**UAS PENGOLAHAN CITRA**

Dosen pengampu: Oscar Hadikaryana, Mt



Anggota:

Rahman Fauza 20201310034

Muhammad Ridho 20221310028

Arya Abdul Mughni 20221310006

Farid Zia Ul Haq

**PRODI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER DAN SISTEM INFORMASI**

**UNIVERSITAS KEBANGSAAN REPUBLIK INDONESIA**

**2024**

**PERCOBAAN 8 kompersi citra**

**Code**

import os

from PIL import Image

import cv2  # Untuk mengambil gambar dari kamera

def get\_size\_format(b, factor=1024, suffix="B"):

    """

    Scale bytes to its proper byte format

    e.g:

    1253656 => '1.20MB'

    1253656678 => '1.17GB'

    """

    for unit in ["", "K", "M", "G", "T", "P", "E", "Z"]:

        if b < factor:

            return f"{b:.2f}{unit}{suffix}"

        b /= factor

    return f"{b:.2f}Y{suffix}"

def compress\_img(image\_name, new\_size\_ratio=0.9, quality=90, width=None, height=None, to\_jpg=True):

    # Load the image to memory

    img = Image.open(image\_name)

    # Print the original image shape

    print("[\*] Image shape:", img.size)

    # Get the original image size in bytes

    image\_size = os.path.getsize(image\_name)

    # Print the size before compression/resizing

    print("[\*] Size before compression:", get\_size\_format(image\_size))

    if new\_size\_ratio < 1.0:

        # If resizing ratio is below 1.0, then multiply width & height with this ratio to reduce image size

        img = img.resize((int(img.size[0] \* new\_size\_ratio), int(img.size[1] \* new\_size\_ratio)), Image.Resampling.LANCZOS)

        # Print new image shape

        print("[+] New Image shape:", img.size)

    elif width and height:

        # If width and height are set, resize with them instead

        img = img.resize((width, height), Image.Resampling.LANCZOS)

        # Print new image shape

        print("[+] New Image shape:", img.size)

    # Split the filename and extension

    filename, ext = os.path.splitext(image\_name)

    # Make new filename appending \_compressed to the original file name

    if to\_jpg:

        # Change the extension to JPEG

        new\_filename = f"{filename}\_compressed.jpg"

    else:

        # Retain the same extension of the original image

        new\_filename = f"{filename}\_compressed{ext}"

    try:

        # Save the image with the corresponding quality and optimize set to True

        img.save(new\_filename, quality=quality, optimize=True)

    except OSError:

        # Convert the image to RGB mode first

        img = img.convert("RGB")

        # Save the image with the corresponding quality and optimize set to True

        img.save(new\_filename, quality=quality, optimize=True)

    print("[+] New file saved:", new\_filename)

    # Get the new image size in bytes

    new\_image\_size = os.path.getsize(new\_filename)

    # Print the new size in a good format

    print("[+] Size after compression:", get\_size\_format(new\_image\_size))

    # Calculate the saving bytes

    saving\_diff = new\_image\_size - image\_size

    # Print the saving percentage

    print(f"[+] Image size change: {saving\_diff / image\_size \* 100:.2f}% of the original image size.")

if \_\_name\_\_ == "\_\_main\_\_":

    # Capture an image from the camera

    camera = cv2.VideoCapture(0)

    if not camera.isOpened():

        print("Error: Unable to access the camera.")

        exit()

    print("Press 'Space' to capture the image or 'ESC' to exit.")

    while True:

        ret, frame = camera.read()

        if not ret:

            print("Error: Failed to capture the frame.")

            break

        # Display the camera feed

        cv2.imshow("Camera", frame)

        # Wait for key press

        key = cv2.waitKey(1) & 0xFF

        if key == 27:  # ESC key to exit

            print("Exiting...")

            break

        elif key == 32:  # Space key to capture

            image\_path = "captured\_image.jpg"

            cv2.imwrite(image\_path, frame)

            print(f"Image captured and saved to {image\_path}")

            break

    camera.release()

    cv2.destroyAllWindows()

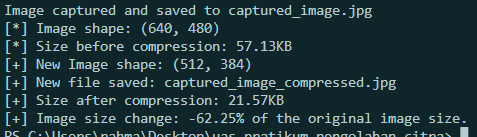
    # Compress the captured image

    compress\_img(image\_path, new\_size\_ratio=0.8, quality=85, to\_jpg=True)

kode kami menggunakan Pustaka open sv untuk mengambil gambar secara langsung dari camera. Hasil dari kompersi citra bisa dilihat pada gambar dibawah



Gambar asli Hasil kompersi



**HASIL ANALISIS**

Dari hasil yang kami dapatkan kompersi img cukup memperlihatkan bahwa penuruunan size dan dimensi terlihat cukup baik dengan penurunan size setelah konversi sebesar 21.57kb dari yang asalnya 57.13kb. serta dimensi yang di perkecil dari 640.480 ke 512.384.

**PERCOBAAN 9 Pengukuran Kualitas Citra**

**CODE**

import cv2

import numpy as np

from skimage.metrics import structural\_similarity as ssim

import matplotlib.pyplot as plt

def calculate\_psnr(original, compressed):

    if original is None or compressed is None:

        raise ValueError("Salah satu gambar tidak valid. Pastikan file tersedia dan dapat dibaca.")

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

    if mse == 0:

        return 100

    max\_pixel = 255.0

    psnr = 20 \* np.log10(max\_pixel / np.sqrt(mse))

    return psnr

# Membaca gambar

original = cv2.imread(r'C:\Users\rahma\Desktop\uas pratikum pengolahan citra\captured\_image.jpg', cv2.IMREAD\_GRAYSCALE)

compressed = cv2.imread(r'C:\Users\rahma\Desktop\uas pratikum pengolahan citra\captured\_image\_compressed.jpg', cv2.IMREAD\_GRAYSCALE)

# Validasi gambar

if original is not None and compressed is not None:

    # Resize the 'compressed' image to match the dimensions of the 'original' image

    compressed\_resized = cv2.resize(compressed, (original.shape[1], original.shape[0]))

    # Perhitungan PSNR dan SSIM

    psnr\_value = calculate\_psnr(original, compressed\_resized)

    # Calculate SSIM using the resized image

    ssim\_value = ssim(original, compressed\_resized)

    # Menampilkan hasil PSNR dan SSIM

    print(f"PSNR: {psnr\_value:.2f} dB")

    print(f"SSIM: {ssim\_value:.2f}")

    # Menampilkan gambar menggunakan matplotlib

    plt.figure(figsize=(10, 5))

    # Gambar asli

    plt.subplot(1, 2, 1)

    plt.imshow(original, cmap='gray')

    plt.title('Gambar Asli')

    plt.axis('off')

    # Gambar terkompresi

    plt.subplot(1, 2, 2)

    plt.imshow(compressed\_resized, cmap='gray')

    plt.title('Gambar Terkompresi')

    plt.axis('off')

    # Menampilkan plot

    plt.tight\_layout()

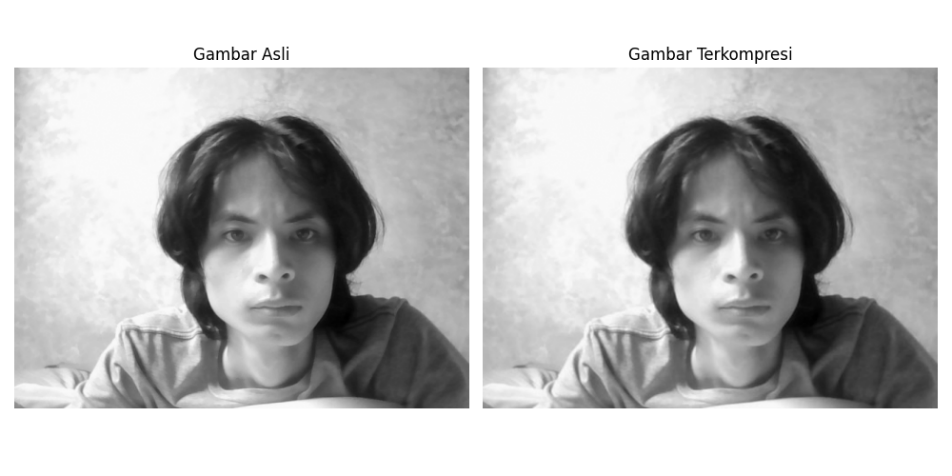
    plt.show()

else:

    print("Error: Tidak dapat memuat salah satu atau kedua gambar.")

path gambar menggunakan gamabar yang asli dan telah di kompres ini dibuat untuk memudahkan penggunaan code. Untuk hasil dan analisis bisa dilihat di bawah ini.

**HASIL DAN ANALISIS**





Hasil kompresi menunjukkan kualitas citra yang cukup baik dengan perbedaan visual minimal.

**PERCOBAAN 10 Deteksi dan Analisis Kontur**

**CODE**

import cv2

img = cv2.imread('tts1.png', cv2.IMREAD\_GRAYSCALE)

edges = cv2.Canny(img, 100, 200)

contours, \_ = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

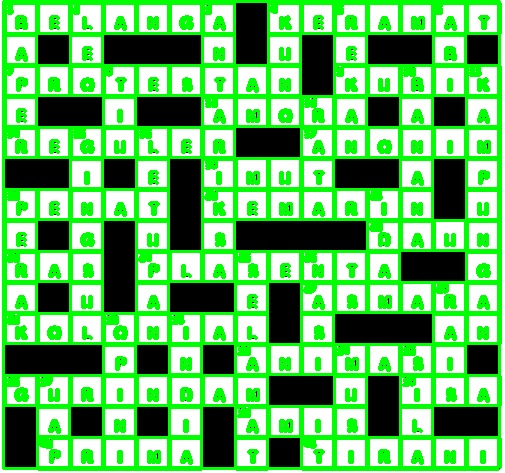
result = cv2.cvtColor(img, cv2.COLOR\_GRAY2BGR)

cv2.drawContours(result, contours, -1, (0, 255, 0), 2)

cv2.imwrite('contours.jpg', result)

kami mengguakan code yang sudah di perbaharui untuk deteksi dan analisis kontur dari percobaan code awal dan yang telah dimodifikasi. Berikut hasil dan Analisa yang di dapatkan.

**HASIL DAN ANALISA**

****

Dari hasil dan Analisa yang kami dapatkan memperlihatkan jumlah kontur yang terdeteksi cukup baik.

Untuk lebih jelasnya pada percobaan 8 9 dan 10 bisa dilihat di video ini: <https://youtu.be/t2VUyU3h7fE>

Source code: <https://github.com/Rahmanfauza/tugas-uas-pratikum-pengolahan-citra.git>