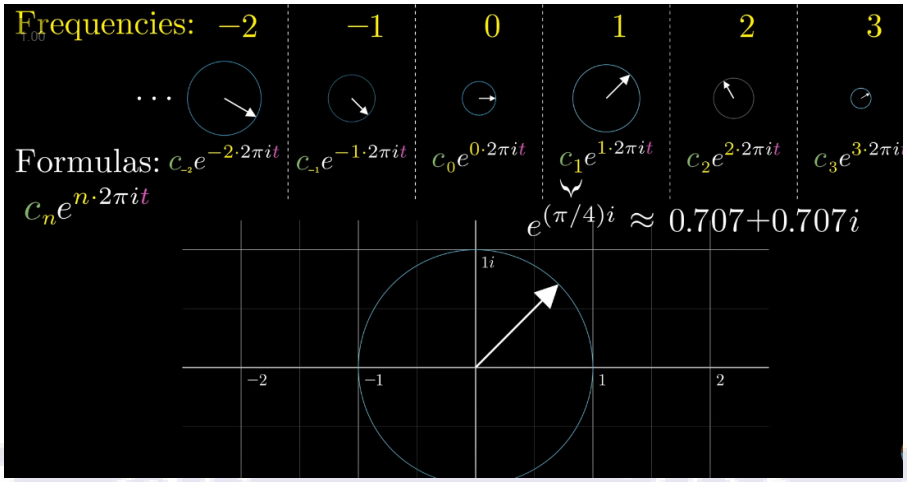

fig 2.4.1.2. Rotation of circles at different frequencies.

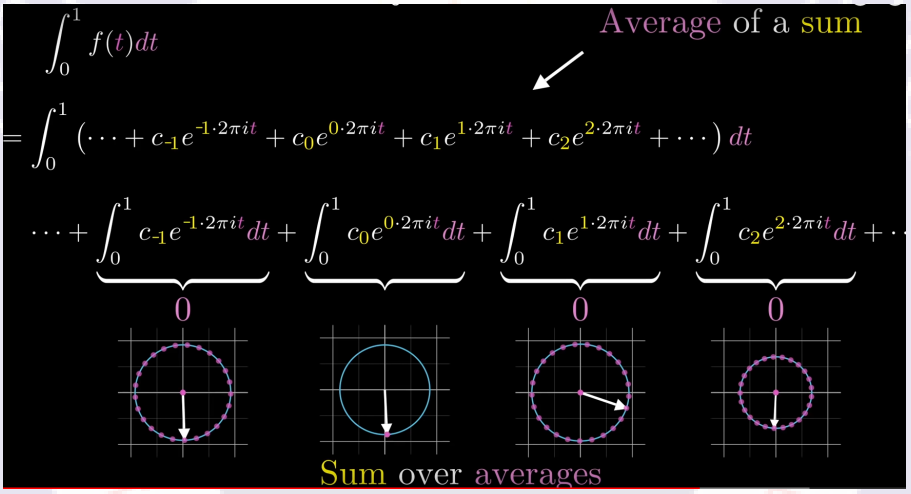

fig 2.4.1.3. function as a sum of multiple waves.

Combination of all rotating vectors with variable amplitude forms the signal consisting of sine and cosine waves. To calculate amplitude of any vector, we need to stop that rotating vector while making others rotate as then they will nullify to be zero and we could get the coefficient. Hence we multiply them with e^{-nit} to get the average.

2.4.2 EPICYCLES

Fourier series can be explained as expressing a repetitive curve as the sum of sine curves. Since the "summation of sine waves" interpretation shows how many waves are there at each frequency, it is widely used in engineering, physics, and mathematics. The main idea in this interpretation is that sine and cosine functions are mutually orthogonal, like vectors that are perpendicular to each other. One can think of a sine wave as the distance covered by the shadow of a ball that turns around a circle. These circles that represent different sine waves of different frequencies are called epicycles. Using these epicycles we can demonstrate how any wave of any shape is made up of tiny sine waves. Now since every image is also composed of tiny sine waves, we can also draw any image using epicycles.

CHAPTER 3

DESIGNING THEORY

3.1 ENHANCEMENT

The enhancement feature of the platform provides us with several features for analyzing and editing an image. We have provided two separate axes where the user can see the original image and the modified image simultaneously. There are multiple filters and noise-reducing features with control of brightness, contrast, and vignette modifications. Apart from all the exciting features, if one decides to undo any edits, we have provided an Undo and Redo buttons for users to skip anytime they want. And to start fresh, there is also a Reset button.

3.1.1 MODIFICATIONS

Modification provides features of several color conversions, such as red, blue, green, cyan, pink and yellow. We have also provided features of RGB to grayscale conversions, RGB to black and white conversions and image sharpening.

3.1.2 NOISE

Inside the Enhancement platform, we have added three noise-adding features, Periodic noise, Salt & Pepper noise, and Gaussian noise. Periodic noise adds ripple-like noise inside images after converting the image into a grayscale image. Salt & Pepper noise applies multiple dots of different colors on the image, making an effect of spilled salt and pepper. The Gaussian noise adds a noise that makes the image look a little blur, and faded and makes it rough.

3.1.3 FILTERS

We have added multiple noise removing filters that can enhance images and remove noises such as Gaussian noise, Salt and Pepper noise, periodic noise. We have added the Gaussian filter, mean filter, median filter.

3.1.4 OPERATIONS

Inside the enhancement section, we have subdivided features under operations. We have included histogram equalization, histogram transformation. In addition, we have added the fourier transform feature followed by log and power transformation. In edge detection, we have added two edge detections, Sobel and Canny edge detection. The operation section also provides a rotation feature and flip feature.

3.1.5 ADJUSTMENTS

Another section inside the enhancement platform is adjustments. Under the adjustments we have included control of brightness, contrast and vignette of the image. This section enables more manipulation of image data and can be very helpful in studying an image.

3.1.6 MENU

It is the main interface of the platform that gives direct access to all the features we provide with an about button that contains the description of the platform with the names of the group members and the project guide. This interface has an exit button that will close the platform with a greeting note.

The UI in the main menu contains a background image and descriptive tags with images as buttons giving an idea of what the feature is about.

3.1.7 HISTORY CONTROL

This is an additional function in the image enhancement platform that provides us with an undo and redo feature, enabling us to skip back and reverse edits whenever we feel like it.
For each modification, image data is being stored by the name of that filter and a unique variable is appended into a list to be extracted when undo and redo is called. As the operations increase, a counter also increases which works as an index for the list and with the value at the index, we recall that particular image data to be edited again.

3.2 CRYPTOGRAPHY

3.2.1 ENCRYPTION

This feature enables us to browse any image from our system and encrypt it using a password. This password is set by the user and the image can only be decrypted when the password has been fed correctly.

If the password is wrong, it will show an error and the image will not be decrypted.

As it takes the used defined 6 digit password, it calls the key generation function.

3.2.1.1 KEY GENERATION

This function takes the image as well as password as input. Then by using the dimensions of the images, it makes a key using that password. Key will be having same dimensions as the image. After that this binary key is converted to decimal form using simple conversion formula.

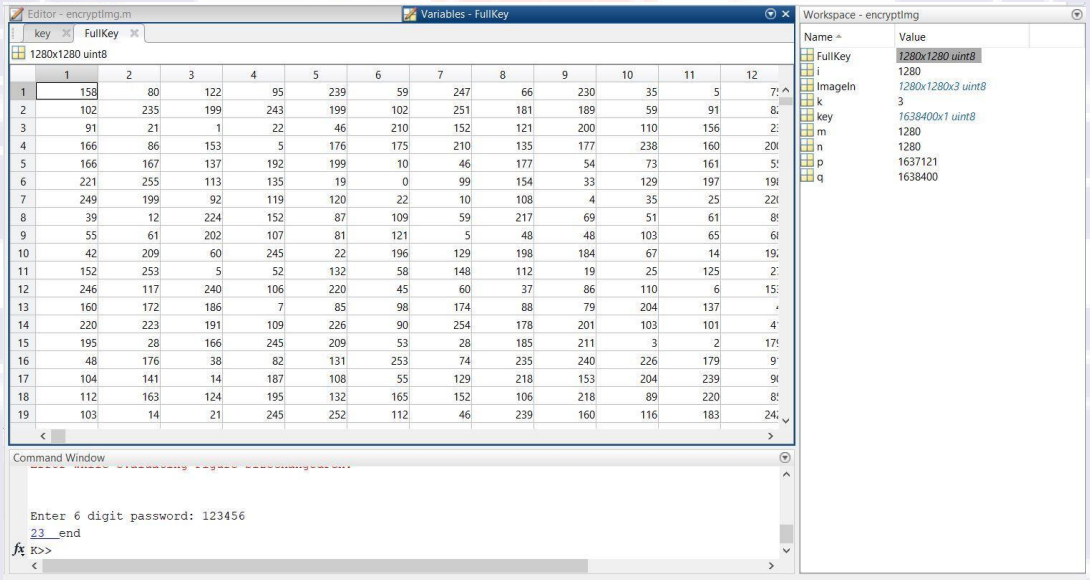

fig 3.2.1.1.1 Key Generation

3.2.1.2 IMAGE PROCESSING

Once key is generated, image process function (encryptImg) is called with key passed to it. This function uses bitwise XOR to process image. When key and image are passed through XOR gate, we get the encrypted image.

XOR gate have one beautiful characteristic i.e. when same key is XOR with the encrypted image, we get teh decrypted imge.

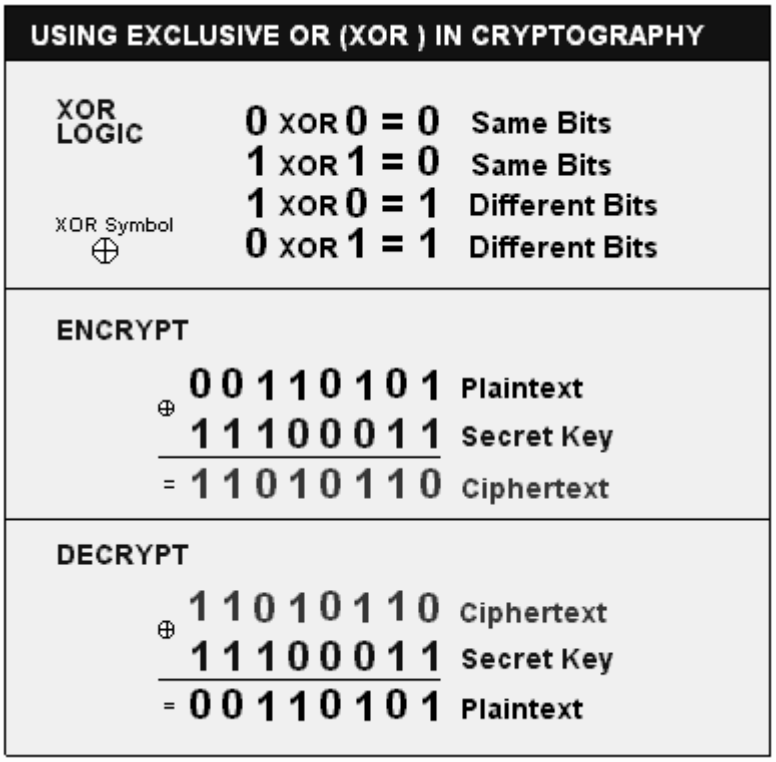

fig 3.2.1.2 Encryption and Decryption in steganography.

3.2.2 DECRYPTION

As per above fig., decryption doesn't require another or different piece of code. With the help of XOR gate, we can encrypt and decrypt using same key which can only by generated by that particular 6 digit password.

3.3 STEGANOGRAPHY

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination.

In this project, we are changing the last bit or least significant bit of the image as per our message in orer to send the hidden data. We need to be displayed, we are using logical function. Steganography is different from cryptography as here we only hide information by change least significant bit but in encryption we are processing the whole image.

3.4 EPICYCLES

When the figure is drawn by the user, we capture the x and y coordinates of that figure and converts it into complex number. Then we decide the no. of circles which is equal to the no. of samples taken. After that we calculate the fourier transform as well the frequency of rotation, amplitude or radius and its phase. Additional sorting feature is also provided that sorts the epicycles as per its radius. After that using these values we plot the circles for that particular instance. With increase in time we move to forward coordinate. This way we can trace the head of vectors to form the desired figure.

Chapter 4

Project Simulation

4.1 Main

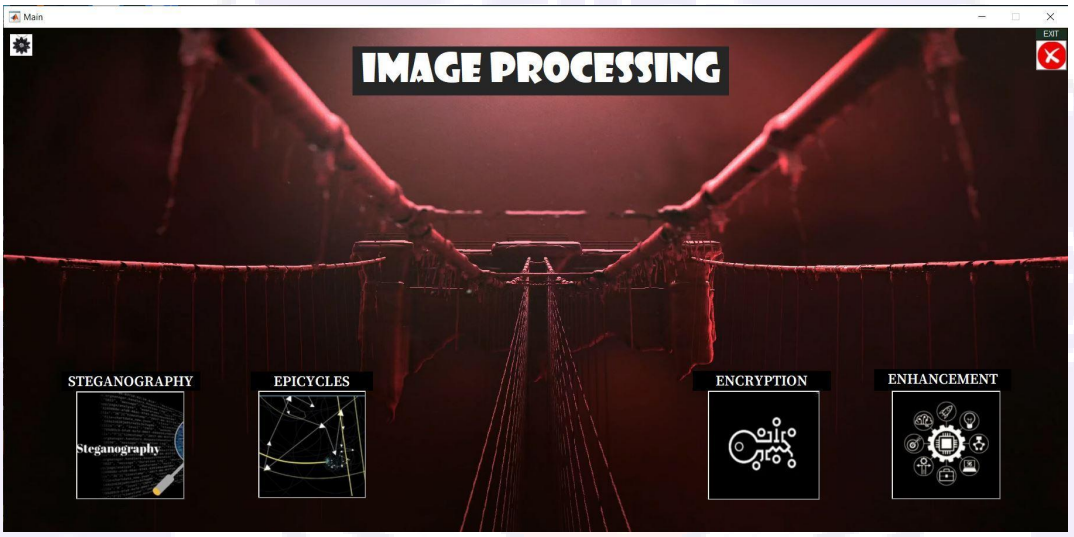

fig 4.1. Main menu of the platform.

4.2 Enhancement

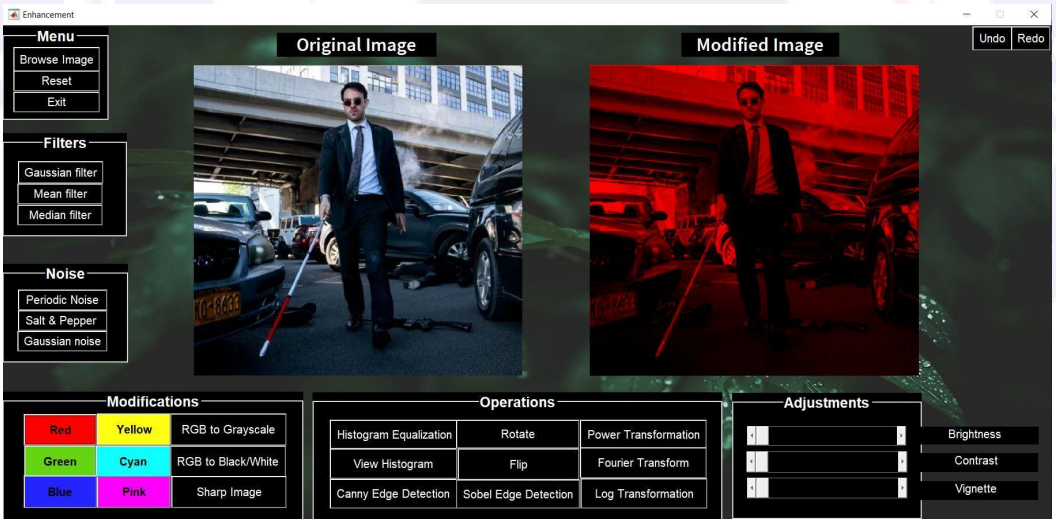

fig 4.2.1. Enhancement platform demonstrating noise features.

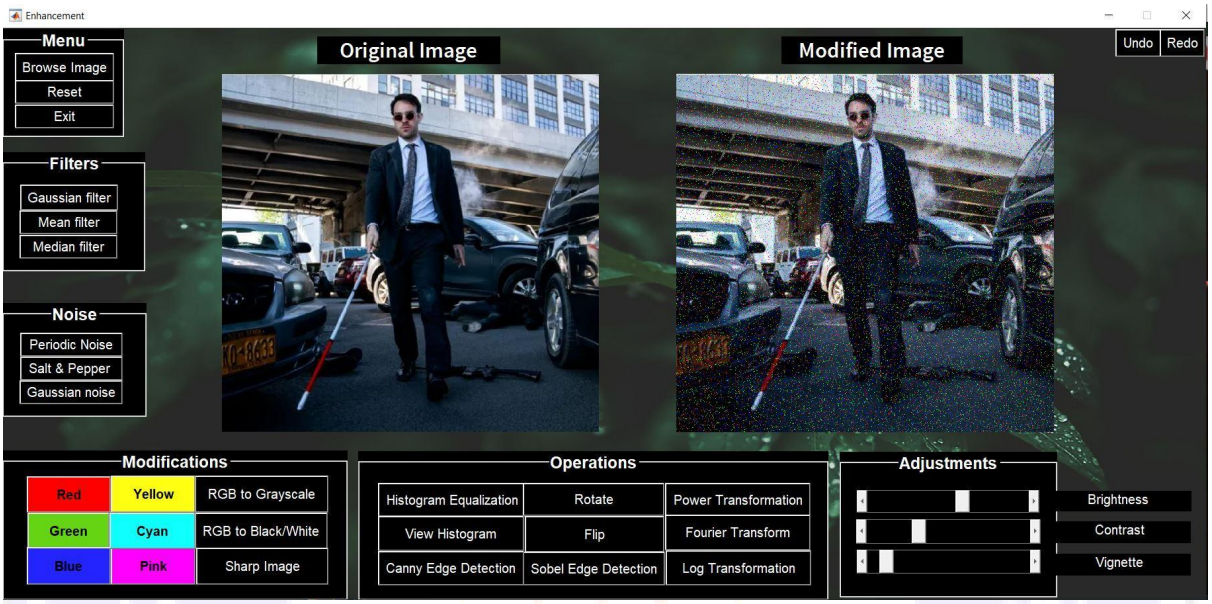

fig 4.2.2. Enhancement platform demonstrating noise features.

4.3 Steganography

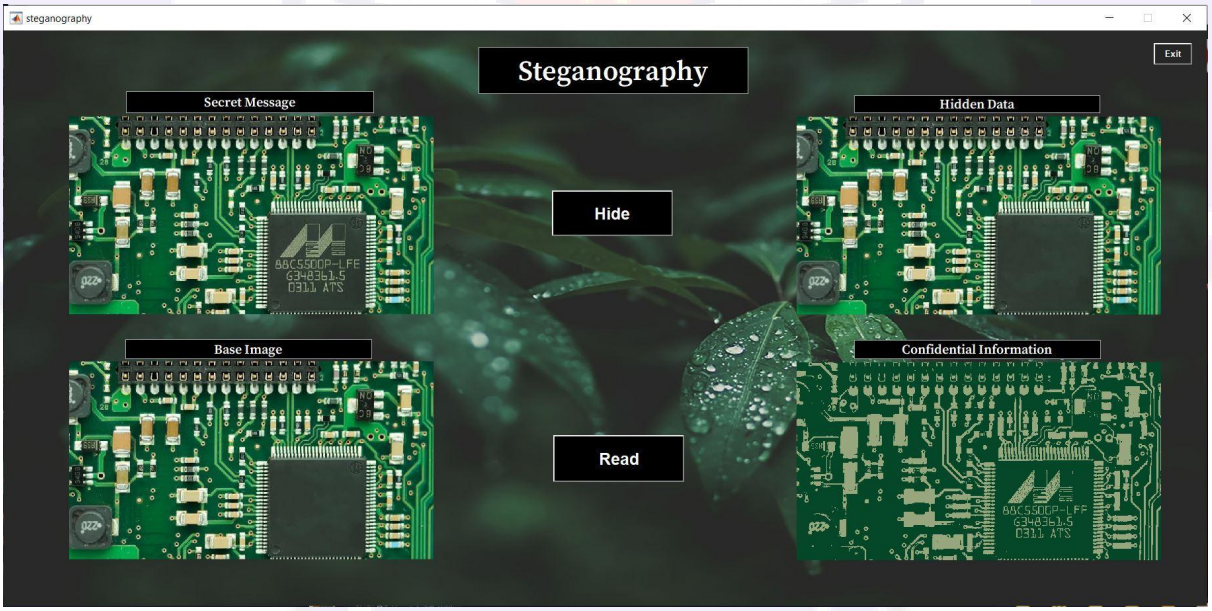

fig 4.3.1. Steganography User Interface.

4.4 Encryption

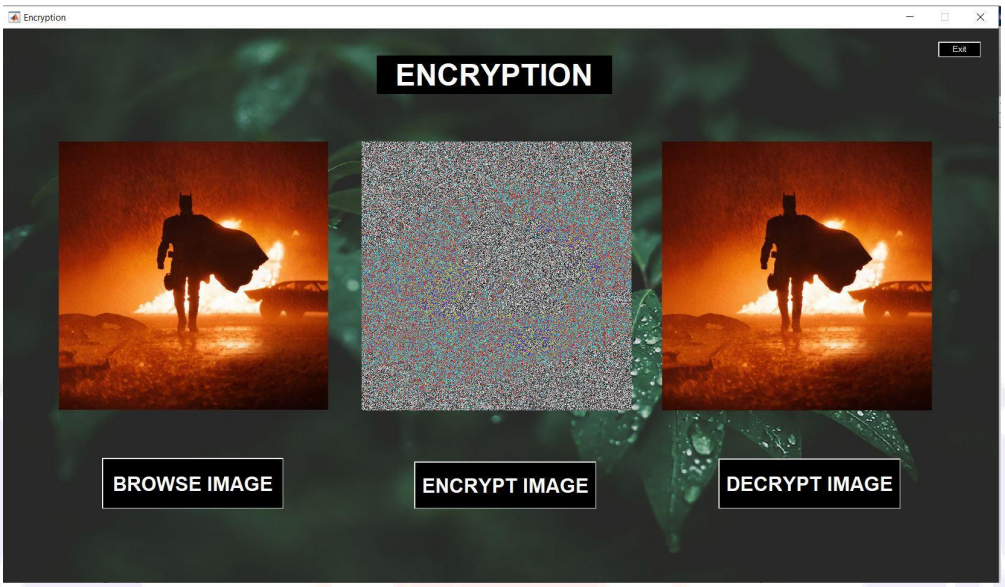

fig.4.4.1. Encryption and Decryption of an Image

4.5 Epicycles

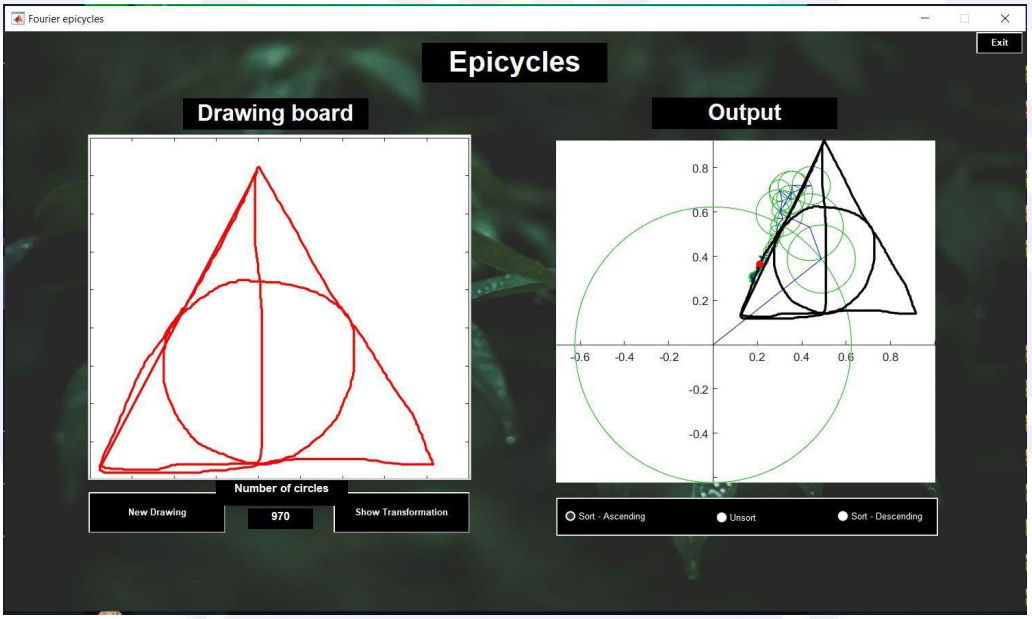

fig 4.5.1 Drawing an image using Epicycles

Chapter 5

CHALLENGES AND FUTURE IMPROVEMENTS

5.1 Challenges Faced

Creating a ready-to-go platform with our vision was very challenging and at the same time educational for us. We wanted to ensure our platform was built with a completely new idea and was kept original. Building different features like encryption, steganography, image enhancement, and epicycles and making a single platform to access them all easily was our first challenge when we learned about the GUIDE platform and made a completely interactive and easy to surf user interface platform. We also overcame challenges faced in building the algorithm for encryption and steganography with research and hard work. Simulating the Fourier epicycles was a tough task. Our vision was to create an informative yet beautiful representation of how the Fourier series plays an important role in everyday lives and ways in which we could introduce future students to the Fourier series. After multiple sessions and research, we were finally able to do so with added features. We faced and resolved all challenges that were in our way of building this platform and the only one left is to showcase the platform's potential and importance.

5.2 Future Improvements

We have a complete running platform with multiple features introducing us to different aspects and perspectives of image enhancement ensuring error-free performance. Yet many features and concepts can be added to the platform exploring more about images and further, videos. Our platform holds the potential to explore other fields than just images. We plan to explore the video and audio processing aspects, where we can use our current features on video files. We can explore how audio is processed and the ways with which any audio or video can be enhanced.

To make it available for users everywhere, we can create a setup file that can be executable wherever and whenever needed without the requirement of MATLAB being installed.

Providing all these features, and understanding how storage is important for everyone, we can build a compression algorithm that would reduce the size of the file with minimum data loss. Keeping in mind object and motion detection there are limitless ways to make the platform versatile and better.

CHAPTER 6
CONCLUSION

6.1 Conclusion

This project introduces various ways in which MATLAB can be found useful in understanding and analyzing different aspects of an image. In addition to making all the features easy to access and explore by the users, we have built a complete interactive GUI that not only enhances the look of the platform but also makes it easy to understand the motive behind every feature. The platform introduced a different perspective of understanding the Fourier series and epicycles and demonstrated that every image consists of combinations of multiple sine functions. With steganography and encryption highlighting many features of the platform, we have successfully built a complete up and running platform ready to explore.

CHAPTER 7

LITERATURE REFERENCES

"To refer a research paper"-

1. Bilgecan Dede, "Purrier Series (Meow) and Making Images Speak".
2. Grant Sanderson, Vivek Verma, "But what is a Fourier series? From heat flow to circle drawings ", 13 June 2019.
3. Victor Martinez-Cagigal, "GUI that computes the required epicycles to match a custom drawing by using DFTs".
4.

"To refer to videos and articles" -

1. https://www.youtube.com/watch?v=mEN7DTdHbAU
2. https://www.youtube.com/watch?v=2hfoX51f6sg
3. https://www.youtube.com/watch?v=ds0cmAV-Yek
4. https://in.mathworks.com/help/images/image-enhancement-and-restoration.html?s_tid=srchtitle_image%20enhancement_5
5. https://www.youtube.com/watch?v=nAVgUfk-ALE
6. https://dokumen.tips/engineering/digital-image-processing-image-enhancement-55b0f8078ee4a.html?page=3
7. https://matlabacademy.mathworks.com/R2021b/portal.html?course=imageprocessing#chapter=2&lesson=2&section=1

Appendix

1. Main

```
function varargout = Main(varargin)
% MAIN MATLAB code for Main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Main_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Main_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Main

% Last Modified by GUIDE v2.5 01-May-2022 18:52:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line output for Main
handles.output = hObject;

ah = axes('unit','normalized', 'position', [0 0 1 1]);
bg = imread('daredevil.jpg'); imagesc(bg);
set(ah, 'handlevisibility', 'off', 'visible', 'off');
uistack(ah, 'bottom');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Main wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in advanceOPBtn.
function advanceOPBtn_Callback(hObject, eventdata, handles)
% hObject    handle to advanceOPBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
EnhanceImg();


% --- Executes on button press in encryptionBtn.
function encryptionBtn_Callback(hObject, eventdata, handles)
% hObject    handle to encryptionBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Encryption();

% --- Executes on button press in steganographyBtn.
function steganographyBtn_Callback(hObject, eventdata, handles)
% hObject    handle to steganographyBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
steganography();

% --- Executes on button press in exitBtn.
function exitBtn_Callback(hObject, eventdata, handles)
% hObject    handle to exitBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
msgbox('Thanks for using this platform')
pause(1)
close();
close();

% --- Executes on button press in epicyclesBtn.
function epicyclesBtn_Callback(hObject, eventdata, handles)
% hObject    handle to epicyclesBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
epicycles();
```

```
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

myicon = imread("About2.jpg");
h = msgbox(["Guide:";"Dr. Deepak Jhanwar";"";"Made by:";"1) Gayatri Chauhan";"2)
Arpit Goyal";"3) Divya Meena"],"About","custom",myicon);
pause(10);
```

2. Enhancement

```
function varargout = EnhanceImg(varargin)
%ENHANCEIMG MATLAB code file for EnhanceImg.fig
%      ENHANCEIMG, by itself, creates a new ENHANCEIMG or raises the existing
%      singleton*.
%%
%      ENHANCEIMG('Property','Value',...) creates a new ENHANCEIMG using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to EnhanceImg_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      ENHANCEIMG('CALLBACK') and ENHANCEIMG('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in ENHANCEIMG.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help EnhanceImg

% Last Modified by GUIDE v2.5 11-May-2022 01:19:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @EnhanceImg_OpeningFcn, ...
                   'gui_OutputFcn',  @EnhanceImg_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
%       H = ENHANCEIMG returns the handle to a new ENHANCEIMG or the handle to
%       the existing singleton*.

if nargin && ischar(varargin{1})
  gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

global store;
global ind;

% --- Executes just before EnhanceImg is made visible.
function EnhanceImg_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject       handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%       command line (see VARARGIN)

% Choose default command line output for EnhanceImg
handles.output = hObject;

ah = axes('unit','normalized', 'position', [0 0 1 1]);
bg = imread('leaf.jpg'); imagesc(bg);
set(ah, 'handlevisibility', 'off', 'visible', 'off');
uistack(ah, 'bottom');

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes EnhanceImg wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = EnhanceImg_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
b=uigetfile()
global store;
global ind;
a= imread(b);
axes(handles.OrgImage);
imshow(a);
setappdata(0,'a',a)
setappdata(0,'b',a)
imwrite(a,'OrgImg.jpg','jpg');
store = 'bw';
ind = 2;

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'b');
imshow(a);
setappdata(0,'a',a)
```

```
% --- Executes on button press in ExitBtn.
function ExitBtn_Callback(hObject, eventdata, handles)
% hObject    handle to ExitBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close();

% --- Executes on button press in PeriodicN.
function PeriodicN_Callback(hObject, eventdata, handles)
% hObject    handle to PeriodicN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
[m n k] = size(a);
minS=min(m,n);
maxS=max(m,n);
x=(m-minS)/2;
y=(n-minS)/2;
a=a(x+1:minS,y+1:maxS-y,:);
if k > 1
  a = rgb2gray(a);
end
rowVector = (1 : m)';
period = 100;
amplitude = 0.5;
offset = 1 - amplitude;
cosVector = amplitude * (1 + cos(2 * pi * rowVector / period))/2 + offset;
ripplesImage = repmat(cosVector, [1, minS]);
noise = ripplesImage .* double(a);
axes(handles.EnhancedImg);
imshow(noise, [0 255]);
imwrite(noise,'PeriodicNImg.jpg','jpg');
setappdata(0,'a',noise)
ind = ind+2;
store = append(store,'pn');

% --- Executes on button press in SnPN.
function SnPN_Callback(hObject, eventdata, handles)
% hObject    handle to SnPN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
noise= imnoise(a,'salt & pepper', 0.02);
axes(handles.EnhancedImg);
imshow(noise);
imwrite(noise,'SnPNoiseImg.jpg','jpg');
setappdata(0,'a',noise)
ind = ind+2;
store = append(store,'sn');

% --- Executes on button press in GaussianN.
function GaussianN_Callback(hObject, eventdata, handles)
% hObject      handle to GaussianN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
noise=imnoise(a, 'gaussian');
axes(handles.EnhancedImg);
imshow(noise);
imwrite(noise,'GnoiseImg.jpg','jpg');
setappdata(0,'a',noise)
ind = ind+2;
store = append(store,'gn');


% --- Executes on button press in Red.
function Red_Callback(hObject, eventdata, handles)
% hObject      handle to Red (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
red=a;
red(:,:,2)=0;
red(:,:,3)=0;
axes(handles.EnhancedImg);
imshow(red)
```

```
imwrite(red,'redImg.jpg','jpg');
setappdata(0,'a',red)
ind = ind+2;
store = append(store,'re');


% --- Executes on button press in Green.
function Green_Callback(hObject, eventdata, handles)
% hObject    handle to Green (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
green=a;
green(:,:,1)=0;
green(:,:,3)=0;
setappdata(0,'filename', green);
setappdata(0,'ImRotation', green);
axes(handles.EnhancedImg);
imshow(green)
imwrite(green,'GreenImg.jpg','jpg');
setappdata(0,'a',green)
ind = ind+2;
store = append(store,'gr');

% --- Executes on button press in Blue.
function Blue_Callback(hObject, eventdata, handles)
% hObject    handle to Blue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
blue=a;
blue(:,:,1)=0;
blue(:,:,2)=0;
setappdata(0,'filename', blue);
setappdata(0,'ImRotation', blue);
axes(handles.EnhancedImg);
imshow(blue)
```

```
imwrite(blue,'blueImg.jpg','jpg');
setappdata(0,'a',blue)
setappdata(0,'table',blue)
ind = ind+2;
store = append(store,'bl');

% --- Executes on button press in RtoG.
function RtoG_Callback(hObject, eventdata, handles)
% hObject    handle to RtoG (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0, 'a');
a_gray=rgb2gray(a);
axes(handles.EnhancedImg);
imshow(a_gray)
imwrite(gray,'GrayImg.jpg','jpg');
setappdata(0,'a',gray)
ind = ind+2;
store = append(store,'rg');

% --- Executes on button press in RtoBW.
function RtoBW_Callback(hObject, eventdata, handles)
% hObject    handle to RtoBW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0, 'a');
a_bw=im2bw(a,.57);
axes(handles.EnhancedImg);
imshow(a_bw)
imwrite(a_bw,'RGB2bwImg.jpg','jpg');
setappdata(0,'a',a_bw)
ind = ind+2;
store = append(store,'rb');

% --- Executes on button press in Sharp.
function Sharp_Callback(hObject, eventdata, handles)
% hObject    handle to Sharp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a= getappdata(0,'a');
sharp=imsharpen(a,'Radius',2,'Amount',10);
axes(handles.EnhancedImg);
imshow(sharp);
imwrite(sharp,'SharpenImage.jpg','jpg');
setappdata(0,'a',sharp)
ind = ind+2;
store = append(store,'sh');

% --- Executes on button press in Pink.
function Pink_Callback(hObject, eventdata, handles)
% hObject      handle to Pink (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
pink=a;
pink(:,:,2)=0;
axes(handles.EnhancedImg);
imshow(pink)
imwrite(pink,'PinkImg.jpg','jpg');
setappdata(0,'a',pink)
ind = ind+2;
store = append(store,'pi');

% --- Executes on button press in Yellow.
function Yellow_Callback(hObject, eventdata, handles)
% hObject      handle to Yellow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
yellow=a;
yellow(:,:,3)=0;
axes(handles.EnhancedImg);
imshow(yellow)
imwrite(yellow,'YellImg.jpg','jpg');
```

```
setappdata(0,'a',yellow)
ind = ind+2;
store = append(store,'ye');

% --- Executes on button press in Cyan.
function Cyan_Callback(hObject, eventdata, handles)
% hObject    handle to Cyan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
cyan=a;
cyan(:,:,1)=0;
axes(handles.EnhancedImg);
imshow(cyan)
imwrite(cyan,'CyanImg.jpg','jpg');
setappdata(0,'a',cyan)
ind = ind+2;
store = append(store,'cy');

% --- Executes on button press in GaussianF.
function GaussianF_Callback(hObject, eventdata, handles)
% hObject    handle to GaussianF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
gfilter= fspecial('gaussian');
gausF = imfilter(a,gfilter);
axes(handles.EnhancedImg);
imshow(gausF);
imwrite(gausF,'GFilterImg.jpg','jpg');
setappdata(0,'a',gausF)
ind = ind+2;
store = append(store,'gf');

% --- Executes on button press in MeanF.
function MeanF_Callback(hObject, eventdata, handles)
% hObject    handle to MeanF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
h=fspecial('average',3);
mfiltimg=imfilter(a,h);
axes(handles.EnhancedImg);
imshow(mfiltimg);
imwrite(mfiltimg,'mfiltimg.jpg','jpg');
setappdata(0,'a',mfiltimg)
ind = ind+2;
store = append(store,'mn');

% --- Executes on button press in MedianF.
function MedianF_Callback(hObject, eventdata, handles)
% hObject      handle to MedianF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
a=getappdata(0,'a');
R = a(:,:,1);
G = a(:,:,2);
B = a(:,:,3);
c(:,:,1) = medfilt2(R);
c(:,:,2) = medfilt2(G);
c(:,:,3) = medfilt2(B);
axes(handles.EnhancedImg);
imshow(c);
imwrite(c,'MedFilterImg.jpg','jpg');
setappdata(0,'a',c)
ind = ind+2;
store = append(store,'md');

% --- Executes on slider movement.
function BrightSlider_Callback(hObject, eventdata, handles)
% hObject      handle to BrightSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
```

```
global store;
global ind;
a=getappdata(0,'a');
% set(handleToSlider, 'min', 0);
% set(handleToSlider, 'max', 100);
value=get(hObject,'Value');
value=value*200 - 50;
bright = value + a;
axes(handles.EnhancedImg);
imshow(bright)
imwrite(bright,'brightImg.jpg','jpg');
pause(1);
setappdata(0,'a',bright)
ind = ind+2;
store = append(store,'br');



% --- Executes on slider movement.
function ContrastSlider_Callback(hObject, eventdata, handles)
% hObject    handle to ContrastSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global store;
global ind;
a=getappdata(0,'a');
value=get(hObject,'Value');
contrastIMG=a*value*5;
axes(handles.EnhancedImg);
imshow(contrastIMG)
imwrite(contrastIMG,'contrastImg.jpg','jpg');
pause(1);
setappdata(0,'a',contrastIMG)
ind = ind+2;
store = append(store,'co');



% --- Executes on slider movement.
function VignetteSlider_Callback(hObject, eventdata, handles)
% hObject    handle to VignetteSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
value=get(hObject,'Value');
[m n k] =  size(a);
Center = [m/2, n/2];
for row=1:m
        for col=1:n
        vignette(row, col) = sqrt((row-Center(1))^2+(col-Center(2))^2);
        end
end
vignette = vignette*value / max(vignette(:));
vignette = 1 - vignette;
vignette = cat(3, vignette, vignette, vignette);
vignette = uint8(double(a) .* vignette);
axes(handles.EnhancedImg);
imshow(vignette)
imwrite(vignette,'vigneImg.jpg','jpg');
pause(1);
setappdata(0,'a',vignette)
global store;
global ind;
ind = ind+2;
store = append(store,'vi');

% --- Executes on button press in HistEq.
function HistEq_Callback(hObject, eventdata, handles)
% hObject      handle to HistEq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
a_hist= histeq(a);
axes(handles.EnhancedImg);
imshow(a_hist)
imwrite(a_hist,'HistEqImg.jpg','jpg');
setappdata(0,'a',a_hist)
global store;
global ind;
ind = ind+2;
store = append(store,'hi');


% --- Executes on button press in ViewHist.
```

```
function ViewHist_Callback(hObject, eventdata, handles)
% hObject    handle to ViewHist (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
input = a;
input = rgb2gray(input);
axes(handles.EnhancedImg)
imhist(input)
imwrite(input,'ViewHist.jpg','jpg');
setappdata(0,'a',input)
global store;
global ind;
ind = ind+2;
store = append(store,'vh');

% --- Executes on button press in Rotate.
function Rotate_Callback(hObject, eventdata, handles)
% hObject    handle to Rotate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getappdata(0,'a');
rotate=imrotate(a,90);
axes(handles.EnhancedImg);
imshow(rotate);
imwrite(rotate,'RotateImg.jpg','jpg');
setappdata(0,'a',rotate)
global store;
global ind;
ind = ind+2;
store = append(store,'ro');

% --- Executes on button press in flip.
function flip_Callback(hObject, eventdata, handles)
% hObject    handle to flip (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I2=flipdim(I,2);
axes(handles.EnhancedImg);
imshow(I2);
imwrite(I2,'FlipImg.jpg','jpg');
```

```
setappdata(0,'a',I2)
global store;
global ind;
ind = ind+2;
store = append(store,'fl');

% --- Executes on button press in powerTrans.
function powerTrans_Callback(hObject, eventdata, handles)
% hObject    handle to powerTrans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getappdata(0,'a');
a=rgb2gray(a);
a = im2double(a);
[m,n] = size(a);
ex=zeros(m,n);
gamma=0.55;
c=1;
for i=1:m
        for j=1:n
        ex(i,j)= c*(a(i,j)^gamma);
        end
end

axes(handles.EnhancedImg);
imshow(ex);
imwrite(ex,'powerTransformImg.jpg','jpg');
setappdata(0,'a',ex)
global store;
global ind;
ind = ind+2;
store = append(store,'pt');

% --- Executes on button press in logTrans.
function logTrans_Callback(hObject, eventdata, handles)
% hObject    handle to logTrans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getappdata(0,'a');
a=rgb2gray(a);
a=im2double(a);
[row,col]=size(a);
```

```
c=1;
ex=zeros(row,col);
for i=1:row
  for j=1:col
          ex(i,j)=c*log(1+a(i,j));
  end
end
axes(handles.EnhancedImg);
imshow(ex);
imwrite(ex,'logTransformImg.jpg','jpg');
setappdata(0,'a',ex)
global store;
global ind;
ind = ind+2;
store = append(store,'lt');

% --- Executes on button press in fourierTrans.
function fourierTrans_Callback(hObject, eventdata, handles)
% hObject    handle to fourierTrans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
af= fft2(a);
axes(handles.EnhancedImg);
imshow(mat2gray(log(1+abs(af))));
imwrite(af,'fourierTransform.jpg', 'jpg');
setappdata(0,'a',af)
global store;
global ind;
ind = ind+2;
store = append(store,'ft');


% --- Executes on button press in sobelEdge.
function sobelEdge_Callback(hObject, eventdata, handles)
% hObject    handle to sobelEdge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getappdata(0,'a');
m1=rgb2gray(a);
m2=double(m1);
f1=[-1 -2 1;0 0 0;1 2 -1];
```

```
[r,c]=size(m1);
rr=zeros(r-3,c-3);
for q1=1:(r-3)
        for p1=1:(c-3)
        m1=m2(q1:(q1+2),p1:(p1+2));
        res=f1.*m1;
        rr(q1,p1)=sum(sum(res));

        end
end
axes(handles.EnhancedImg);
imshow(rr);
imwrite(rr,'SobelImg.jpg','jpg');
setappdata(0,'a',rr)
global store;
global ind;
ind = ind+2;
store = append(store,'se');

% --- Executes on button press in cannyEdge.
function cannyEdge_Callback(hObject, eventdata, handles)
% hObject    handle to cannyEdge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getappdata(0,'a');
y=rgb2gray(a);

x1=edge(y,'canny');
axes(handles.EnhancedImg);
imshow(x1);
imwrite(x1,'CannyEdge.jpg','jpg');
setappdata(0,'a',x1)
global store;
global ind;
ind = ind+2;
store = append(store,'ce');


% --- Executes on button press in Undo.
function Undo_Callback(hObject, eventdata, handles)
% hObject    handle to Undo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
if ind>=2
ind = ind-2;
undo = UndoRedo(store,ind);
undo = imread(undo);
setappdata(0,'a',undo)
axes(handles.EnhancedImg);
imshow(undo);
end

% --- Executes on button press in Redo.
function Redo_Callback(hObject, eventdata, handles)
% hObject    handle to Redo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
global store;
global ind;
[m n] = size(store);
if n>ind
ind = ind+2;
redo = UndoRedo(store,ind);
redo = imread(redo);
setappdata(0,'a',redo)
axes(handles.EnhancedImg);
imshow(redo);
end

function [display] = UndoRedo(store,ind)

 temp = store(ind+1:ind+2);

 switch temp
        case 're'
        display = 'redImg.jpg';
        case 'bl'
        display = 'blueImg.jpg';
        case 'gr'
        display = 'GreenImg.jpg';
        case 'cy'
        display = 'CyanImg.jpg';
```

```
case 'pi'
display = 'PinkImg.jpg';
case 'ye'
display = 'YellImg.jpg';
case 'pn'
display = 'PeriodicNImg.jpg';
case 'sn'
display = 'SnPNoiseImg.jpg';
case 'gn'
display = 'GnoiseImg.jpg';
case 'rg'
display = 'GrayImg.jpg';
case 'rb'
display = 'RGB2bwImg.jpg';
case 'sh'
display = 'SharpenImage.jpg';
case 'gf'
display = 'GFilterImg.jpg';
case 'mn'
display = 'mfiltimg.jpg';
case 'md'
display = 'MedFilterImg.jpg';
case 'br'
display = 'brightImg.jpg';
case 'co'
display = 'contrastImg.jpg';
case 'vi'
display = 'vigneImg.jpg';
case 'hi'
display = 'HistEqImg.jpg';
case 'vh'
display = 'ViewHist.jpg';
case 'ro'
display = 'RotateImg.jpg';
case 'fl'
display = 'FlipImg.jpg';
case 'pt'
display = 'powerTransformImg.jpg';
case 'lt'
display = 'logTransformImg.jpg';
case 'ft'
display = 'fourierTransform.jpg';
```

```
        case 'se'
        display = 'SobelImg.jpg';
        case 'ce'
        display = 'CannyEdge.jpg';
        case 'bw'
        display = 'OrgImg.jpg';
        otherwise
        display = 'itallends.jpg';
 end

end
```

3. Encryption

```
function varargout = Encryption(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Encryption_OpeningFcn, ...
                   'gui_OutputFcn',  @Encryption_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end

function Encryption_OpeningFcn(hObject, eventdata, handles, varargin)


handles.output = hObject;

ah = axes('unit','normalized', 'position', [0 0 1 1]);
bg = imread('leaf.jpg'); imagesc(bg);
set(ah, 'handlevisibility', 'off', 'visible', 'off');
uistack(ah, 'bottom');
```

```
guidata(hObject, handles);
clear all;
clc;

global Img;
global EncImg;
global DecImg;

function varargout = Encryption_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function pushbutton1_Callback(hObject, eventdata, handles)

global Img;
X = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png;*.jpeg*','pick a jpge file');
Img = imread(X);
  axes(handles.axes1)
  imshow(Img);

guidata(hObject, handles);

function pushbutton2_Callback(hObject, eventdata, handles)
global Img ;
global EncImg;
[n m k] = size(Img);
pass = input("\nEnter 6 digit password: ");
key = Key(n*m,pass);
EncImg = encryptImg(Img,key);
axes(handles.axes2)
imshow(EncImg);
imwrite(EncImg,'Encoded.jpg','jpg');
guidata(hObject, handles);

function pushbutton3_Callback(hObject, eventdata, handles)
global DecImg;
global EncImg;
[n m k] = size(EncImg);
pass = input("\nEnter 6 digit password: ");
key = Key(n*m,pass);
```

```
DecImg = encryptImg(EncImg,key);
axes(handles.axes3);
imshow(DecImg);
imwrite(DecImg,'Decoded.jpg','jpg');
guidata(hObject, handles);



% --- Executes on button press in exit.
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close();

function [key] = Key(n,pass)
n = n*8;
temp = zeros(n,1,'uint8');
p =  pass * 0.000001;
t = 0;

for i = 2 : n
        t = 1 - 2* p* p;
        if (t > 0.0)
        temp(i-1) = 1;
        end
        p =  t;

end

key = zeros(n/8,1,'uint8');
for i = 1 : n/8

        for j = 1 : 8
        key(i) = key(i) + temp(j*i)* 2^ (8-j) ;
        end

end

function [ImageOut] = encryptImg(ImageIn,key)
```

```
[m n k] = size(ImageIn);
for i = 1 : n
        p = (1+(i-1)*m);
        q = (((i-1)*m)+m);
        FullKey(:,i) = key( p : q );

end

for i = 1 : k
        Img = ImageIn(:,:,i);
for x = 1 : m
        for y = 1 : n
        Image(x,y) = bitxor(Img(x,y),FullKey(x,y));
        end
end
ImageOut(:,:,i) = Image(:,:,1);
end

return;
```

4. Steganography

```
function varargout = steganography(varargin)
% STEGANOGRAPHY MATLAB code for steganography.fig
%       STEGANOGRAPHY, by itself, creates a new STEGANOGRAPHY or raises the
existing
%       singleton*.
%
%       H = STEGANOGRAPHY returns the handle to a new STEGANOGRAPHY or the
handle to
%       the existing singleton*.
%
%       STEGANOGRAPHY('CALLBACK',hObject,eventData,handles,...) calls the local
%       function named CALLBACK in STEGANOGRAPHY.M with the given input
arguments.
%
%       STEGANOGRAPHY('Property','Value',...) creates a new STEGANOGRAPHY or
raises the
%       existing singleton*.  Starting from the left, property value pairs are
%       applied to the GUI before steganography_OpeningFcn gets called.  An
```

```
%       unrecognized property name or invalid value makes property application
%       stop.  All inputs are passed to steganography_OpeningFcn via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help steganography

% Last Modified by GUIDE v2.5 02-May-2022 02:20:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @steganography_OpeningFcn, ...
                   'gui_OutputFcn',  @steganography_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before steganography is made visible.
function steganography_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject       handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
% varargin   command line arguments to steganography (see VARARGIN)

% Choose default command line output for steganography
handles.output = hObject;
```

```
ah = axes('unit','normalized', 'position', [0 0 1 1]);
bg = imread('leaf.jpg'); imagesc(bg);
set(ah, 'handlevisibility', 'off', 'visible', 'off');
uistack(ah, 'bottom');
Base = imread('pcbBase.jpg');
Message = imread('pcbMsg.jpeg');
axes(handles.axes3);
imshow(Message);
axes(handles.axes4);
imshow(Base);
guidata(hObject, handles);

% UIWAIT makes steganography wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = steganography_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata, handles)
% hObject    handle to Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=uigetfile()
filename=a;
setappdata(0,'filename', filename);
a= imread(a);
axes(handles.axes1);
imshow(a);
setappdata(0,'a',a)
setappdata(0, 'filename',a);
plot(handles.axes1, 'a')
```

```
% --- Executes on button press in Exit.
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close steganography

% --- Executes on button press in hide.
function hide_Callback(hObject, eventdata, handles)
% hObject    handle to hide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Base = imread('pcbBase.jpg');
Message = imread('pcbMsg.jpeg');
Msg = imbinarize(rgb2gray(Message));
Msg = imresize(Msg,size(Base(:,:,1)));
New = Base;
New(:,:,1) = bitset(New(:,:,1),1,Msg);
axes(handles.axes5);
imshow(New);
imwrite(New,'pcbMsgIm.bmp');
clear all;

% --- Executes on button press in read.
function read_Callback(hObject, eventdata, handles)
% hObject    handle to read (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Im = imread('pcbMsgIm.bmp');
[indexedImage, map] = rgb2ind(Im, 6);
MessageImage = bitget(Im(:,:,1),1);
Im =(logical(MessageImage));
rgbImage2 = ind2rgb(Im, map);
axes(handles.axes6);
imshow(rgbImage2);f
```

5. Epicycles

```
function varargout = epicycles(varargin)
% EPICYCLES MATLAB code for epicycles.fig
```

```
%      EPICYCLES, by itself, creates a new EPICYCLES or raises the existing
%      singleton*.
%
%      H = EPICYCLES returns the handle to a new EPICYCLES or the handle to
%      the existing singleton*.
%
%      EPICYCLES('CALLBACK',hObject,eventData,handles,...) calls the local
f

%      function named CALLBACK in EPICYCLES.M with the given input arguments.
%
%      EPICYCLES('Property','Value',...) creates a new EPICYCLES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before epicycles_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to epicycles_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help epicycles

% Last Modified by GUIDE v2.5 18-May-2022 22:43:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @epicycles_OpeningFcn, ...
                   'gui_OutputFcn',  @epicycles_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT
global n;
n = 2;

% --- Executes just before epicycles is made visible.
function epicycles_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to epicycles (see VARARGIN)

% Choose default command line output for epicycles
handles.output = hObject;

ah = axes('unit','normalized', 'position', [0 0 1 1]);
bg = imread('leaf.jpg'); imagesc(bg);
set(ah, 'handlevisibility', 'off', 'visible', 'off');
uistack(ah, 'bottom');


% Update handles structure
guidata(hObject, handles);
set_up_the_drawing(handles);
% UIWAIT makes epicycles wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = epicycles_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in button_draw_reset.
function button_draw_reset_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to button_draw_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global xy;

cla(handles.draw);
cla(handles.result);
xy = [];

set_up_the_drawing(handles);

% --- Executes on button press in button_draw_ok.
function button_draw_ok_Callback(hObject, eventdata, handles)
% hObject    handle to button_draw_ok (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global xy;
global n;
fourier_epicycles(xy(:,1), xy(:,2),n,handles);



function set_up_the_drawing(handles)
global xy;
xy = [];
axes(handles.draw);
try
        hFH = imfreehand();
if isempty(hFH)
  return;
end
xy = hFH.getPosition;
xy = [xy; xy(1,:)];
delete(hFH);
xCoordinates = xy(:, 1);
yCoordinates = xy(:, 2);
plot(xCoordinates, yCoordinates, 'r', 'LineWidth', 2);
hold off;
set(handles.text_nocircles,'String', size(xy,1));
catch me
end
```

```
% --- Executes when selected object is changed in uibuttongroup1.
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uibuttongroup1
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
switch(get(eventdata.NewValue,'Tag'))
        case 'Ascending'
        global n;
        n = 1;
        case 'Descending'
        global n;
        n = 2;
        otherwise
        global n;
        n = 0;
end
  fourier_epicycles(xy(:,1), xy(:,2),n,handles);

function fourier_epicycles(curve_x, curve_y,n,handles)
        Z = complex(curve_x(:), curve_y(:));
        N = length(Z);
        X = fft(Z, N)/N;
        freq = 0:N-1;
        radius = abs(X);
        phase = angle(X);

        if n == 1
        [radius, idx] = sort(radius, 'ascend');
        X = X(idx);
        freq = freq(idx);
        phase = phase(idx);
        elseif n == 2
        [radius, idx] = sort(radius, 'descend');
        X = X(idx);
        freq = freq(idx);
        phase = phase(idx);
        end

        time_step = 2*pi/length(X);
```

```
time = 0;
wave = [];
while 1
[x, y] = draw_epicycles(freq, radius, phase, time, wave,handles);
wave = [wave; [x,y]];
time = time + time_step;
end


function [x, y] = draw_epicycles(freq, radius, phase, time, wave, handles)
x = 0;
y = 0;
N = length(freq);
centers = zeros(N,2);
radii_lines = zeros(N,4);
for i = 1:N
prevx = x;
prevy = y;

x = x + radius(i) * cos(freq(i)*time + phase(i));
y = y + radius(i) * sin(freq(i)*time + phase(i));

centers(i,:) = [prevx, prevy];

radii_lines(i,:) = [prevx, x, prevy, y];
end

axes(handles.result);
cla;
viscircles(centers, radius, 'Color', 0.7 * [0, 1, 0], 'LineWidth', 0.1);
hold on;

plot( radii_lines(:,1:2), radii_lines(:,3:4), 'Color', [0 0 1], 'LineWidth', 0.1);
hold on;

if ~isempty(wave), plot( wave(:,1), wave(:,2), 'k', 'LineWidth', 2); hold on; end

plot( x, y, 'or', 'MarkerFaceColor', 'r');
hold off;
axis equal;
%axis off;
```

```
        drawnow;


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close();
```