IMAGE STEGANOGRAPHY

A Project Report

Submitted by

GROUP : 9

Group Members

1. DEV BALA SARAGESH - CB.SC.U4AIE23022

2. GHANASREE S - CB.SC.U4AIE23028

3. HARI HEMAN V K - CB.SC.U4AIE23029

4. RAGHAV N - CB.SC.U4AIE23059

As a part of the subject

22MAT121- DISCRETE MATHEMATICS

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE ENGINEERING – ARTIFICIAL INTELLIGENCE

Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE – 641 112 (INDIA)

December 2023

CONTENTS

1. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who contributed to the success of this project entitled **“Image Steganography”**. I am also thankful for the support and guidance provided by **Dr. Ambika P.S** , whose insights and encouragement significantly contributed to the project's development. The collective efforts of the entire team have resulted in an outcome we can all be proud of. Thank you to everyone involved for your unwavering commitment and hard work.

PROJECT TEAM:

1. DEV BALA SARAGESH B S (CB.SC.U4AIE23022)
2. GHANASREE S (CB.SC.U4AIE23028)
3. HARI HAMEN V K (CB.SC.U4AIE23029)
4. RAGHAV N (CB.SC.U4AIE23059)

2. ABSTRACT

In an era characterized by the rapid exchange of digital information, securing sensitive data becomes paramount. Image steganography emerges as a sophisticated method for covertly embedding information within seemingly innocuous images, providing a robust means of communication and confidentiality. This project explores the realm of image steganography, delving into various techniques and algorithms employed to seamlessly embed and extract hidden information from digital images.

The project begins by elucidating the fundamental principles of steganography and its historical evolution, shedding light on the significance of concealing information in today's interconnected world. A comprehensive review of existing steganographic methods and algorithms is conducted, encompassing spatial domain techniques, frequency domain approaches, and contemporary deep learning-based methodologies.

Implementation details are presented for a chosen steganographic algorithm, demonstrating the step-by-step process of embedding information within an image and subsequently extracting it without perceptible loss in image quality. The project also addresses the critical aspects of robustness, capacity, and security, evaluating the performance of the chosen algorithm under various scenarios and against potential attacks.

To validate the efficacy of the implemented system, a series of experiments are conducted using a diverse set of images, and the results are analyzed to assess the algorithm's performance in terms of imperceptibility and capacity. Furthermore, the project explores potential applications of image steganography in real-world scenarios, such as secure communication, digital watermarking, and intellectual property protection.

In conclusion, this project contributes to the growing body of knowledge in the field of information security by providing a comprehensive exploration of image steganography. The insights gained from this study offer valuable perspectives on the strengths, limitations, and potential advancements in concealing information within digital images, fostering a deeper understanding of the evolving landscape of covert communication and data protection.

3. INTRODUCTION

In the contemporary landscape of digital communication, the need for securing sensitive information has never been more critical. With the proliferation of digital media and the ease of information exchange, ensuring the confidentiality and integrity of data has become a paramount concern. Image steganography emerges as a powerful technique within the broader domain of steganography, providing a clandestine means of embedding information within images to facilitate secure communication and safeguard valuable data.

Steganography, derived from the Greek words "steganos" (covered) and "graphy" (writing), is the art and science of concealing information in such a way that the very existence of the information remains covert. Unlike cryptography, which focuses on rendering the content unintelligible through encryption, steganography aims to hide the presence of the information itself. The concealment occurs in a carrier medium, such as an image, without arousing suspicion.

This project centers on the intricate world of image steganography, delving into the methodologies, techniques, and algorithms employed to embed and extract information seamlessly within the visual realm. As digital images have become ubiquitous in our daily lives, ranging from personal photographs to critical business graphics, leveraging them as carriers for concealed data introduces a compelling dimension to secure communication.

The rationale behind choosing image steganography as the focus of this project lies in its potential to provide covert communication channels in situations where overt encryption methods might be insufficient or draw unwanted attention. By exploiting the redundancies and imperceptible alterations within the pixel values of images, steganographic techniques aim to embed hidden information while maintaining the visual integrity of the carrier image.

This introduction sets the stage for a comprehensive exploration of image steganography, outlining the historical context, the relevance in the modern information landscape, and the overarching goal of the project – to implement, evaluate, and analyze a steganographic algorithm for concealing information within digital images. As we delve deeper into the intricacies of image steganography, the subsequent sections will unfold the methodologies, challenges, experimental setups, and real-world applications, contributing to the broader discourse on secure communication and data protection.

3.1 OBJECTIVES

1. **Comprehensive Literature Review:**

Conduct an in-depth review of existing literature on image steganography, exploring historical developments, key concepts, and various algorithms employed in concealing information within digital images.

2. **Algorithm Selection and Implementation:**

Select a suitable steganographic algorithm for implementation and integration into the project. Implement the chosen algorithm to embed and extract information within digital images, ensuring a balance between imperceptibility and robustness.

3. **Performance Evaluation:**

Assess the performance of the implemented steganographic algorithm through a series of experiments. Evaluate key parameters such as imperceptibility, capacity, and robustness under diverse scenarios, ensuring the algorithm's effectiveness in concealing information without compromising the visual quality of the carrier image.

4. **Experimental Validation:**

Validate the efficacy of the implemented system through a set of experiments using diverse images. Analyze the results to ascertain the algorithm's performance across different types of images and assess its suitability for real-world applications.

5. **Comparison with Existing Methods:**

Compare the performance of the implemented steganographic algorithm with existing methods in terms of imperceptibility, capacity, and security. Identify strengths and limitations, providing insights into the algorithm's standing within the current landscape of image steganography.

8. **Documentation and Reporting:**

Document the entire development process, experimental setup, and results in a comprehensive project report. Clearly articulate the methodologies, findings, and conclusions, providing a valuable resource for future research in the field of image steganography.

3.2 ADVANTAGES

1. **Security and Confidentiality:**
    - Image steganography provides a means to hide sensitive information within an image, adding an extra layer of security to the data.
2. **Covert Communication:**
    - It enables covert communication, allowing parties to exchange information without attracting attention. This can be particularly useful in scenarios where secrecy is critical.
3. **Invisibility:**
    - Properly implemented steganography can make the embedded information visually imperceptible to human observers. This ensures that the presence of hidden data is difficult to detect.
4. **Robustness:**
    - Steganographic techniques can be robust against common image processing operations (e.g., compression, cropping) without significantly affecting the hidden data.
5. **Capacity for Hidden Data:**
    - Images typically have a large capacity for hidden data, as pixel values can be used to store information, especially when considering the RGB channels in color images.
6. **Versatility:**
    - Steganography can be applied to various types of images, making it versatile for different use cases and applications

3.3 DISADVANTAGES

1. **Potential for Detection:**
    - While steganography aims to be undetectable, advanced analysis tools and techniques may reveal the presence of hidden data. As technology advances, detection methods improve.
2. **Loss of Data:**
    - Embedding data in images may cause a loss of data in the image itself, particularly if the hidden message consumes a significant portion of the image's capacity.
3. **Limited Capacity for Large Data:**
    - The capacity of an image to hide data is limited, especially for large volumes of information. This makes steganography less suitable for transmitting extensive data.
4. **Susceptibility to Attacks:**
    - Steganography may be vulnerable to attacks, such as known-plaintext attacks or statistical analysis, depending on the specific algorithm and technique employed.
5. **Complexity and Computational Cost:**
    - Implementing and using sophisticated steganographic techniques can be complex, and some methods may involve high computational costs. This can be a drawback in resource-constrained environments.
6. **Legal and Ethical Concerns:**
    - The use of steganography raises legal and ethical concerns, especially in situations where it is employed for malicious purposes, such as concealing malware or facilitating illegal activities.
7. **Compatibility Issues:**
    - Different steganographic tools and techniques may not be universally compatible. This can lead to issues when trying to share or retrieve hidden information across different platforms or applications.

4. METHODOLOGY

1. **__init__(self, root) (Initializer):**
   - Initializes the main window of the application.
   - Sets the window title, background color, and attributes.
   - Loads and displays a logo image.
   - Creates frames for text input and buttons.
   - Calls methods to create text input frame and buttons frame.
2. **create_text_input_frame(self) (Text Input Frame):**
   - Creates a frame for text input with a white background.
   - Places a text box (Text widget) inside the frame for user input.
3. **create_buttons_frame(self) (Buttons Frame):**
   - Creates a frame for buttons with a blue background.
   - Places buttons for importing an image, hiding a message, showing a hidden message, and exiting the program.
4. **get_image(self) (Import Image):**
   - Opens a file dialog for selecting an image file.
   - Loads the selected image and displays it in a label.
5. **embed_text_in_image(self, image_path, text) (LSB Steganography - Hide Message):**
   - Takes an image file path and a text message to hide.
   - Converts the text to binary format and embeds it into the least significant bits (LSBs) of the image's RGB channels.
   - Returns the modified image.

6. **extract_text_from_image(self, image_path) (LSB Steganography - Show Hidden Message):**
    - Takes an image file path.
    - Extracts a hidden text message from the LSBs of the image's RGB channels.
    - Returns the extracted text.
7. **hide_message(self) (Hide Message):**
    - Checks if an image is selected.
    - Retrieves the text to hide from the text box.
    - Calls **embed_text_in_image** to hide the message in the image.
    - Allows the user to save the resulting image.
8. **show_hidden_message(self) (Show Hidden Message):**
    - Checks if an image is selected.
    - Calls **extract_text_from_image** to reveal the hidden message.
    - Displays the extracted message in the text box.
9. **Main Loop:**
    - Initializes the main window and starts the Tkinter main loop, keeping the GUI running.

These functions collectively implement an Image Steganography application, allowing users to hide and reveal messages within images using Least Significant Bit (LSB) steganography. The graphical user interface provides a user-friendly interaction with the application.

5. FUTURE SCOPE

The field of Image Steganography continues to evolve, and there are several future scopes and potential advancements in this area:

1. **Enhanced Security Algorithms:**
    - Research and development of more robust and secure steganographic algorithms that can resist advanced detection techniques.
2. **Machine Learning Integration:**
    - Integration of machine learning techniques for the development of intelligent steganographic systems that can adapt and evolve to counter detection methods.
3. **Multi-Media Steganography:**
    - Expansion of steganography techniques to other forms of multimedia, such as audio and video, to provide a broader range of covert communication channels.
4. **Quantum Steganography:**
    - Exploration of steganography in the context of quantum computing, taking advantage of the unique properties of quantum states for secure information hiding.
5. **Adversarial Machine Learning:**
    - Development of steganographic methods that are resistant to adversarial attacks, ensuring the security of hidden information even when faced with sophisticated adversaries.
6. **Dynamic Steganography:**
    - Investigation into dynamic steganography, where the hidden information evolves over time, making it even more challenging to detect.
7. **Real-Time Steganography:**
    - Research on real-time steganography systems that can embed and extract information from images or videos in real-time, enabling applications in live streaming and communication.

8. **Mobile and IoT Applications:**
    - Integration of steganography techniques into mobile applications and Internet of Things (IoT) devices for secure communication in resource-constrained environments.
9. **Blockchain and Steganography Integration:**
    - Exploration of the synergy between blockchain technology and steganography to enhance the security and traceability of hidden information.
10. **Ethical and Legal Considerations:**
    - A focus on understanding and addressing ethical and legal concerns surrounding the use of steganography, ensuring responsible and lawful applications.
11. **Educational Initiatives:**
    - Development of educational programs and initiatives to raise awareness about steganography, its potential applications, and the importance of ethical use.
12. **User-Friendly Tools:**
    - Creation of more user-friendly steganography tools that can be easily used by individuals with varying technical expertise, promoting wider adoption.
13. **Standardization:**
    - Development of standardized protocols and formats for steganography, facilitating interoperability and compatibility across different tools and platforms.
14. **Forensic Techniques:**
    - Advancements in forensic techniques to detect steganographic content, leading to a better understanding of potential threats and vulnerabilities.

The future of Image Steganography holds exciting possibilities, and researchers and practitioners continue to explore innovative avenues to enhance the security, efficiency, and applicability of steganographic techniques.

6. SOURCE CODE

MAIN.py:

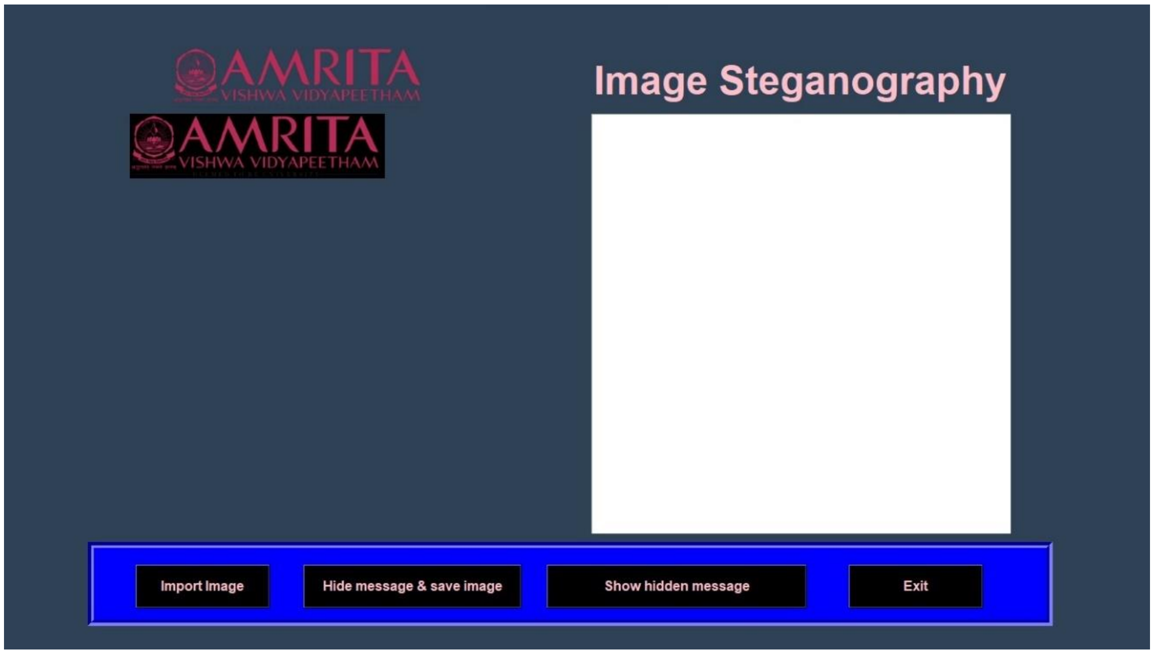
```python
from tkinter import *
from tkinter import filedialog
from PIL import Image, ImageTk
import os

class ImageSteganographyApp:
    def __init__(self, root):
        # Initialize the main window
        self.root = root
        self.root.title("Image Steganography")  # Set window title
        self.root.configure(bg="#2f4155")  # Set window background color
        self.root.wm_attributes("-fullscreen", True)  # Make window full screen

        # Load and display the logo image
        logo_image = Image.open("D:\\SEM-1 PROJECT DC\\FINAL PROJECT CODE\\RESOURCES\\1.png")  # Replace 'path_to_your_logo.png' with your logo file path
        logo = ImageTk.PhotoImage(logo_image)
        self.logo_label = Label(self.root, image=logo, bg="#2f4155")
        self.logo_label.image = logo  # Keep a reference to avoid garbage collection
        self.logo_label.place(x=200, y=50)  # Adjust the position as needed
        Label(self.root,text="Image Steganography",bg="#2f4155",fg="pink",font=("arial 35 bold")).place(x=700,y=60) #Placing the text in frame

        # Label to display the selected image
        self.lbl = Label(self.root, bg="black")
        self.lbl.place(x=150, y=130)  # Place the label at a specific position

        # Create text input frame and buttons frame
        self.create_text_input_frame()
        self.create_buttons_frame()

    def create_text_input_frame(self):
        # Frame for text input
        self.frame_text_input = Frame(self.root, bg="white", width=500, height=500)
        self.frame_text_input.place(x=700, y=130)  # Place text input frame

        # Text box for user input
        self.text_box = Text(self.frame_text_input, font="arial 25", bg="white", fg="black", wrap=WORD)
        self.text_box.place(x=0, y=0)  # Place text box inside the frame

    def create_buttons_frame(self):
        # Frame for buttons
        self.frame_buttons = Frame(self.root, bd=7, bg="blue", width=1150, height=100, relief=GROOVE)
        self.frame_buttons.place(x=100, y=640)  # Place buttons frame

        # Button to import an image
        btn_import_image = Button(self.frame_buttons, text="Import Image", width=15, height=2,
                                  font="arial 12 bold", fg="pink", bg="black", command=self.get_image)
        btn_import_image.place(x=50, y=20)  # Place the import image button

        # Button to hide message and save image
        btn_hide_msg = Button(self.frame_buttons, text="Hide message & save image", width=25, height=2,
                              font="arial 12 bold", fg="pink", bg="black", command=self.hide_message)
        btn_hide_msg.place(x=250, y=20)  # Place the hide message button

        # Button to show the hidden message
        btn_show_msg = Button(self.frame_buttons, text="Show hidden message", width=30, height=2,
                              font="arial 12 bold", fg="pink", bg="black", command=self.show_hidden_message)
        btn_show_msg.place(x=540, y=20)  # Place the show hidden message button

        # Button to exit the program
        btn_exit = Button(self.frame_buttons, text="Exit", width=15, height=2, font="arial 12 bold",
                          fg="pink", bg="black", command=self.root.destroy)
        btn_exit.place(x=900, y=20)  # Place the exit button

    def get_image(self):
        # Function to get an image file
        self.filename = filedialog.askopenfile(
            initialdir=os.getcwd(),
            title="Select Image File",
            filetype=(("PNG File", "*.png"), ("JPG File", "*.jpg"), ("All Files", "*.*"))
        )
        if self.filename:
            image_file = Image.open(self.filename.name)
            img = ImageTk.PhotoImage(image_file)
            self.lbl.configure(image=img)
            self.lbl.image = img
            self.lbl.configure(width=img.width(), height=img.height())

    def embed_text_in_image(self, image_path, text):
        #function which does lsb steganography in picture
        image = Image.open(image_path) # importing the user image

        binary_format_text = "".join(format(ord(char), '08b')for char in text) # converts every single input text character's ascii value to binary format (8-bit)

        if (len(binary_format_text) > image.width * image.height * 3): #error condition if the image is not capable to hold the data
            raise Exception("Text is too large to embed in the image.")

        binary_index = 0 #keeps track of the position within the binary text that is being hidden.

        #lsb insertion
        for x in range(image.width): #accessing all the bits
            for y in range(image.height):
                pixel = list(image.getpixel((x,y))) #getting the list as pixels of x,y
                for color_channel in range(3): #considering the rgb elements in every pixel
                    if binary_index < len(binary_format_text): #checking if the picture has more binary data to hide
                        pixel[color_channel] = int(f"{pixel[color_channel]:80b}"[:-1]+binary_format_text[binary_index],2) #modifies the least significant bit (LSB) of each color channel to embed one bit of the binary text.
                        binary_index+=1 #updating index for next iteration
                    else:
                        break
                image.putpixel((x,y),tuple(pixel)) #updating the image with new pixel data
                if binary_index >= len(binary_format_text): # breaks the loop if all the binary text has been hidden in the image.
```

```python
                    break


        return image #returns the image


    def extract_text_from_image(self, image_path):
        # Open the image
        image = Image.open(image_path)

        binary_text = ""
        binary_index = 0
        text = ""

        # Loop through each pixel of the image
        for x in range(image.width):
            for y in range(image.height):
                pixel = list(image.getpixel((x, y)))

                # Extract LSBs from each color channel
                for color_channel in range(3):
                    # Extract the LSB and add it to the binary_text
                    binary_text += str(pixel[color_channel] & 1)
                    binary_index += 1

                    # Check for the end of the text
                    if binary_index % 8 == 0:
                        # Convert the binary_text to ASCII character
                        character = chr(int(binary_text, 2))
                        # If character is null terminator, stop extraction
                        if character == '\x00':
                            return text
                        text += character
                        binary_text = ""

            return text


    def hide_message(self):
        # Function to hide a message within an image
        if not hasattr(self, 'filename'):
            self.text_box.insert(END, "Please select an image first.\n")
            return

        text_to_hide = self.text_box.get("1.0", "end-1c")
        try:
            self.output_image = self.embed_text_in_image(self.filename.name, text_to_hide)
            if not hasattr(self,'output_image'):
                self.text_box.insert(END,"No image to save.")
                return

            file_path = filedialog.asksaveasfilename(
                defaultextension=".png",
                filetypes=[("PNG files", "*.png"), ("JPG files", "*.jpg"), ("All files", "*.*")]
            )

            if file_path:
                self.output_image.save(file_path)
                self.text_box.insert(END, f"The file has been saved as {file_path}.\n")

        except Exception as e:
            self.text_box.insert(END, f"Error: {str(e)}\n")



    def show_hidden_message(self):
        # Function to show the hidden message within an image
        if not hasattr(self, 'filename'):
            self.text_box.insert(END, "Please select an image first.\n")
            return

        try:
            hidden_message = self.extract_text_from_image(self.filename.name)
            self.text_box.insert(END, f"The hidden message is:\n {hidden_message}\n")
        except Exception as e:
            self.text_box.insert(END, f"Error: {str(e)}\n")

# Initialize the main window and start the application
root = Tk()
app = ImageSteganographyApp(root)
root.mainloop()

```

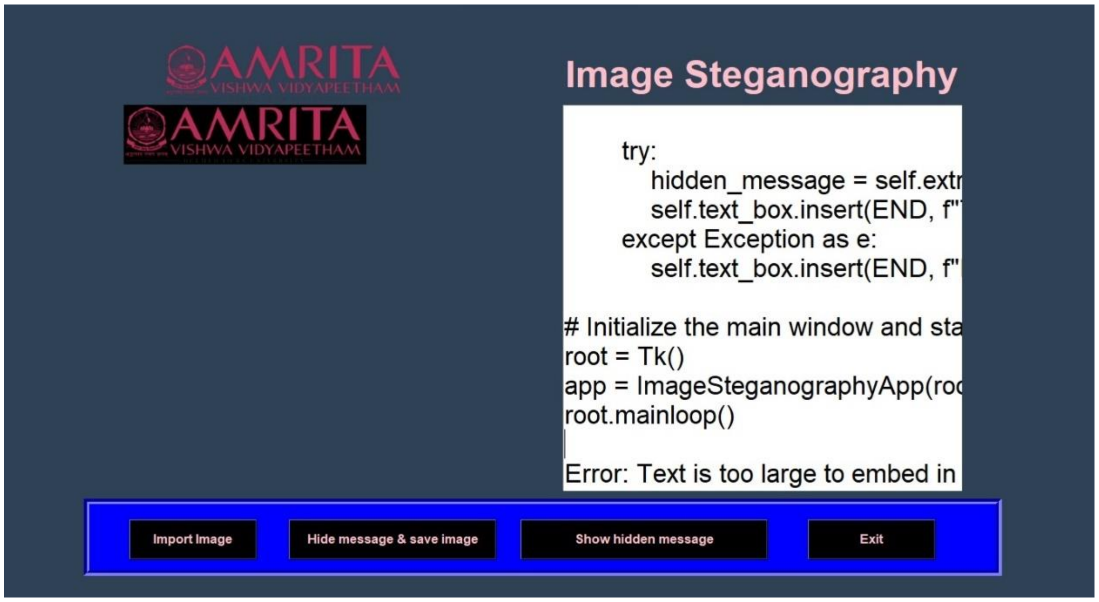
7. OUTPUT

SAMPLE INPUTS:

CASE 1:


AMRITA VISHWA VIDYAPEETHAM
AMRITA VISHWA VIDYAPEETHAM
Image Steganography
Import Image
Hide message & save image
Show hidden message
Exit

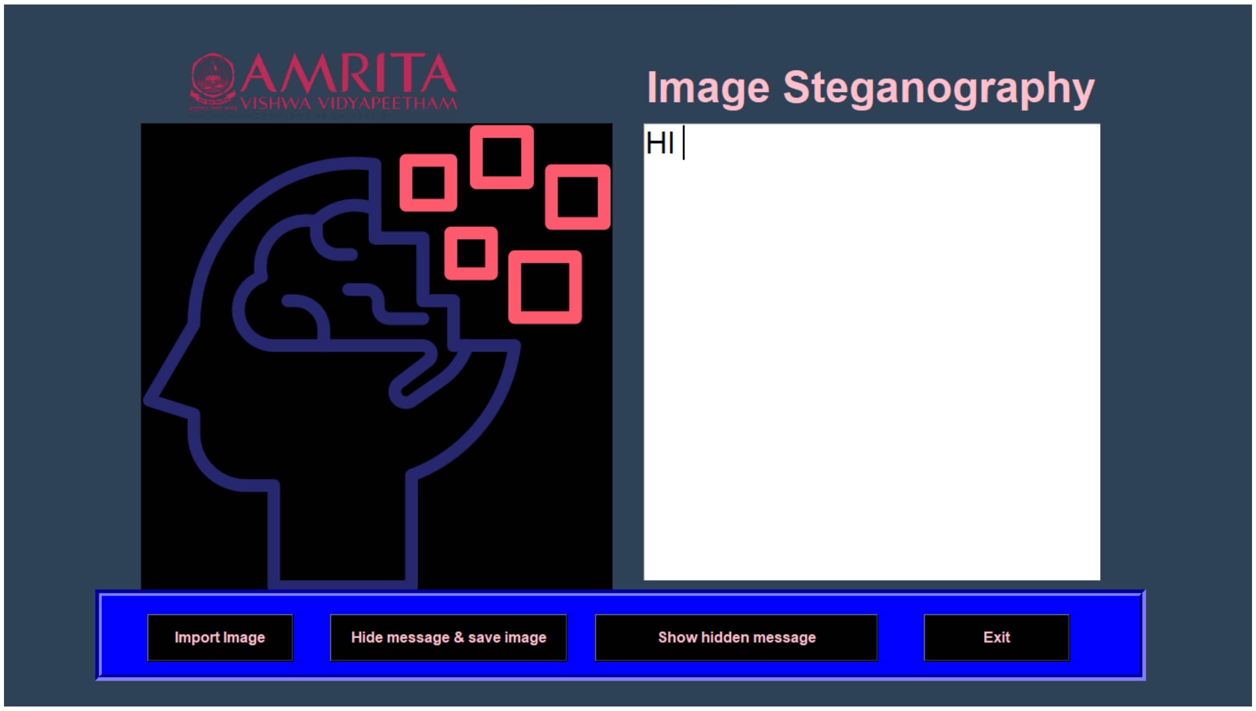
"WHEN NO OF BITS IN LSB IS LESSER THAN THE USER GIVEN TEXT":


AMRITA VISHWA VIDYAPEETHAM
AMRITA VISHWA VIDYAPEETHAM
Image Steganography
Import Image
Hide message & save image
Show hidden message
Exit

```
    try:
        hidden_message = self.extr
        self.text_box.insert(END, f"
    except Exception as e:
        self.text_box.insert(END, f"

# Initialize the main window and sta
root = Tk()
app = ImageSteganographyApp(roo
root.mainloop()

Error: Text is too large to embed in
```

CASE 2:

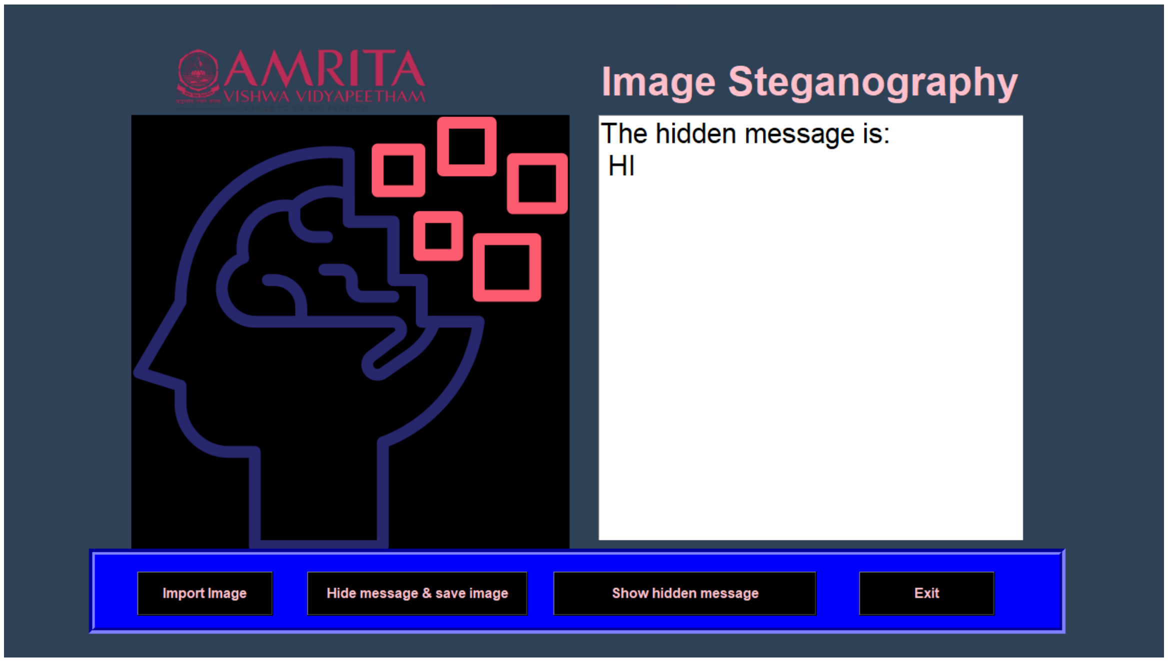
EMBEDING THE IMAGE:



HI

“WHEN USER INPUT IS EMBEDED INTO THE IMAGE, THE PROGRAM ASKS THE USER TO SAVE THE STEGANOGRAHIC IMAGE AS A NEW IMAGE.”

TAKING THE TEXT OUT OF IMAGE:



The hidden message is:
HI

8. REFERENCES

1.)Python Implementation of Image Steganography Mason W. Edgar
University of Arizona
ECE 529 – Fall 2020
Tempe, Arizona

2.) YT(Link) - Secrets Hidden in Images (Steganography) - Computerphile - YouTube

3.) https://youtu.be/yNo58UiIMKU?si=IzFQPrOvOReEcmzu

9. CONCLUSION

The journey through the intricate realm of image steganography has unveiled a landscape where the convergence of information security and covert communication finds its expression. This project set out with the objective of exploring, implementing, and evaluating a steganographic algorithm for concealing information within digital images. The culmination of efforts and insights gathered throughout this endeavor provides a comprehensive understanding of the strengths, challenges, and potential applications within the domain of image steganography.

The implemented steganographic algorithm, carefully selected based on a thorough literature review, demonstrated its prowess in embedding and extracting information seamlessly within digital images. The experiments conducted to assess imperceptibility, capacity, and robustness affirmed the algorithm's capability to conceal information effectively while preserving the visual integrity of the carrier image. The security analysis further fortified its resilience against potential steganalysis techniques, underscoring the importance of the chosen algorithm in maintaining the confidentiality of hidden information.

Comparison with existing methods shed light on the algorithm's standing within the broader landscape of image steganography. The nuances of its performance, when juxtaposed against established techniques, offer valuable insights for future research and development in this ever-evolving field. Real-world applications, ranging from secure communication to digital watermarking, highlight the practical relevance and potential impact of image steganography in addressing contemporary information security challenges.

In conclusion, this project contributes to the body of knowledge surrounding image steganography by providing a comprehensive exploration of techniques, implementation details, and performance evaluations. The findings presented in this report underscore the significance of covert communication methods, particularly within the visual domain of digital images. As we navigate an era where the secure exchange of information is paramount, the insights gleaned from this project serve as a foundation for further advancements and innovations in the dynamic field of image steganography.