Image Steganography Project Report

1. Introduction

Steganography is a method of hiding secret data within an ordinary, non-secret file or message to avoid detection. The hidden data is embedded in such a way that it is not easily noticeable. This project demonstrates a simple image steganography technique using Python, which hides messages inside image files using the Least Significant Bit (LSB) method.

2. Tools and Libraries Used

- Python 3.x
- Pillow library (for image processing)

Install dependencies using:

pip install pillow

3. Encoding Code (Hide Message)

```
from PIL import Image

def encode_image(input_image_path, output_image_path, secret_message):
    image = Image.open(input_image_path)
    encoded = image.copy()
    width, height = image.size
    index = 0

    secret_message += '###'

    binary_message = ''.join(format(ord(char), '08b') for char in secret_message)

    for row in range(height):
        for col in range(width):
            if index < len(binary_message):
                r, g, b = image.getpixel((col, row))
                r = (r & ~1) | int(binary_message[index])
                index += 1
                encoded.putpixel((col, row), (r, g, b))
            else:
                encoded.save(output_image_path)
                print("Message encoded successfully!")
                return

    print("Message too long to encode.")
```

Image Steganography Project Report

4. Decoding Code (Extract Message)

```
def decode_image(stego_image_path):
    image = Image.open(stego_image_path)
    width, height = image.size
    binary_data = ''

    for row in range(height):
        for col in range(width):
            r, g, b = image.getpixel((col, row))
            binary_data += str(r & 1)

    chars = [chr(int(binary_data[i:i+8], 2)) for i in range(0, len(binary_data), 8)]
    message = ''.join(chars)

    if '###' in message:
        return message[:message.index('###')]
    else:
        return "No hidden message found"
```

5. Testing Example

```
# Encoding Example
encode_image("input.png", "output_stego.png", "Hello! This is secret.")

# Decoding Example
print(decode_image("output_stego.png"))
```

6. Output Description

- The 'output_stego.png' image will visually look the same as 'input.png'.
- The message can be retrieved accurately using the decode function.
- Delimiter '###' ensures the end of message is clearly marked.

7. Conclusion

This project successfully demonstrates how to hide and retrieve text messages within images using simple Python code. The technique used ensures that no visible change occurs in the image, making it effective for basic secret communication.