LAMPIRAN 1

BERITA ACARA BIMBINGAN SKRIPSI

PANITIA PELAKSANA PROGRAM
FAKULTAS SOSIAL DAN HUMANIORA
UNIVERSITAS NURUL JADID
PROBOLINGGO JAWA TIMUR

Pf. Nurul Jadid
Karanganyar Paiton
Probolinggo 67291
T. 0888 1077 077
sosham@unuja.ac.id

BERITA ACARA BIMBINGAN SKRIPSI

1. Nama Mahasiswa : Siti Su'aibah
2. NIM : 2042200026
3. Prodi : Pendidikan Matematika
4. Judul Skripsi : KODE LINEAR UNTUK STEGANOGRAFI CITRA RGB

5. Konsultasi :

TANGGAL	MATERI BIMBINGAN	KETERANGAN KONSULTASI/ARAHAN	PARAF
19/03 2024	- Pemisahan kanal warna pada gambar RGB	- Proses encode pada masing-masing kanal.	
23/03 2024	- Program encode	- Melanjutkan program	
25/04 2024	- Proses encode (program)	- Program proses decode	
2/05 2024	- Program proses decode	- decode gambar grayscale - decode gambar RGB	
12/05 2024	- Menyimpan posisi ke dalam file teks.	- Cara membaca / membuka file di python.	
13/05 2024	- Membuka / membaca file teks. di Python	- Memisah variabel baru yang berisi list posisi	
16/05 2024	- Proses decode	- Proses decode (program)	
23/05 2024	- Proses decode (program)	- Teori coding, Penerapan matriks menjadi dua bagian	
6/06 2024	- Penerapan matriks menjadi dua bagian	- Perkalian dengan matriks generator	
20/06 2024	- Perkalian dengan matriks generator	- Penyisipan kata kode baru.	
27/06 2024	- Penyisipan pesan pada citra RGB	- Mencari nilai PSNR	
4/07 2024	- Nilai PSNR	- Menulis hasil (BAB 4)	
10/07 2024	- Bimbingan Bab 2, penulisan	- Penulisan, Bab 3 & 4	
22/07 2024	- Hasil.		

6. Bimbingan telah selesai pada tanggal: .....

Dosen Pembimbing,

Nur Hamid, Ph.D

LAMPIRAN 2

PROGRAM PYTHON

2.1 Program Steganografi (Penyisipan dan Enkode Pesan)

```
from PIL import Image, ImageOps
import numpy as np
import random
import matplotlib.pyplot as plt

##KUMPULAN FUNGSI##
#Mengubah matriks ke bilangan biner
def convert_to_binary_matrix(matrix):
    binary_matrix = np.vectorize(np.binary_repr)(matrix, width=8)
    return binary_matrix

# Fungsi untuk mengubah nilai piksel ke bilangan biner
def piksel_biner_split(nilai_piksel):
    string_biner = format(nilai_piksel, '08b')
    # Jika panjang bit 8, stringnya 4. karena 8/2=4.
    return string_biner[:4], string_biner[4:]
```

```python
# Fungsi untuk operasi logika biner XOR
def biner_xor(bin1, bin2):
    return ''.join('1' if b1 != b2 else '0' for b1, b2 in zip(bin1, bin2))

# Fungsi untuk proses perkalian antara elemen biner dengan matriks generator
def perkalian(elemen, generator):
    # Nilai awal untuk hasil operasi XOR yang akan dilakukan pada setiap iterasi
    result = '00000000'
    # Iterasi menggunakan enumerate
    for i, bit in enumerate(elemen):
        if bit == '1':
            result = biner_xor(result, generator[i][0])
    return result

# Fungsi untuk proses perkalian kedua matriks secara keseluruhan
def perkalian_matriks(matriks, generator):
    baris, kolom = matriks.shape
    result = np.empty((baris, kolom), dtype=object)

    for i in range(baris):
        for j in range(kolom):
```

```
            result[i, j] = perkalian(matriks[i, j], generator)

    return result

def size(gs):
    return len(gs), len(gs[0])

def ope_xor(x, y):
    int_x = int(x,2)
    int_y = int(y,2)

    return int_x^int_y
    #return (int(x) ^ int(y)) % 256

# Fungsi untuk mengubah bilangan bulat menjadi bilangan biner dengan panjang 8 bit
def convert_to_binary_8bit(elemen):
    return format(elemen, '08b')

# Fungsi untuk menyembunyikan pesan ke dalam gambar dengan posisi acak
def steganografi(matriks_cover, p):
    gs = matriks_cover.copy()
    bgs, kgs = size(gs) #bgs: baris gs, kgs: kolom gs, gs: cover
```

```
    bp, kp = size(p)
    total_posisi = bgs * kgs
    if bp * kp > total_posisi:
        raise ValueError("Pesan terlalu besar untuk disembunyikan di gambar cover.")

    pos = [(i // kgs, i % kgs) for i in range(total_posisi)]
    random.shuffle(pos)
    pos = pos[:bp * kp]

    idx = 0
    for i in range(bp):
        for j in range(kp):
            entri=matriks_cover[pos[idx][0]][pos[idx][1]]
            entri_biner=convert_to_binary_8bit(entri)
            gs[pos[idx][0]][pos[idx][1]] = ope_xor(entri_biner, p[i][j])
            #print(entri, p[i][j], gs[pos[idx][0]][pos[idx][1]])
            idx += 1
    return gs, pos

# Fungsi untuk menyimpan posisi ke file
def simpan_posisi(pos, file_path):
    with open(file_path, 'w') as f:
```

```
        for p in pos:
            f.write(f"{p[0]} {p[1]}\n")

# Fungsi untuk menghitung MSE
def calculate_mse(original, modified):
    mse = np.mean((original - modified) ** 2)
    return mse

# Fungsi untuk menghitung PSNR
def calculate_psnr(original, modified):
    mse = calculate_mse(original, modified)
    if mse == 0:  # Prevent division by zero
        return float('inf')
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr

##PROSES STEGANOGRAFI##
# Memuat gambar dan pesan
path_cover = 'danau.jpg'
path_pesan = 'wisuda.jpg'
```

```
cover = Image.open(path_cover)
pesan = Image.open(path_pesan)

# Mengubah gambar ke RGB dan matriks
coverRGB = cover.convert("RGB")
matriks_cover = np.array(coverRGB)

pesanRGB = pesan.convert("RGB")
matriks_pesan = np.array(pesanRGB)

# Pisahkan kanal warna
cover_r, cover_g, cover_b = coverRGB.split()
pesan_r, pesan_g, pesan_b = pesanRGB.split()


# Simpan setiap kanal warna ke file
cover_r.save('cover_kanal_merah.png')
cover_g.save('cover_kanal_hijau.png')
cover_b.save('cover_kanal_biru.png')

pesan_r.save('pesan_kanal_merah.png')
pesan_g.save('pesan_kanal_hijau.png')
```

```
pesan_b.save('pesan_kanal_biru.png')

#np.set_printoptions(threshold=np.inf)

# Mengubah gambar ke matriks
matriks_cover_r = np.array(cover_r)
matriks_cover_g = np.array(cover_g)
matriks_cover_b = np.array(cover_b)

print("\nMatriks cover Merah(R)")
print(matriks_cover_r)

print("\nMatriks cover Hijau(G)")
print(matriks_cover_g)

print("\nMatriks cover Biru(B)")
print(matriks_cover_b)

matriks_pesan_r = np.array(pesan_r)
matriks_pesan_g = np.array(pesan_g)
matriks_pesan_b = np.array(pesan_b)
```

```
print("\nMatriks pesan Merah(R)")
print(matriks_pesan_r)

print("\nMatriks pesan Hijau(G)")
print(matriks_pesan_g)

print("\nMatriks pesan Biru(B)")
print(matriks_pesan_b)

#Matriks pesan biner
matriks_cover_biner_r = convert_to_binary_matrix(matriks_cover_r)
matriks_cover_biner_g = convert_to_binary_matrix(matriks_cover_g)
matriks_cover_biner_b = convert_to_binary_matrix(matriks_cover_b)

print("\nMatriks cover biner Merah(R)")
print(matriks_cover_biner_r)

print("\nMatriks cover biner Hijau(G)")
print(matriks_cover_biner_g)

print("\nMatriks cover biner Biru(B)")
print(matriks_cover_biner_b)
```

```
#Matriks pesan biner
matriks_pesan_biner_r = convert_to_binary_matrix(matriks_pesan_r)
matriks_pesan_biner_g = convert_to_binary_matrix(matriks_pesan_g)
matriks_pesan_biner_b = convert_to_binary_matrix(matriks_pesan_b)

print("\nMatriks pesan biner Merah(R)")
print(matriks_pesan_biner_r)

print("\nMatriks pesan biner Hijau(G)")
print(matriks_pesan_biner_g)

print("\nMatriks pesan biner Biru(B)")
print(matriks_pesan_biner_b)

# Mendapatkan pesan dalam bentuk biner yang terbagi menjadi dua bagian
biner_split_matriks_r = np.vectorize(piksel_biner_split)(matriks_pesan_r)
biner_split_matriks_g = np.vectorize(piksel_biner_split)(matriks_pesan_g)
biner_split_matriks_b = np.vectorize(piksel_biner_split)(matriks_pesan_b)

# Matriks hasil dari np.vectorize akan berbentuk (2, H, W), kita ubah menjadi (H, W, 2)
biner_split_matriks_r = np.transpose(biner_split_matriks_r, (1, 2, 0))
```

```
biner_split_matriks_g = np.transpose(biner_split_matriks_g, (1, 2, 0))
biner_split_matriks_b = np.transpose(biner_split_matriks_b, (1, 2, 0))

# Mendapatkan bit pertama dan kedua dari masing-masing kanal
x_1_r = biner_split_matriks_r[:, :, 0]
x_2_r = biner_split_matriks_r[:, :, 1]

x_1_g = biner_split_matriks_g[:, :, 0]
x_2_g = biner_split_matriks_g[:, :, 1]

x_1_b = biner_split_matriks_b[:, :, 0]
x_2_b = biner_split_matriks_b[:, :, 1]

print("\nPesan bagian bit pertama Merah(R):")
print(x_1_r[:10, :10])

print("\nPesan bagian bit kedua Merah(R):")
print(x_2_r[:10, :10])

print("\nPesan bagian bit pertama Hijau(G):")
print(x_1_g[:10, :10])
```

```
print("\nPesan bagian bit kedua Hijau(G):")
print(x_2_g[:10, :10])

print("\nPesan bagian bit pertama Biru(B):")
print(x_1_b[:10, :10])

print("\nPesan bagian bit kedua Biru(B):")
print(x_2_b[:10, :10])

# Matriks Generator
G = np.array([['11110000'], ['00111100'], ['00001111'], ['10101010']])

print("\nMatriks Generator/Matriks Pembangun")
print(G)

# Melakukan perkalian matriks pesan dengan matriks generator dari masing-masing kanal warna
pesan_bagian_1_r = perkalian_matriks(x_1_r, G)
pesan_bagian_2_r = perkalian_matriks(x_2_r, G)

print("\nHasil perkalian pesan bagian bit pertama Merah(R):")
print(pesan_bagian_1_r[:10, :10])
```

```
print("\nHasil perkalian pesan bagian bit kedua Merah(R):")
print(pesan_bagian_2_r[:10, :10])

pesan_bagian_1_g = perkalian_matriks(x_1_g, G)
pesan_bagian_2_g = perkalian_matriks(x_2_g, G)

print("\nHasil perkalian pesan bagian bit pertama Hijau(G):")
print(pesan_bagian_1_g[:10, :10])

print("\nHasil perkalian pesan bagian bit kedua Hijau(G):")
print(pesan_bagian_2_g[:10, :10])

pesan_bagian_1_b = perkalian_matriks(x_1_b, G)
pesan_bagian_2_b = perkalian_matriks(x_2_b, G)

print("\nHasil perkalian pesan bagian bit pertama Biru(B):")
print(pesan_bagian_1_b[:10, :10])

print("\nHasil perkalian pesan bagian bit kedua Biru(B):")
print(pesan_bagian_2_b[:10, :10])

# Mendapatkan dimensi gambar pesan dari masing-masing kanal warna
```

```
tinggi_r, lebar_r = matriks_pesan_r.shape
tinggi_g, lebar_g = matriks_pesan_g.shape
tinggi_b, lebar_b = matriks_pesan_b.shape

# Flatten matriks bagian pertama dan kedua
flatten_bagian_1_r = pesan_bagian_1_r.flatten()
flatten_bagian_2_r = pesan_bagian_2_r.flatten()

flatten_bagian_1_g = pesan_bagian_1_g.flatten()
flatten_bagian_2_g = pesan_bagian_2_g.flatten()

flatten_bagian_1_b = pesan_bagian_1_b.flatten()
flatten_bagian_2_b = pesan_bagian_2_b.flatten()

# Gabungkan kedua bagian
gabungan_flatten_r = np.concatenate((flatten_bagian_1_r, flatten_bagian_2_r))
gabungan_flatten_g = np.concatenate((flatten_bagian_1_g, flatten_bagian_2_g))
gabungan_flatten_b = np.concatenate((flatten_bagian_1_b, flatten_bagian_2_b))

# Reshape menjadi ukuran height x (2 * width)
gabungan_matriks_r = gabungan_flatten_r.reshape((2*tinggi_r, lebar_r))
gabungan_matriks_g = gabungan_flatten_g.reshape((2*tinggi_g, lebar_g))
```

```
gabungan_matriks_b = gabungan_flatten_b.reshape((2*tinggi_b, lebar_b))

print("\nMatriks gabungan kata kode Merah(R):")
print(gabungan_matriks_r)

print("\nMatriks gabungan kata kode Hijau(G):")
print(gabungan_matriks_g)

print("\nMatriks gabungan kata kode Biru(B):")
print(gabungan_matriks_b)

# Proses Steganografi: Menyisipkan pesan
hasil_steganografi_r, pos_r = steganografi(matriks_cover_r, gabungan_matriks_r)
hasil_steganografi_g, pos_g = steganografi(matriks_cover_g, gabungan_matriks_g)
hasil_steganografi_b, pos_b = steganografi(matriks_cover_b, gabungan_matriks_b)

# Simpan hasil steganografi untuk verifikasi
Image.fromarray(np.uint8(hasil_steganografi_r)).save("stego_Jamur_r.png")
Image.fromarray(np.uint8(hasil_steganografi_g)).save("stego_Jamur_g.png")
Image.fromarray(np.uint8(hasil_steganografi_b)).save("stego_Jamur_b.png")

hasil_stego_rgb = np.stack((hasil_steganografi_r, hasil_steganografi_g, hasil_steganografi_b), axis=-1)
```

```
# Konversi matriks pesan tersembunyi ke dalam bentuk gambar
hasil_stego = Image.fromarray(np.uint8(hasil_stego_rgb))
Image.fromarray(np.uint8(hasil_stego)).save("stego_Jamur_rgb.png")


# Menyimpan posisi ke file
simpan_posisi(pos_r, "posisi_r.txt")
simpan_posisi(pos_g, "posisi_g.txt")
simpan_posisi(pos_b, "posisi_b.txt")

##Nilai PSNR##
# Membaca kembali gambar hasil steganografi dan cover asli
hasil_stego = Image.open("stego_Jamur_rgb.png")

# Mengubah gambar ke RGB
stegoRGB = hasil_stego.convert("RGB")

# Mengubah Gambar stego ke bentuk matriks
matriks_stego = np.array(stegoRGB)

# Pisahkan kanal warna
```

```
stegoRGB_r, stegoRGB_g, stegoRGB_b = stegoRGB.split()

matriks_stegoRGB_r = np.array(stegoRGB_r)
matriks_stegoRGB_g = np.array(stegoRGB_g)
matriks_stegoRGB_b = np.array(stegoRGB_b)

print("\nMatriks stego Merah(R)")
print(matriks_stegoRGB_r)

print("\nMatriks stego Hijau(G)")
print(matriks_stegoRGB_g)

print("\nMatriks stego Biru(B)")
print(matriks_stegoRGB_b)


# Menghitung MSE
original_r= np.array(matriks_cover_r)
modified_r= np.array(matriks_stegoRGB_r)

original_g= np.array(matriks_cover_g)
modified_g= np.array(matriks_stegoRGB_g)
```

```
original_b= np.array(matriks_cover_b)
modified_b= np.array(matriks_stegoRGB_b)

mse_r = calculate_mse(original_r, modified_r)
print(f"MSE_r: {mse_r}")

mse_g = calculate_mse(original_g, modified_g)
print(f"MSE_g: {mse_g}")

mse_b = calculate_mse(original_b, modified_b)
print(f"MSE_b: {mse_b}")

# Menghitung nilai PSNR antara gambar cover asli dan gambar hasil steganografi
psnr_value_rgb = calculate_psnr(matriks_cover, matriks_stego)
print(f"Nilai PSNR Gabungan (RGB): {psnr_value_rgb} dB")


##Hasil Stego##

# Buat subplot untuk menampilkan keempat gambar
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
```

```
# Tampilkan gambar-gambar pada subplot
axs[0, 0].imshow(hasil_steganografi_r, cmap='gray')
axs[0, 0].set_title('Hasil Steganografi R')
axs[0, 0].axis('off')

axs[0, 1].imshow(hasil_steganografi_g, cmap='gray')
axs[0, 1].set_title('Hasil Steganografi G')
axs[0, 1].axis('off')

axs[1, 0].imshow(hasil_steganografi_b, cmap='gray')
axs[1, 0].set_title('Hasil Steganografi B')
axs[1, 0].axis('off')

axs[1, 1].imshow(hasil_stego_rgb)
axs[1, 1].set_title('Hasil Steganografi RGB')
axs[1, 1].axis('off')

# Tampilkan subplot
plt.show()
```

2.2 Program Steganografi (Ekstraksi Pesan, Dekode dan Mencari Nilai PSNR)

```
from PIL import Image, ImageOps
import numpy as np
import random
import matplotlib.pyplot as plt

##KUMPULAN FUNGSI##
#Mengubah matriks ke bilangan biner
def convert_to_binary_matrix(matrix):
    binary_matrix = np.vectorize(np.binary_repr)(matrix, width=8)
    return binary_matrix

# Fungsi untuk mengubah nilai piksel ke bilangan biner
def piksel_biner_split(nilai_piksel):
    string_biner = format(nilai_piksel, '08b')
    # Jika panjang bit 8, stringnya 4. karena 8/2=4. atau nanti menyesuaikan dengan panjang bit dan dibagi 2.
    return string_biner[:4], string_biner[4:]

# Fungsi untuk operasi logika biner XOR
def biner_xor(bin1, bin2):
    return ''.join('1' if b1 != b2 else '0' for b1, b2 in zip(bin1, bin2))
```

```
# Fungsi untuk proses perkalian antara elemen biner dengan matriks generator
def perkalian(elemen, generator):
    # Nilai awal untuk hasil operasi XOR yang akan dilakukan pada setiap iterasi
    result = '00000000'
    # Iterasi menggunakan enumerate
    for i, bit in enumerate(elemen):
        if bit == '1':
            result = biner_xor(result, generator[i][0])
    return result

# Fungsi untuk proses perkalian kedua matriks secara keseluruhan
def perkalian_matriks(matriks, generator):
    baris, kolom = matriks.shape
    result = np.empty((baris, kolom), dtype=object)

    for i in range(baris):
        for j in range(kolom):
            result[i, j] = perkalian(matriks[i, j], generator)

    return result
```

```
def size(gs):
    return len(gs), len(gs[0])

def ope_xor(x, y):
    int_x = int(x,2)
    int_y = int(y,2)

    return int_x^int_y
    #return (int(x) ^ int(y)) % 256

# Fungsi untuk mengubah bilangan bulat menjadi bilangan biner dengan panjang 8 bit
def convert_to_binary_8bit(elemen):
    return format(elemen, '08b')

# Fungsi untuk menyembunyikan pesan ke dalam gambar dengan posisi acak
def steganografi(matriks_cover, p):
    gs = matriks_cover.copy()
    bgs, kgs = size(gs) #bgs: baris gs, kgs: kolom gs, gs: cover
    bp, kp = size(p)
    total_posisi = bgs * kgs
    if bp * kp > total_posisi:
        raise ValueError("Pesan terlalu besar untuk disembunyikan di gambar cover.")
```

```
    pos = [(i // kgs, i % kgs) for i in range(total_posisi)]
    random.shuffle(pos)
    pos = pos[:bp * kp]

    idx = 0
    for i in range(bp):
        for j in range(kp):
            entri=matriks_cover[pos[idx][0]][pos[idx][1]]
            entri_biner=convert_to_binary_8bit(entri)
            gs[pos[idx][0]][pos[idx][1]] = ope_xor(entri_biner, p[i][j])
            #print(entri, p[i][j], gs[pos[idx][0]][pos[idx][1]])
            idx += 1
    return gs, pos

# Fungsi untuk menyimpan posisi ke file
def simpan_posisi(pos, file_path):
    with open(file_path, 'w') as f:
        for p in pos:
            f.write(f"{p[0]} {p[1]}\n")

# Fungsi untuk memuat posisi dari file
```

```python
def muat_posisi(file_path):
    pos = []
    with open(file_path, 'r') as f:
        lines = f.readlines()
        for line in lines:
            x, y = map(int, line.strip().split())
            pos.append((x, y))
    return pos

# Fungsi untuk mendekode pesan dari gambar
def dekode_steganografi(gs, pos, matriks_cover):
    dekode_pesan = []
    for p in pos:
        #mengubah masing-masing entri ke bentuk biner
        pesan_biner = convert_to_binary_8bit(gs[p[0],p[1]])
        cover_biner = convert_to_binary_8bit(matriks_cover[p[0],p[1]])
        dekode_pesan.append(ope_xor(pesan_biner,cover_biner))
    return dekode_pesan

# Fungsi untuk menghitung MSE
def calculate_mse(original, modified):
    mse = np.mean((original - modified) ** 2)
```

```
    return mse

# Fungsi untuk menghitung PSNR
def calculate_psnr(original, modified):
    mse = calculate_mse(original, modified)
    if mse == 0:  # Prevent division by zero
        return float('inf')
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr

#Fungsi untuk pengembalian kata kode ke bentuk pesan biner 4-bit
def pesan_4bit(kata_kode):
    a = int(kata_kode[1])
    b = (int(kata_kode[3]) - int(kata_kode[1])) % 2
    c = int(kata_kode[7])
    d = (int(kata_kode[0]) - int(kata_kode[1])) % 2
    return np.array([a, b, c, d])

#Fungsi untuk konversi biner ke integer
def konversi_biner_ke_int(matriks):
    return np.array([[int(bit, 2) for bit in baris] for baris in matriks])
```

```
#Fungsi untuk membuat matriks dari hasil pesan biner 4 bit
def proses_matriks(matriks):
    hasil = []
    for baris in matriks:
        hasil_baris = []
        for elemen in baris:
            kata_kode = format(elemen, '08b')
            pesan_kode = pesan_4bit(kata_kode)
            pesan_biner = ''.join(map(str, pesan_kode))
            hasil_baris.append(pesan_biner)
        hasil.append(hasil_baris)
    return np.array(hasil)

#Fungsi untuk menggabungkan 2 matriks pesan biner 4-bit menjadi 8-bit
def gabungkan_matriks(matriks1, matriks2):
    hasil_gabungan = []
    for i in range(matriks1.shape[0]):
        baris_gabungan = []
        for j in range(matriks1.shape[1]):
            elemen_gabungan = matriks1[i, j] + matriks2[i, j]
            baris_gabungan.append(elemen_gabungan)
```

```
        hasil_gabungan.append(baris_gabungan)
    return np.array(hasil_gabungan)


##PROSES STEGANOGRAFI##
# Memuat gambar dan pesan
path_cover = 'polos.jpg'
path_pesan = 'jamur.jpg'
path_stego = 'stego_Jamur_rgb.png'

cover = Image.open(path_cover)
pesan = Image.open(path_pesan)
stego = Image.open(path_stego)

# Mengubah gambar ke RGB dan matriks
coverRGB = cover.convert("RGB")
matriks_cover = np.array(coverRGB)

pesanRGB = pesan.convert("RGB")
matriks_pesan = np.array(pesanRGB)

stegoRGB = stego.convert("RGB")
```

```
matriks_stego = np.array(stegoRGB)

# Pisahkan kanal warna
cover_r, cover_g, cover_b = coverRGB.split()
pesan_r, pesan_g, pesan_b = pesanRGB.split()
stego_r, stego_g, stego_b = stegoRGB.split()

# Mengubah gambar ke matriks
matriks_cover_r = np.array(cover_r)
matriks_cover_g = np.array(cover_g)
matriks_cover_b = np.array(cover_b)

print("\nMatriks cover Merah(R)")
print(matriks_cover_r)

print("\nMatriks cover Hijau(G)")
print(matriks_cover_g)

print("\nMatriks cover Biru(B)")
print(matriks_cover_b)

matriks_pesan_r = np.array(pesan_r)
```

```
matriks_pesan_g = np.array(pesan_g)
matriks_pesan_b = np.array(pesan_b)

matriks_stego_r = np.array(stego_r)
matriks_stego_g = np.array(stego_g)
matriks_stego_b = np.array(stego_b)

print("\nMatriks stego Merah(R)")
print(matriks_stego_r)

print("\nMatriks stego Hijau(G)")
print(matriks_stego_g)

print("\nMatriks stego Biru(B)")
print(matriks_stego_b)

#Matriks pesan biner
matriks_cover_biner_r = convert_to_binary_matrix(matriks_cover_r)
matriks_cover_biner_g = convert_to_binary_matrix(matriks_cover_g)
matriks_cover_biner_b = convert_to_binary_matrix(matriks_cover_b)

print("\nMatriks cover biner Merah(R)")
```

```
print(matriks_cover_biner_r)

print("\nMatriks cover biner Hijau(G)")
print(matriks_cover_biner_g)

print("\nMatriks cover biner Biru(B)")
print(matriks_cover_biner_b)

matriks_pesan_biner_r = convert_to_binary_matrix(matriks_pesan_r)
matriks_pesan_biner_g = convert_to_binary_matrix(matriks_pesan_g)
matriks_pesan_biner_b = convert_to_binary_matrix(matriks_pesan_b)

matriks_stego_biner_r = convert_to_binary_matrix(matriks_stego_r)
matriks_stego_biner_g = convert_to_binary_matrix(matriks_stego_g)
matriks_stego_biner_b = convert_to_binary_matrix(matriks_stego_b)

print("\nMatriks stego biner Merah(R)")
print(matriks_stego_biner_r)

print("\nMatriks stego biner Hijau(G)")
print(matriks_stego_biner_g)
```

```
print("\nMatriks stego biner Biru(B)")
print(matriks_stego_biner_b)

# Mendapatkan pesan dalam bentuk biner yang terbagi menjadi dua bagian
biner_split_matriks_r = np.vectorize(piksel_biner_split)(matriks_pesan_r)
biner_split_matriks_g = np.vectorize(piksel_biner_split)(matriks_pesan_g)
biner_split_matriks_b = np.vectorize(piksel_biner_split)(matriks_pesan_b)

# Matriks hasil dari np.vectorize akan berbentuk (2, H, W), kita ubah menjadi (H, W, 2)
biner_split_matriks_r = np.transpose(biner_split_matriks_r, (1, 2, 0))
biner_split_matriks_g = np.transpose(biner_split_matriks_g, (1, 2, 0))
biner_split_matriks_b = np.transpose(biner_split_matriks_b, (1, 2, 0))

# Mendapatkan bit pertama dan kedua dari masing-masing kanal
x_1_r = biner_split_matriks_r[:, :, 0]
x_2_r = biner_split_matriks_r[:, :, 1]

x_1_g = biner_split_matriks_g[:, :, 0]
x_2_g = biner_split_matriks_g[:, :, 1]

x_1_b = biner_split_matriks_b[:, :, 0]
x_2_b = biner_split_matriks_b[:, :, 1]
```

```
# Matriks Generator
G = np.array([['11110000'], ['00111100'], ['00001111'], ['10101010']])

# Melakukan perkalian matriks pesan dengan matriks generator dari masing-masing kanal warna
pesan_bagian_1_r = perkalian_matriks(x_1_r, G)
pesan_bagian_2_r = perkalian_matriks(x_2_r, G)


pesan_bagian_1_g = perkalian_matriks(x_1_g, G)
pesan_bagian_2_g = perkalian_matriks(x_2_g, G)

pesan_bagian_1_b = perkalian_matriks(x_1_b, G)
pesan_bagian_2_b = perkalian_matriks(x_2_b, G)

# Mendapatkan dimensi gambar pesan dari masing-masing kanal warna
tinggi_r, lebar_r = matriks_pesan_r.shape
tinggi_g, lebar_g = matriks_pesan_g.shape
tinggi_b, lebar_b = matriks_pesan_b.shape

# Flatten matriks bagian pertama dan kedua
```

```
flatten_bagian_1_r = pesan_bagian_1_r.flatten()
flatten_bagian_2_r = pesan_bagian_2_r.flatten()

flatten_bagian_1_g = pesan_bagian_1_g.flatten()
flatten_bagian_2_g = pesan_bagian_2_g.flatten()

flatten_bagian_1_b = pesan_bagian_1_b.flatten()
flatten_bagian_2_b = pesan_bagian_2_b.flatten()

# Gabungkan kedua bagian
gabungan_flatten_r = np.concatenate((flatten_bagian_1_r, flatten_bagian_2_r))
gabungan_flatten_g = np.concatenate((flatten_bagian_1_g, flatten_bagian_2_g))
gabungan_flatten_b = np.concatenate((flatten_bagian_1_b, flatten_bagian_2_b))

# Reshape menjadi ukuran height x (2 * width)
gabungan_matriks_r = gabungan_flatten_r.reshape((2*tinggi_r, lebar_r))
gabungan_matriks_g = gabungan_flatten_g.reshape((2*tinggi_g, lebar_g))
gabungan_matriks_b = gabungan_flatten_b.reshape((2*tinggi_b, lebar_b))


# Proses Dekode: Mengekstrak pesan
# Memuat posisi dari file
```

```
posisi_termuat_r = muat_posisi("posisi_r.txt")
posisi_termuat_g = muat_posisi("posisi_g.txt")
posisi_termuat_b = muat_posisi("posisi_b.txt")

# Dekode pesan dari gambar hasil steganografi
pesan_tersembunyi_r = dekode_steganografi(matriks_stego_r, posisi_termuat_r, matriks_cover_r)
pesan_tersembunyi_g = dekode_steganografi(matriks_stego_g, posisi_termuat_g, matriks_cover_g)
pesan_tersembunyi_b = dekode_steganografi(matriks_stego_b, posisi_termuat_b, matriks_cover_b)

# Ubah pesan tersembunyi ke dalam bentuk matriks dengan elemen bilangan biner 8 bit
bp_r_r, kp_r_r = gabungan_matriks_r.shape
matriks_pesan_tersembunyi_r = np.array(pesan_tersembunyi_r).reshape(bp_r_r, kp_r_r)

bp_r_g, kp_r_g = gabungan_matriks_g.shape
matriks_pesan_tersembunyi_g = np.array(pesan_tersembunyi_g).reshape(bp_r_g, kp_r_g)

bp_r_b, kp_r_b = gabungan_matriks_b.shape
matriks_pesan_tersembunyi_b = np.array(pesan_tersembunyi_b).reshape(bp_r_b, kp_r_b)

#Matriks pesan biner
pesan_tersembunyi_r = np.vectorize(convert_to_binary_8bit)
matriks_pesan_tersembunyi_bin_r = pesan_tersembunyi_r(matriks_pesan_tersembunyi_r)
```

```
pesan_tersembunyi_g = np.vectorize(convert_to_binary_8bit)
matriks_pesan_tersembunyi_bin_g = pesan_tersembunyi_g(matriks_pesan_tersembunyi_g)

pesan_tersembunyi_b = np.vectorize(convert_to_binary_8bit)
matriks_pesan_tersembunyi_bin_b = pesan_tersembunyi_b(matriks_pesan_tersembunyi_b)

print("\nPesan tersembunyi Merah(R):")
print(matriks_pesan_tersembunyi_bin_r)

print("\nPesan tersembunyi Hijau(G):")
print(matriks_pesan_tersembunyi_bin_g)

print("\nPesan tersembunyi Biru(B):")
print(matriks_pesan_tersembunyi_bin_b)


##Mengembalikan kata kode yang sudah digabung##
# Flatten matriks gabungan
flatten_gabungan_r = matriks_pesan_tersembunyi_bin_r.flatten()
flatten_gabungan_g = matriks_pesan_tersembunyi_bin_g.flatten()
flatten_gabungan_b = matriks_pesan_tersembunyi_bin_b.flatten()
```

```
# Tentukan ukuran asli masing-masing matriks
size_matriks_r = tinggi_r * lebar_r
size_matriks_g = tinggi_g * lebar_g
size_matriks_b = tinggi_b * lebar_b

# Pisahkan menjadi dua matriks
flatten_bagian_1_r = flatten_gabungan_r[:size_matriks_r]
flatten_bagian_2_r = flatten_gabungan_r[size_matriks_r:]

flatten_bagian_1_g = flatten_gabungan_g[:size_matriks_g]
flatten_bagian_2_g = flatten_gabungan_g[size_matriks_g:]

flatten_bagian_1_b = flatten_gabungan_b[:size_matriks_b]
flatten_bagian_2_b = flatten_gabungan_b[size_matriks_b:]

# Reshape kembali ke ukuran asli
matriks_bagian_1_r = flatten_bagian_1_r.reshape((tinggi_r, lebar_r))
matriks_bagian_2_r = flatten_bagian_2_r.reshape((tinggi_r, lebar_r))

print("\nKata kode bagian 1 Merah(R):")
print(matriks_bagian_1_r)
```

```
print("\nKata kode bagian 2 Merah(R):")
print(matriks_bagian_2_r)

matriks_bagian_1_g = flatten_bagian_1_g.reshape((tinggi_g, lebar_g))
matriks_bagian_2_g = flatten_bagian_2_g.reshape((tinggi_g, lebar_g))

print("\nKata kode bagian 1 Hijau(G):")
print(matriks_bagian_1_g)

print("\nKata kode bagian 2 Hijau(G):")
print(matriks_bagian_2_g)

matriks_bagian_1_b = flatten_bagian_1_b.reshape((tinggi_b, lebar_b))
matriks_bagian_2_b = flatten_bagian_2_b.reshape((tinggi_b, lebar_b))

print("\nKata kode bagian 1 Biru(B):")
print(matriks_bagian_1_b)

print("\nKata kode bagian 2 Biru(B):")
print(matriks_bagian_2_b)
```

```
# Matriks biner
matriks_bagian_pertama_r = np.array(matriks_bagian_1_r)
matriks_bagian_kedua_r = np.array(matriks_bagian_2_r)

matriks_bagian_pertama_g = np.array(matriks_bagian_1_g)
matriks_bagian_kedua_g = np.array(matriks_bagian_2_g)

matriks_bagian_pertama_b = np.array(matriks_bagian_1_b)
matriks_bagian_kedua_b = np.array(matriks_bagian_2_b)

# Konversi string biner ke integer
matriks_1_r = konversi_biner_ke_int(matriks_bagian_pertama_r)
matriks_2_r = konversi_biner_ke_int(matriks_bagian_kedua_r)

matriks_1_g = konversi_biner_ke_int(matriks_bagian_pertama_g)
matriks_2_g = konversi_biner_ke_int(matriks_bagian_kedua_g)

matriks_1_b = konversi_biner_ke_int(matriks_bagian_pertama_b)
matriks_2_b = konversi_biner_ke_int(matriks_bagian_kedua_b)

# Proses pesan dari setiap elemen matriks
```

```
pesan_bagian_pertama_r = proses_matriks(matriks_1_r)
pesan_bagian_kedua_r = proses_matriks(matriks_2_r)

pesan_bagian_pertama_g = proses_matriks(matriks_1_g)
pesan_bagian_kedua_g = proses_matriks(matriks_2_g)

pesan_bagian_pertama_b = proses_matriks(matriks_1_b)
pesan_bagian_kedua_b = proses_matriks(matriks_2_b)

# Gabungkan matriks
hasil_gabungan_r = gabungkan_matriks(pesan_bagian_pertama_r, pesan_bagian_kedua_r)
hasil_gabungan_g = gabungkan_matriks(pesan_bagian_pertama_g, pesan_bagian_kedua_g)
hasil_gabungan_b = gabungkan_matriks(pesan_bagian_pertama_b, pesan_bagian_kedua_b)

matriks_gabungan_r=konversi_biner_ke_int(hasil_gabungan_r)
matriks_gabungan_g=konversi_biner_ke_int(hasil_gabungan_g)
matriks_gabungan_b=konversi_biner_ke_int(hasil_gabungan_b)

# Simpan hasil steganografi untuk verifikasi
Image.fromarray(np.uint8(matriks_gabungan_r)).save("hasil_r.png")
Image.fromarray(np.uint8(matriks_gabungan_g)).save("hasil_g.png")
Image.fromarray(np.uint8(matriks_gabungan_b)).save("hasil_b.png")
```

```
hasil_pesan_rgb = np.stack((matriks_gabungan_r, matriks_gabungan_g, matriks_gabungan_b), axis=-1)

# Konversi matriks pesan tersembunyi ke dalam bentuk gambar
hasil_pesan = Image.fromarray(np.uint8(hasil_pesan_rgb))
Image.fromarray(np.uint8(hasil_pesan)).save("hasil_wisuda_rgb.png")

print("\nPesan bagian bit pertama Merah(R):\n", pesan_bagian_pertama_r)
print("\nPesan bagian bit kedua Merah(R):\n", pesan_bagian_kedua_r)
print("\nHasil gabungan matriks pesan Merah(R):\n", hasil_gabungan_r)

print("\nPesan bagian bit pertama Hijau(G):\n", pesan_bagian_pertama_g)
print("\nPesan bagian bit kedua Hijau(G):\n", pesan_bagian_kedua_g)
print("\nHasil gabungan matriks pesan Hijau(G):\n", hasil_gabungan_g)

print("\nPesan bagian bit pertama Biru(B):\n", pesan_bagian_pertama_b)
print("\nPesan bagian bit kedua Biru(B):\n", pesan_bagian_kedua_b)
print("\nHasil gabungan matriks pesan Biru(B):\n", hasil_gabungan_b)

##Hasil Pesan##
# Buat subplot untuk menampilkan keempat gambar
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
```

```
# Tampilkan gambar-gambar pada subplot
axs[0, 0].imshow(matriks_gabungan_r, cmap='gray')
axs[0, 0].set_title('Pesan yang diekstrak R')
axs[0, 0].axis('off')

axs[0, 1].imshow(matriks_gabungan_g, cmap='gray')
axs[0, 1].set_title('Pesan yang diekstrak G')
axs[0, 1].axis('off')

axs[1, 0].imshow(matriks_gabungan_b, cmap='gray')
axs[1, 0].set_title('Pesan yang diekstrak B')
axs[1, 0].axis('off')

axs[1, 1].imshow(hasil_pesan_rgb)
axs[1, 1].set_title('Pesan yang diekstrak RGB')
axs[1, 1].axis('off')

# Tampilkan subplot
plt.show()

##Gambar Lengkap RGB
```

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# Tampilkan gambar-gambar pada subplot
axs[0, 0].imshow(matriks_cover)
axs[0, 0].set_title('Cover RGB')
axs[0, 0].axis('off')

axs[0, 1].imshow(matriks_pesan)
axs[0, 1].set_title('Pesan yang disisipkan RGB')
axs[0, 1].axis('off')

axs[1, 0].imshow(stego)
axs[1, 0].set_title('Hasil Steganografi RGB')
axs[1, 0].axis('off')

axs[1, 1].imshow(hasil_pesan_rgb)
axs[1, 1].set_title('Pesan yang diekstrak RGB')
axs[1, 1].axis('off')

# Tampilkan subplot
plt.show()
```

LAMPIRAN 3

KETERANGAN HASIL CEK PLAGIASI

YAYASAN NURUL JADID PAITON
FAKULTAS SOSIAL DAN HUMANIORA
UNIVERSITAS NURUL JADID
PROBOLINGGO JAWA TIMUR

PP. Nurul Jadid
Karanganyar Paiton
Probolinggo 67291
t: 08803077077
soshum@unuja.ac.id

KETERANGAN HASIL CHECK PLAGIASI

Yang bertanda tangan di bawah ini, tim check plagiasi Fakultas Sosial dan Humaniora menerangkan dengan sebenarnya, bahwa telah dilakukan check plagiasi dengan persentase 24% (Exclude Quotes dan Exclude Bibliogrphy) pada tugas akhir/skripsi mahasiswa berikut

Nama : SITI SU'AIBAH

NIM : 2042200026

Judul Skripsi : KODE LINIER UNTUK STEGANOGRAFI CITRA

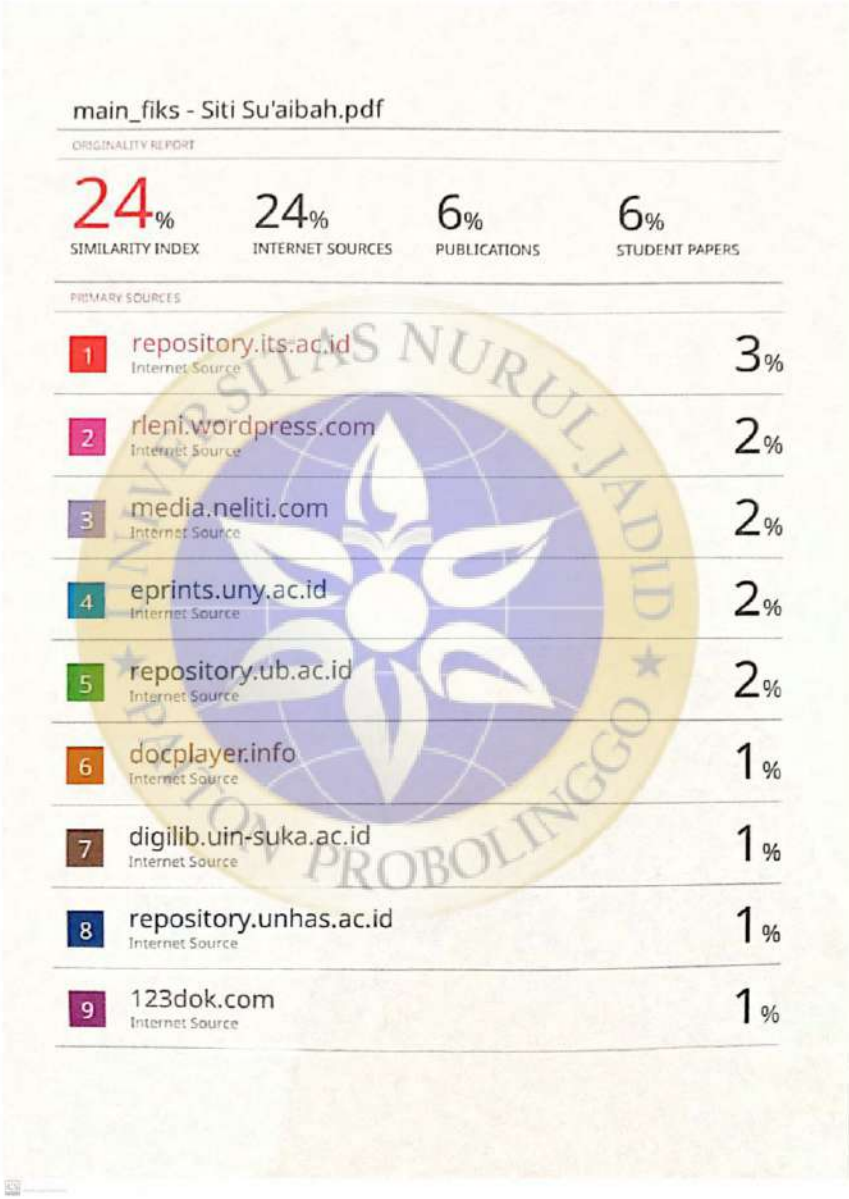
Demikian keterangan ini dibuat dengan sebenarnya dan untuk dijadikan persyaratan kelayakan mengikuti sidang tugas akhir/skripsi.

Paiton, 28 Juli 2024
Ketua Tim,

RM. FARUQ, S.H.I

LAMPIRAN 4

HASIL CEK PLAGIASI



main_fiks - Siti Su'aibah.pdf

ORIGINALITY REPORT

24%	24%	6%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.its.ac.id Internet Source	3%
2	rleni.wordpress.com Internet Source	2%
3	media.neliti.com Internet Source	2%
4	eprints.uny.ac.id Internet Source	2%
5	repository.ub.ac.id Internet Source	2%
6	docplayer.info Internet Source	1%
7	digilib.uin-suka.ac.id Internet Source	1%
8	repository.unhas.ac.id Internet Source	1%
9	123dok.com Internet Source	1%

RIWAYAT HIDUP

SITI SU'AIBAH berdomisili di Desa Randumerak, Kecamatan Paiton, Kabupaten Probolinggo. Penulis lahir di Probolinggo pada tanggal 16 Juni 2002, Anak kedua dari dua bersaudara psangan Bapak Syamsuddin dan Ibu Sulastri. Pendidikan dasarnya ditempuh di MI Azzainiyah III dan selesai pada tahun 2014. Pendidikan menengah pertamanya di SMP Nurul Jadid (SMPNJ), selesai tahun 2017. Pendidikan menengah atasnya ditempuh di SMA Nurul Jadid (SMANJ), selesai tahun 2020. Penulis menempuh sarjana Pendidikan Matematika di Fakultas Sosial dan Humaniora Universitas Nurul Jadid Paiton Probolinggo.

Seiring dengan berbagai aktivitas yang penulis tekuni selama perkuliahan, penulis juga dapat menyelesaikan tugas akhir atau skripsi dengan judul "Kode Linear untuk Steganografi Citra RGB" sebagai salah satu syarat untuk menyelesaikan studi pendidikan jenjang program Strata Satu (S1) Pendidikan Matematika.