UJIAN AKHIR SEMESTER STEGANOGRAFI

PENERAPAN TEKNIK LSB (LEAST SIGNIFICANT-BIT) PADA IMAGE BERFORMAT PNG (PORTABLE NETWORK GRAPHIC)

DISUSUN OLEH:

NI KADEK EVI DIANASARI (2008561021)

DOSEN PENGAMPU:

I Komang Ari Mogi, S.Kom., M.Kom.

PROGRAM STUDI INFORMATIKA

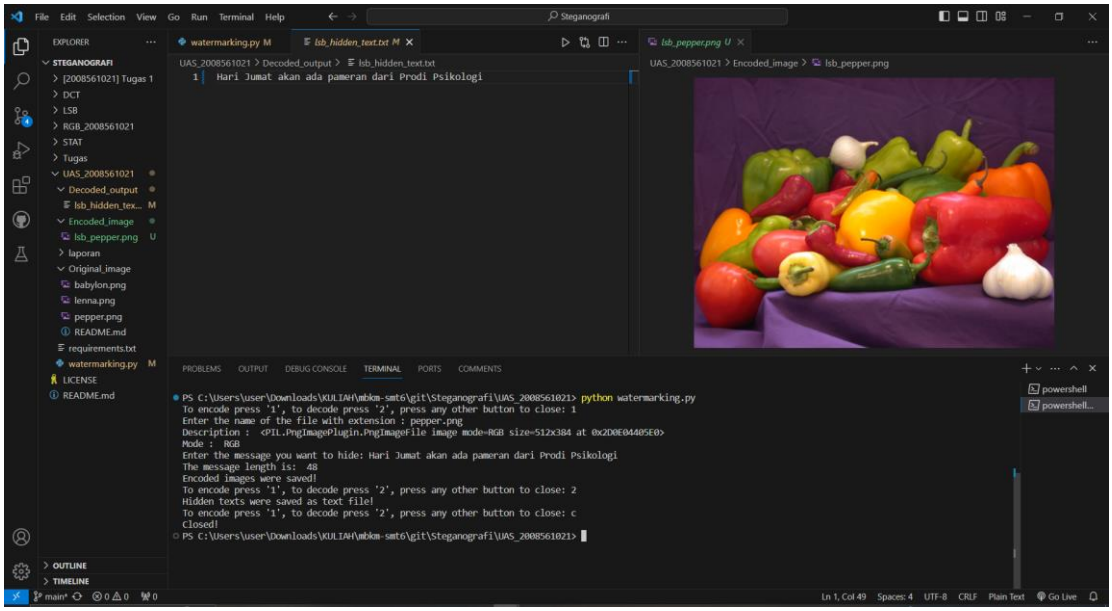
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

2023

Disini saya menggunakan Metode LSB dengan format gambar PNG yang dieksekusi. Berikut adalah penjelasan kode serta hasil yang telah dibuat :

1. **Hasil dan Eksekusi di terminal**



2. **Impor Library**

   Mengimpor beberapa pustaka yang akan digunakan :

   - Os : Modul ini digunakan untuk manipulasi direktori, seperti membuat, menghapus, dan mengganti direktori.
   - Shutil : Modul ini digunakan untuk menghapus folder dan kontennya.
   - cv2 (OpenCV) : OpenCV (Open Source Computer Vision Library) digunakan untuk operasi pengolahan citra.
   - sys : Modul ini digunakan untuk keluar dari program jika diperlukan.
   - numpy (np) : Numpy adalah pustaka untuk komputasi ilmiah dengan dukungan untuk array dan matriks. Dalam kode ini, digunakan untuk mengoperasikan matriks, terutama untuk tabel kuantisasi.
   - Itertools : Modul ini menyediakan fungsi untuk membuat iterator untuk permutasi dan kombinasi. Digunakan untuk menghasilkan iterasi selama proses tertentu.
   - matplotlib.pyplot as plt : digunakan untuk membuat plot.
   - PIL (Python Imaging Library) : Pustaka ini digunakan untuk manipulasi gambar, termasuk membuka dan menyimpan gambar dalam berbagai format.
   - Pathlib : Modul ini menyediakan kelas `Path` yang digunakan untuk bekerja dengan jalur file dan direktori.

Penggalan Code Impor Library
`import os` `import shutil`

```
import cv2
import sys
import numpy as np
import itertools
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path
```

3. **Kelas LSB**

Membuat kelas LSB yang memiliki metode untuk menyandikan dan mendekode pesan dalam gambar. Metode encode_image menyandikan pesan ke dalam gambar menggunakan LSB.

- img : Gambar asli yang akan diencode.
- msg : Pesan teks yang akan disembunyikan dalam gambar.
- length : Panjang pesan teks.
- encoded : Salinan gambar asli untuk menyimpan gambar yang sudah diencode.
- width dan height : Lebar dan tinggi gambar.
- index : Indeks untuk mengiterasi melalui pesan.
- (for row in range(height): for col in range(width):), setiap piksel dari gambar diambil dan diubah nilainya sesuai dengan pesan yang akan disembunyikan.
- Pada piksel pertama (baris=0, kolom=0), panjang pesan disematkan.
- Untuk setiap karakter berikutnya dalam pesan, ASCII karakter tersebut disematkan dalam komponen warna biru (b) dari piksel.
- Hasilnya adalah gambar yang menyematkan pesan teks di dalamnya.

Penggalan Code Metode Encode dengan LSB

```
class LSB():
    # Bagian encoding:
    def encode_image(self, img, msg):
        # Dapatkan panjang pesan
        length = len(msg)

        # Periksa apakah pesan terlalu panjang
        if length > 255:
            print("Teks terlalu panjang! (Jangan melebihi 255
karakter)")
            return False

        # Buat salinan gambar asli untuk menyimpan gambar yang
sudah diencode
```

```
        encoded = img.copy()

        # Dapatkan lebar dan tinggi gambar
        width, height = img.size

        # Inisialisasi indeks untuk mengiterasi melalui pesan
        index = 0

        # Iterasi melalui setiap piksel dalam gambar
        for row in range(height):
            for col in range(width):
                # Dapatkan nilai RGB dari piksel
                if img.mode != 'RGB':
                    r, g, b, a = img.getpixel((col, row))
                elif img.mode == 'RGB':
                    r, g, b = img.getpixel((col, row))

                # Tentukan nilai ASCII yang akan disematkan dalam
piksel
                # Piksel pertama (baris=0, kolom=0) digunakan
untuk menyimpan panjang pesan
                if row == 0 and col == 0 and index < length:
                    asc = length
                elif index <= length:
                    c = msg[index - 1]
                    asc = ord(c)
                else:
                    asc = b

                # Letakkan nilai RGB baru ke dalam gambar yang
sudah diencode
                encoded.putpixel((col, row), (r, g, asc))

                # Pindah ke karakter berikutnya dalam pesan
                index += 1

        # Kembalikan gambar yang sudah diencode
        return encoded
```

4. **Metode Pendekodean**

Metode decode_image mendekode pesan dari LSB. Kode tersebut adalah bagian dari implementasi kelas LSB untuk melakukan decoding (mengambil pesan yang disembunyikan) dari gambar yang telah diencode menggunakan metode Least Significant Bit (LSB).

- img: Gambar yang akan di-decode.
- width dan height: Lebar dan tinggi gambar.
- msg: Variabel untuk menyimpan pesan yang akan diambil.
- index: Indeks untuk mengiterasi melalui piksel gambar.
- length: Variabel untuk menyimpan panjang pesan yang tersemat di piksel pertama (baris=0, kolom=0).
- (for row in range(height): for col in range(width):), setiap piksel dari gambar diambil dan nilai-nilainya diperiksa.
- Piksel pertama (baris=0, kolom=0) menyimpan panjang pesan, yang kemudian digunakan untuk menentukan kapan harus berhenti mengambil karakter.
- Setiap nilai ASCII dari komponen warna biru (b) diambil dan dikonversi menjadi karakter. Nilai-nilai ini membentuk pesan yang disembunyikan di dalam gambar.
- Hasilnya adalah pesan yang telah diambil dari gambar yang telah diencode.

Penggalan Code Decode Image

```
    #decoding part :
    def decode_image(self,img):
        width, height = img.size
        msg = ""
        index = 0
        for row in range(height):
            for col in range(width):
                if img.mode != 'RGB':
                    r, g, b ,a = img.getpixel((col, row))
                elif img.mode == 'RGB':
                    r, g, b = img.getpixel((col, row))
                # first pixel r value is length of message
                if row == 0 and col == 0:
                    length = b
                elif index <= length:
                    msg += chr(b)
                index += 1
        lsb_decoded_image_file = "lsb_" + original_image_file
        return msg
```

5. **Pemrosesan Utama**

Menghapus folder "Encoded_image" dan "Decoded_output" jika sudah ada. Membuat folder baru dengan nama yang sama. Mendeklarasikan dua variabel sebagai variabel global.

- Bagian #deleting previous folders digunakan untuk menghapus folder-folder yang mungkin sudah ada sebelumnya. os.path.exists("Encoded_image/") memeriksa apakah folder "Encoded_image" sudah ada. Sedangkan shutil.rmtree("Encoded_image/") menghapus folder "Encoded_image" dan seluruh isinya jika sudah ada.
- Setelah menghapus folder yang sudah ada, program menciptakan folder baru "Encoded_image" dan "Decoded_output" untuk menyimpan hasil encoding dan decoding nantinya. os.makedirs("Encoded_image/") menciptakan folder baru "Encoded_image/". Sedangkan os.makedirs("Decoded_output/") menciptakan folder baru "Decoded_output/".
- original_image_file dan lsb_encoded_image_file adalah variabel global yang digunakan untuk menyimpan nama file gambar asli dan file gambar yang telah diencode. Kedua variabel ini diinisialisasi dengan string kosong ("") untuk sementara, dan nantinya akan diisi dengan nama file yang diinput oleh pengguna saat program berjalan.

Penggalan Code Pemrosesan Utama

```
#driver part :
#deleting previous folders :
if os.path.exists("Encoded_image/"):
    shutil.rmtree("Encoded_image/")
if os.path.exists("Decoded_output/"):
    shutil.rmtree("Decoded_output/")
#creating new folders :
os.makedirs("Encoded_image/")
os.makedirs("Decoded_output/")
original_image_file = ""    # to make the file name global
variable
lsb_encoded_image_file = ""
```

6. **Menu Utama dan Loop Pengguna**

Pada Kode ini menampilkan menu yang dapat dipilih user. Ada dua menu yaitu untuk mengencode dan decode. User dapat menekan angka 1 dan 2 untuk memilih menu, dan menu akan terus ditampilkan selama user menekan salah satu dari tombol tersebut yaitu Terus meminta user untuk memilih antara menyandikan, mendekode, atau menutup program. Tetapi jika user menginput diluar dari angka 1 dan 2 maka kode berhasil dieksekusi dan aplikasi ditutup.

Penggalan Code Menu Utama dan Loop Pengguna

```
while True:
    m = input("To encode press '1', to decode press '2', press any
other button to close: ")

    if m == "1":
        os.chdir("Original_image/")
        original_image_file = input("Enter the name of the file
with extension : ")
        lsb_img = Image.open(original_image_file)
        print("Description : ",lsb_img,"\nMode : ", lsb_img.mode)
        secret_msg = input("Enter the message you want to hide: ")
        print("The message length is: ",len(secret_msg))
        os.chdir("..")
        os.chdir("Encoded_image/")
        lsb_img_encoded = LSB().encode_image(lsb_img, secret_msg)
        lsb_encoded_image_file = "lsb_" + original_image_file
        lsb_img_encoded.save(lsb_encoded_image_file)
        print("Encoded images were saved!")
        os.chdir("..")

    elif m == "2":
        os.chdir("Encoded_image/")
        lsb_img = Image.open(lsb_encoded_image_file)
        os.chdir("..") #going back to parent directory
        os.chdir("Decoded_output/")
        lsb_hidden_text = LSB().decode_image(lsb_img)
        file = open("lsb_hidden_text.txt","w")
        file.write(lsb_hidden_text) # saving hidden text as text
file
        file.close()
        file.close()
        print("Hidden texts were saved as text file!")
        os.chdir("..")

    else:
        print("Closed!")
        break
```

7. **Kode Lengkap**

   Kode lengkap dapat diakses juga pada link github berikut : https://github.com/cipEpic/Steganografi/tree/main/UAS_2008561021.

Source Code Lengkap

```
import os
import shutil
import cv2
import sys
import numpy as np
import itertools
import matplotlib.pyplot as plt
from PIL import Image
from pathlib import Path

class LSB():
    # Bagian encoding:
    def encode_image(self, img, msg):
        # Dapatkan panjang pesan
        length = len(msg)

        # Periksa apakah pesan terlalu panjang
        if length > 255:
            print("Teks terlalu panjang! (Jangan melebihi 255 karakter)")
            return False

        # Buat salinan gambar asli untuk menyimpan gambar yang sudah
diencode
        encoded = img.copy()

        # Dapatkan lebar dan tinggi gambar
        width, height = img.size

        # Inisialisasi indeks untuk mengiterasi melalui pesan
        index = 0

        # Iterasi melalui setiap piksel dalam gambar
        for row in range(height):
            for col in range(width):
                # Dapatkan nilai RGB dari piksel
                if img.mode != 'RGB':
                    r, g, b, a = img.getpixel((col, row))
                elif img.mode == 'RGB':
                    r, g, b = img.getpixel((col, row))

                # Tentukan nilai ASCII yang akan disematkan dalam piksel
                # Piksel pertama (baris=0, kolom=0) digunakan untuk
menyimpan panjang pesan
                if row == 0 and col == 0 and index < length:
                    asc = length
```

```
                elif index <= length:
                    c = msg[index - 1]
                    asc = ord(c)
                else:
                    asc = b

                # Letakkan nilai RGB baru ke dalam gambar yang sudah
diencode
                encoded.putpixel((col, row), (r, g, asc))

                # Pindah ke karakter berikutnya dalam pesan
                index += 1

        # Kembalikan gambar yang sudah diencode
        return encoded

    #decoding part :
    def decode_image(self,img):
        width, height = img.size
        msg = ""
        index = 0
        for row in range(height):
            for col in range(width):
                if img.mode != 'RGB':
                    r, g, b ,a = img.getpixel((col, row))
                elif img.mode == 'RGB':
                    r, g, b = img.getpixel((col, row))
                # first pixel r value is length of message
                if row == 0 and col == 0:
                    length = b
                elif index <= length:
                    msg += chr(b)
                index += 1
        lsb_decoded_image_file = "lsb_" + original_image_file
        return msg


#driver part :
#deleting previous folders :
if os.path.exists("Encoded_image/"):
    shutil.rmtree("Encoded_image/")
if os.path.exists("Decoded_output/"):
    shutil.rmtree("Decoded_output/")
#creating new folders :
os.makedirs("Encoded_image/")
os.makedirs("Decoded_output/")
original_image_file = ""    # to make the file name global variable
lsb_encoded_image_file = ""
```

```
while True:
    m = input("To encode press '1', to decode press '2', press any other
button to close: ")

    if m == "1":
        os.chdir("Original_image/")
        original_image_file = input("Enter the name of the file with
extension : ")
        lsb_img = Image.open(original_image_file)
        print("Description : ",lsb_img,"\nMode : ", lsb_img.mode)
        secret_msg = input("Enter the message you want to hide: ")
        print("The message length is: ",len(secret_msg))
        os.chdir("..")
        os.chdir("Encoded_image/")
        lsb_img_encoded = LSB().encode_image(lsb_img, secret_msg)
        lsb_encoded_image_file = "lsb_" + original_image_file
        lsb_img_encoded.save(lsb_encoded_image_file)
        print("Encoded images were saved!")
        os.chdir("..")

    elif m == "2":
        os.chdir("Encoded_image/")
        lsb_img = Image.open(lsb_encoded_image_file)
        os.chdir("..") #going back to parent directory
        os.chdir("Decoded_output/")
        lsb_hidden_text = LSB().decode_image(lsb_img)
        file = open("lsb_hidden_text.txt","w")
        file.write(lsb_hidden_text) # saving hidden text as text file
        file.close()
        file.close()
        print("Hidden texts were saved as text file!")
        os.chdir("..")

    else:
        print("Closed!")
        break
```