LAPORAN PRAKTIKUM 11 IMAGE COMPRESSION

Nama Kelompok : Lukman Hakim (2005013)
Dea Faradisa (2005007)
Siti Putri Rohayati (2005023)
Sri Wahyuni (2005025)

Kelas : D4RPL3

Deskripsi :

Kompresi data adalah bagian yang sangat penting dari dunia digital, di mana kita memiliki banyak file dengan ukuran besar. Banyak faktor dari hal tersebut diantaranya smartphone memiliki kualitas kamera yang lebih baik dan gambar yang diambil darinya juga membutuhkan lebih banyak penyimpanan, dengan kombinasi piksel yang lebih kompleks, lebih banyak penyimpanan yang dibutuhkan. Kompresi dengan format data tertentu seperti JPEG dan PNG, memungkinkan sebagian data hilang. Disisi Lain informasi detail dari gambar dibutuhkan untuk memproses gambar tersebut. Maka kita perlu menggunakan metode konversi lossless yang artinya tidak boleh ada kehilangan data.

A. RLE Sederhana untuk data

a. Encoding

```
def encode_message(message):
    encoded_string = ""
    i = 0
    while (i <= len(message)-1):
        count = 1
        ch = message[i]
        j = i
        while (j < len(message)-1):
            if (message[j] == message[j + 1]):
                count = count + 1
                j = j + 1
            else:
                break
        encoded_string = encoded_string + str(count) + ch
        i = j + 1
    return encoded_string
```

Kode di atas adalah fungsi Python yang disebut encode_message yang mengambil satu parameter, yaitu pesan yang akan dikodekan. Fungsi ini mencoba mengenkripsi pesan menggunakan algoritma Run-Length Encoding (RLE), yang memampatkan data dengan menghitung jumlah karakter berurutan dalam pesan dan menyimpannya sebagai nilai numerik diikuti dengan karakter.

Pertama, fungsi mendeklarasikan variabel encoded_string yang menyimpan hasil encoding. Fungsi kemudian mengulangi setiap karakter dalam pesan menggunakan while loop hingga mencapai karakter terakhir. Variabel i digunakan untuk menunjukkan indeks pesan saat ini. Selama iterasi, variabel count diinisialisasi ke 1 dan variabel ch mewakili karakter saat ini. Variabel j diinisialisasi dengan nilai

i, fungsi kemudian mengulangi pesan lagi dan memasukkan while loop di dalam while loop pertama untuk menghitung jumlah karakter yang sama secara berurutan. Jika karakter saat ini sama dengan karakter berikutnya, variabel kuantitas bertambah 1 dan variabel j bertambah 1. Jika tidak, perulangan while lainnya keluar dan satu karakter dihitung bersama dengan jumlah kejadian. ditambahkan ke variabel encoded_string. Variabel i diperbarui dengan nilai j+1 untuk melanjutkan penghitungan karakter berikutnya.

Setelah loop selesai, pesan terenkripsi dikembalikan sebagai string angka dan string menggunakan operator rangkaian + rantai. Fungsi kemudian mengembalikan variabel encoded_string sebagai hasil pengkodean akhir.

b. Decoding

```
def decode_message(our_message):
        decoded_message = ""
        i=0
        j=0
        while (i <= len(our_message)-1):
            run_count = int(our_message[i])
            run_word = our_message[i + 1]
            for j in range(run_count):
                decoded_message = decoded_message+run_word
                j = j + 1
            i = i + 2
        return decoded_message
```

Kode di atas adalah fungsi Python yang disebut decode_message yang mengambil satu parameter, pesan terenkripsi (our_message). Fungsi ini mencoba mendekode pesan yang sebelumnya dikodekan menggunakan algoritma Run-Length Encoding (RLE) dalam fungsi encode_message.

Pertama, fungsi mendeklarasikan variabel 'decoded_message' yang menyimpan hasil dekode. Fungsi kemudian beralih melalui setiap karakter dalam pesan menggunakan loop "while" hingga mencapai karakter terakhir. Variabel 'i' digunakan untuk menunjukkan indeks pesan saat ini. Selama iterasi, variabel "run_count" diinisialisasi dengan karakter numerik pada indeks pesan "i", yang kemudian diubah menjadi tipe data integer menggunakan fungsi int(). Variabel "run_word" diinisialisasi oleh karakter "i + 1" dari indeks pesan, yang muncul setelah karakter numerik sebelumnya. Kemudian fungsi mengulangi jumlah karakter dalam variabel "run_count" menggunakan for loop untuk menambahkan karakter yang didekodekan "run_count" kali dalam variabel "decoded_message". Variabel 'j' diinisialisasi ke '0' pada awal loop dan bertambah pada setiap iterasi untuk melacak jumlah karakter yang didekodekan.

Saat perulangan berakhir, variabel "i" ditambah dengan "2" untuk melanjutkan penghitungan karakter berikutnya (karena setiap karakter terdiri dari dua karakter: tanda angka yang menunjukkan jumlah kemunculan dan karakter sebenarnya). Fungsi kemudian mengembalikan variabel "decoded_message" sebagai hasil decoding akhir.

Dengan demikian, fungsi decode_message dapat digunakan untuk mendekode pesan yang telah dienkripsi menggunakan algoritma Run-Length Encoding (RLE) dalam fungsi encode_message.

c. Display

```
def display():
    our_message = "AuuBBBCCCCCCccccccCCCCCCCCCA"
    encoded_message=encode_message(our_message)
    decoded_message=decode_message(encoded_message)
    print("Original string: ["+our_message+"]\nEncoded string:["+encoded_message+"]\nDecoded string: ["+decoded_message+"]\n")
display()
```

Kode di atas adalah fungsi python yang disebut display yang digunakan untuk menampilkan hasil encoding dan decoding pesan tertentu menggunakan fungsi encode_message dan decode_message.

Dalam kode ini, pesan asli yang akan dienkripsi adalah "AuuBBBCCCCCCccccCCCCCCCCCA". Variabel our_message digunakan sebagai parameter untuk memanggil fungsi encode_message, yang membuat pesan terenkode yang disimpan dalam variabel encoded_message.

Kemudian variabel encoded_message digunakan sebagai parameter untuk memanggil fungsi decode_message, yang menghasilkan pesan yang dikodekan yang disimpan dalam variabel decoded_message.

Terakhir, fungsi cetak digunakan untuk menampilkan pesan asli, pesan terenkripsi, dan pesan terdekripsi dalam format yang sesuai.

B. RLE Gambar Sederhana (Grayscale image)

a. Import library

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
```

Kode ini digunakan untuk pemrosesan gambar dengan OpenCV, perpustakaan pemrosesan gambar dan video. Kita dapat menggunakan modul numpy untuk memanipulasi array dan matriks gambar, modul cv2 untuk membaca dan menulis gambar, dan modul matplotlib.pyplot untuk menampilkan gambar yang diproses dalam bentuk grafik.

Modul sistem operasi digunakan untuk memproses file gambar. Dalam pemrosesan gambar, biasanya kita menggunakan beberapa operasi dasar seperti: Cara mengubah ukuran gambar, memotong gambar, meningkatkan kontras atau kecerahan gambar, dll.

b. Fungsi Basic

```
def show(img, figsize=(10, 10), title="Image"):
    figure=plt.figure(figsize=figsize)

    plt.imshow(img)
    plt.show()

def get_size(filename="car.png"):
    stat = os.stat(filename)
    size=stat.st_size
    return size
```

Kode di atas berisi dua fungsi Python, yaitu "show" dan "get_size". Fungsi Show digunakan untuk menampilkan gambar dengan ukuran tertentu di jendela tampilan. Fungsi ini memiliki tiga parameter yaitu "img" yang berisi gambar yang akan ditampilkan, "figsize" yang merupakan ukuran jendela tampilan dan "title" yang merupakan judul gambar yang akan ditampilkan.

Fungsi ini menggunakan modul matplotlib.pyplot untuk menampilkan gambar di jendela penampil pada ukuran yang ditentukan oleh pengaturan figsize. Kemudian gambar ditampilkan menggunakan fungsi imshow dan judul gambar ditambahkan menggunakan fungsi judul. Fungsi Show menampilkan gambar di jendela tampilan.

Fungsi get_size digunakan untuk mendapatkan file gambar dalam byte. Fungsi ini membutuhkan satu parameter, 'nama file', yang merupakan nama file gambar yang akan dikonversi.

Fungsi ini menggunakan modul "os" untuk mendapatkan informasi gambar yang diinginkan. Informasi ini diambil oleh fungsi os.stat, yang mengembalikan objek stat_result yang berisi informasi tentang file tersebut. Ukuran file gambar kemudian diambil dari objek stat_result menggunakan atribut st_size. Ukuran file ini adalah nilai kembalian dari fungsi get_size.

c. Encoding

```
def RLE_encoding(img, bits=8, binary=True, view=True):
    """
    img: Grayscale img.
    bits: run length makasimum adalah 2^bits
    """
    if binary:
        ret,img = cv2.threshold(img,127,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

        if view:
            show(img)

        encoded = []
        shape=img.shape
        count = 0
        prev = None
        fimg = img.flatten()
        th=127
        for pixel in fimg:
            if binary:
                if pixel<th:
                    pixel=0
                else:
                    pixel=1
            if prev==None:
                prev = pixel
                count+=1
```

```
            else:
                if prev!=pixel:
                    encoded.append((count, prev))
                    prev=pixel
                    count=1
                else:
                    if count<(2**bits)-1:
                        count+=1
                    else:
                        encoded.append((count, prev))
                        prev=pixel
                        count=1
        encoded.append((count, prev))

        return [np.array(encoded), img]

## ujicoba
fpath="car.png"
img = cv2.imread(fpath, 0)
shape=img.shape
[encoded, ori] = RLE_encoding(img, bits=8)
print(encoded)
```

Fungsi 'RLE_encoding' adalah fungsi yang melakukan pengkodean Run-Length Encoding (RLE) pada gambar skala abu-abu. RLE adalah teknik kompresi data yang menggantikan record dengan record dengan jumlah dan nilai yang sama. Fungsi ini menerima tiga parameter, yaitu "img" yang berisi gambar grayscale untuk dikodekan, "bits" yang menentukan panjang roll maksimum dalam bit dan "binary" yang menentukan apakah gambar yang diterima harus dibiarkan biner atau tidak.

Fungsi ini menggunakan modul cv2 untuk mengonversi gambar skala abu-abu menjadi gambar biner jika parameter binernya benar. Jika parameter "view" benar, maka fungsi "show" digunakan untuk menampilkan gambar di jendela tampilan. Pemrosesan RLE dilakukan pada gambar yang telah diubah menjadi array satu dimensi menggunakan flatten(). Fungsi RLE_encoding mengulang setiap piksel gambar dan memeriksa piksel sebelumnya. Jika piksel saat ini sama dengan piksel sebelumnya, hitungannya bertambah satu. Jika piksel saat ini berbeda dari piksel sebelumnya, hitungan dan nilai piksel sebelumnya dimasukkan ke dalam daftar yang disandikan. Saat hitungan telah mencapai maksimum, hitungan dan nilai piksel sebelumnya juga dimasukkan ke dalam daftar yang disandikan. Proses ini berlanjut hingga selesai melalui semua piksel dalam gambar.

Nilai hasil dari fungsi ini adalah daftar dengan dua elemen, 'dikodekan' dan 'img'. "dikodekan" berisi data RLE dari gambar yang dikodekan, sedangkan "img" adalah gambar yang dikonversi ke biner jika parameter "biner" benar. Dalam kode di atas, nilai "dikodekan" adalah keluaran.

d. Decoding

```
def RLE_decode(encoded, shape):
    decoded=[]
    for rl in encoded:
        r,p = rl[0], rl[1]
        decoded.extend([p]*r)
    dimg = np.array(decoded).reshape(shape)
    return dimg

dimg = RLE_decode(encoded, shape)
show(dimg)
```

Kode di atas adalah fungsi decoding dari algoritma Run-Length Encoding (RLE) untuk gambar. Fungsi pengkodean RLE memampatkan citra dengan mengkodekan piksel yang memiliki nilai yang sama dan berurutan menjadi sepasang nilai, yaitu nilai piksel dan panjang urutan piksel. Fungsi dekoding RLE yang ditampilkan di sini mengambil input dalam bentuk gambar yang dikodekan RLE dan ukuran gambar asli, lalu mendekodekannya dengan merekonstruksi gambar asli dari sepasang nilai panjang dan piksel RLE. . Hasil dekompresi kemudian ditampilkan kembali menggunakan fungsi show().

e. Hitung size kompresi

```
cv2.imwrite("lion.tiff", encoded)
cv2.imwrite("car.png", encoded)
cv2.imwrite("kumbang.bmp", encoded)
```

Tujuan dari tiga baris kode tersebut adalah untuk menyimpan citra hasil encoding dalam tiga format berbeda yaitu TIFF, PNG dan BMP.

Fungsi "cv2.imwrite()" digunakan untuk menyimpan gambar dalam format file tertentu. Parameter pertama berisi nama file yang akan disimpan dan ekstensinya, parameter kedua berisi data gambar yang akan disimpan. Dalam hal ini, data citra yang akan disimpan adalah variabel “encoded” yang berisi hasil encoding dari citra tersebut.

Dengan menyimpan hasil coding dalam format file yang berbeda, kita dapat membandingkan ukuran file hasil coding dalam format yang berbeda dan melihat mana yang paling efisien untuk digunakan.

```
files = ["car.png", "lion.tiff", "kumbang.bmp",fpath]
for f in files:
    print(f"File: {f} => Size: {get_size(f)} Bytes")
```

Kode ini menampilkan ukuran file dari beberapa file gambar. File gambar adalah file terenkripsi dari gambar asli menggunakan teknik Run Length Encoding (RLE) yang diterapkan sebelumnya.

Pertama, inisialisasi daftar yang disebut "files" yang berisi beberapa nama file untuk diuji, yaitu "encoded.png", "encoded.tif", "encoded.bmp" dan "liona.jpg" (nama file gambar asli yang dibuat untuk pengkodean digunakan).

Kemudian ulangi setiap file dalam daftar file. Dengan setiap iterasi, fungsi "get_size(f)" dipanggil, yang menggunakan nama file sebagai parameter. Fungsi get_size() menggunakan os.stat() untuk mengembalikan ukuran file dalam byte.

Setelah loop berjalan, program akan menampilkan ukuran setiap file sampel dalam bentuk pesan di layar.

C. RLE Gambar RGB

```
bgr = cv2.imread("gunung.jpg", 1)
show(bgr)

show(cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB))

b, g, r = bgr[:, :, 0], bgr[:, :, 1], bgr[:, :, 2]


[be, imgbe] = RLE_encoding(b, binary=False)
[ge, imgge] = RLE_encoding(g, binary=False)
[re, imgre] = RLE_encoding(r, binary=False)

be.shape, ge.shape, re.shape

np.savez("Data/rgbe.npz", [be, ge, re], dtype=object)
get_size("Data/rgbe.npz")

np.savez("Data/be.npz", be)
np.savez("Data/ge.npz", ge)
np.savez("Data/re.npz", re)
cv2.imwrite("Data/be.tif", be)
cv2.imwrite("Data/ge.tif", ge)
cv2.imwrite("Data/re.tif", re)
files = "bgr"
snp = 0
stif = 0
```

```
for f in files:
    ft="Data/"+f+"e"+".npz"
    snp+=get_size(ft)
    ft="Data/"+f+"e"+".tif"
    stif+=get_size(ft)

print(f"Original size: {get_size('gunung.jpg')/1024} Byte, TIFF: {stif/1024} Byte, NPZ: {snp/1024} Byte")
```

Kode di atas terlebih dahulu membaca gambar berwarna dalam format BGR, yang kemudian ditampilkan. Selanjutnya, gambar diubah dari BGR ke RGB menggunakan fungsi "cv2.cvtColor()" untuk menampilkan gambar dengan warna yang benar. Kemudian citra BGR dibagi menjadi tiga kanal warna B, G dan R, masing-masing kanal diubah menjadi citra biner menggunakan fungsi 'RLE_encoding()' dengan parameter 'binary=False' dan hasilnya disimpan di dalam 'be' . variabel. , 'ge' dan 're', masing-masing untuk saluran B, G dan R. Setelah itu, tiga file "be.npz", "ge.npz" dan "re.npz" disimpan dengan array 'be'. 'ge' dan 're' dengan fungsi 'np.savez()'. Selain itu, tiga file "be.tif", "ge.tif" dan "re.tif" berisi gambar biner "be", "ge" dan "re" dalam format TIFF dengan fungsi "cv2.imwrite". ( )'. Terakhir, ukuran file gambar simpanze.jpg asli, ukuran total ketiga file '*.npz' dan ukuran total ketiga file '*.tif' dihitung.