**📌 Introduction**

This project implements an **image steganography** technique that allows hiding a **text message inside an image** using **Least Significant Bit (LSB) manipulation**.

The program supports **message encoding, decoding, and quality analysis** using metrics like:

* **MSE (Mean Squared Error)**
* **IF (Image Fidelity)**
* **PSNR (Peak Signal-to-Noise Ratio)**
* **SSIM (Structural Similarity Index Measure)**

This method stores **two bits per color channel** instead of one, effectively **doubling the storage capacity** while keeping image distortion minimal.

**📌 Features**

✅ **Message Encoding (Hiding a message inside an image)**  
✅ **Message Decoding (Extracting the hidden message from an image)**  
✅ **Image Quality Metrics Calculation** (MSE, IF, PSNR, SSIM)  
✅ **Minimal Image Distortion**  
✅ **Works on Python 3.10+ Without opencv-contrib-python**

**📌 Requirements**

**1️⃣ Install Dependencies**

sh

CopyEdit

pip install pillow numpy opencv-python

**2️⃣ Ensure You Have Python 3.10+**

Check your Python version:

sh

CopyEdit

python --version

**📌 Code Structure**

php

CopyEdit

📂 Image\_Steganography\_Project

│── final.py # Main script

│── input\_image.png # Input image for encoding

│── output\_image.png # Output image with hidden message

│── README.md # Documentation

**📌 Detailed Explanation of Code**

**1️⃣ Import Required Libraries**

python

CopyEdit

from PIL import Image

import numpy as np

import cv2

* PIL.Image: Handles image processing (open, modify, save images).
* numpy: Provides fast numerical operations on images.
* cv2: Used for **image quality analysis** (PSNR, SSIM).

**2️⃣ Define Constants**

python

CopyEdit

step = 8

* **step** controls **which bits in the pixels are modified** to store data.
* A higher step value **reduces visible distortion**.

**3️⃣ Define Bit Position Calculation**

python

CopyEdit

def get\_bit\_position(x, y, step):

bit1 = (x + y) % step

bit2 = bit1 + 1

return bit1, bit2

* Determines **which two bits** will be used to **store the message** in each pixel.
* This function ensures **even distribution of modifications** across the image.

**4️⃣ SSIM Calculation (Structural Similarity)**

python

CopyEdit

def ssim\_manual(img1, img2):

"""Computes SSIM manually for image quality comparison."""

C1 = 6.5025

C2 = 58.5225

img1 = img1.astype(np.float64)

img2 = img2.astype(np.float64)

kernel = np.ones((3, 3), np.float64) / 9

mu1 = cv2.filter2D(img1, -1, kernel)

mu2 = cv2.filter2D(img2, -1, kernel)

mu1\_sq = mu1 \*\* 2

mu2\_sq = mu2 \*\* 2

mu1\_mu2 = mu1 \* mu2

sigma1\_sq = cv2.filter2D(img1 \*\* 2, -1, kernel) - mu1\_sq

sigma2\_sq = cv2.filter2D(img2 \*\* 2, -1, kernel) - mu2\_sq

sigma12 = cv2.filter2D(img1 \* img2, -1, kernel) - mu1\_mu2

ssim\_map = ((2 \* mu1\_mu2 + C1) \* (2 \* sigma12 + C2)) / ((mu1\_sq + mu2\_sq + C1) \* (sigma1\_sq + sigma2\_sq + C2))

return ssim\_map.mean()

* This function **computes SSIM manually** to avoid needing opencv-contrib-python.
* SSIM measures **how similar two images are**.

**5️⃣ Calculate Image Quality Metrics**

python

CopyEdit

def calculate\_metrics(original\_img\_path, encoded\_img\_path):

original = cv2.imread(original\_img\_path, cv2.IMREAD\_GRAYSCALE)

encoded = cv2.imread(encoded\_img\_path, cv2.IMREAD\_GRAYSCALE)

mse = np.mean((original - encoded) \*\* 2)

if\_score = 1 - (mse / np.mean(original \*\* 2))

max\_pixel = 255.0

psnr = 20 \* np.log10(max\_pixel) - 10 \* np.log10(mse) if mse != 0 else float('inf')

ssim\_score = ssim\_manual(original, encoded)

return {

'MSE': mse,

'IF': if\_score,

'PSNR': psnr,

'SSIM': ssim\_score

}

* Computes **MSE, PSNR, IF, and SSIM** to **evaluate image distortion**.
* Higher **PSNR & SSIM** = **Better quality**.

**6️⃣ Encode Message into an Image**

python

CopyEdit

def encode\_message(image\_path, message, output\_path, step):

original\_img = Image.open(image\_path)

if original\_img.mode != 'RGB':

original\_img = original\_img.convert('RGB')

img = original\_img.copy()

binary\_message = ''.join(format(ord(char), '08b') for char in message)

binary\_message += '00000000' # End delimiter

data\_index = 0

for x in range(img.width):

for y in range(img.height):

if data\_index >= len(binary\_message):

break

pixel = list(img.getpixel((x, y)))

bit\_pos1, bit\_pos2 = get\_bit\_position(x, y, step)

for n in range(3):

if data\_index < len(binary\_message) - 1:

pixel[n] &= ~(1 << bit\_pos1)

pixel[n] |= (int(binary\_message[data\_index]) << bit\_pos1)

data\_index += 1

pixel[n] &= ~(1 << bit\_pos2)

pixel[n] |= (int(binary\_message[data\_index]) << bit\_pos2)

data\_index += 1

img.putpixel((x, y), tuple(pixel))

if data\_index >= len(binary\_message):

break

img.save(output\_path)

metrics = calculate\_metrics(image\_path, output\_path)

return metrics

* **Hides the message** inside the image using **two bits per channel**.

**7️⃣ Decode the Hidden Message**

python

CopyEdit

def decode\_message(image\_path, step):

img = Image.open(image\_path)

binary\_message = ""

for x in range(img.width):

for y in range(img.height):

pixel = img.getpixel((x, y))

bit\_pos1, bit\_pos2 = get\_bit\_position(x, y, step)

for n in range(3):

binary\_message += str((pixel[n] >> bit\_pos1) & 1)

binary\_message += str((pixel[n] >> bit\_pos2) & 1)

if binary\_message[-8:] == '00000000':

break

if binary\_message[-8:] == '00000000':

break

return ''.join(chr(int(binary\_message[i:i+8], 2)) for i in range(0, len(binary\_message)-8, 8))

* **Extracts the hidden message** from the image.

**8️⃣ User Interface**

python

CopyEdit

while True:

print("\nAdvanced Image Steganography Menu:")

print("1. Encrypt")

print("2. Decrypt")

print("3. Calculate Image Quality Metrics")

print("4. Exit")

choice = input("Enter your choice (1-4): ")

if choice == "1":

...

elif choice == "2":

...

elif choice == "3":

...

elif choice == "4":

break

* Provides a **menu-driven user interface**.

**🎯 Summary**

✅ **Hides messages efficiently**  
✅ **Works with Python 3.10+**  
✅ **Stores more data per pixel**  
✅ **Minimal image quality loss**

Let me know if you need further improvements! 🚀💻🔥