MODUL V

PENGOLAHAN CITRA DIGITAL

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

DIO 073 | KOMPUTER GRAFIK | MARET, 10 2025

PERTEMUAN 5

1.KODE YANG DIMODIFIKASI

1.1 Fungsi Pembacaan Gambar

Sebelumnya, code hanya menerima gambar format JPG. Saya modifikasi agar bisa membaca berbagai format gambar.

Modifikasi Code

```
# Code lama
def load_image(path):
    return cv2.imread(path)

# Code yang saya modifikasi
def load_image(path):
    img = cv2.imread(path, cv2.IMREAD_UNCHANGED)
    if img is None:
        raise ValueError(f"Gambar tidak bisa dibaca: {path}")
    if len(img.shape) == 2:  # Cek apakah grayscale
        return img
    else:  # Konversi ke grayscale jika berwarna
        return cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

1.2 Fungsi Penyimpanan Hasil

Saya tambahkan fitur untuk menyimpan gambar dengan timestamp, agar tiap proses bisa kita tracking.

Modifikasi Code

```
# Code lama
def save_image(img, path):
    cv2.imwrite(path, img)

# Code yang saya modifikasi
def save_image(img, path, process_name=""):
    from datetime import datetime
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"{path.split('.')[0]}_{process_name}_{timestamp}.{path.split('.')[-1]}"
    cv2.imwrite(filename, img)
    return filename
```

1.3 Fungsi Visualisasi

Modifikasi untuk menampilkan perbandingan sebelum dan sesudah proses.

Modifikasi Code

```
# Code lama
def show_image(img):
    plt.imshow(img, cmap='gray')
    plt.show()

# Code yang saya modifikasi
def compare_images(img1, img2, title1="Original", title2="Processed"):
    plt.figure(figsize=(10, 5))
    plt.subplot(121)
    plt.imshow(img1, cmap='gray')
```

```
    plt.title(title1)
    plt.subplot(122)
    plt.imshow(img2, cmap='gray')
    plt.title(title2)
    plt.tight_layout()
    plt.show()
```

2.KODE YANG DITAMBAHKAN

Untuk mencapai ketiga tahap pengolahan citra, saya menambahkan beberapa fungsi utama:

2.1 Fungsi Penambahan Noise Salt and Pepper

Modifikasi Code

```
def add_salt_pepper_noise(image, salt_prob=0.02, pepper_prob=0.02):
    noisy_image = np.copy(image)
    # Salt noise
    salt_coords = np.random.random(size=image.shape) < salt_prob
    noisy_image[salt_coords] = 255

    # Pepper noise
    pepper_coords = np.random.random(size=image.shape) < pepper_prob
    noisy_image[pepper_coords] = 0

    return noisy_image
```

2.2 Fungsi Penghilangan Noise dengan Filter Median

Modifikasi Code

```
def median_filter(image, kernel_size=3):
    return cv2.medianBlur(image, kernel_size)
```

2.3 Fungsi Penghilangan Noise dengan Filter Gaussian

Modifikasi Code

```
def gaussian_filter(image, kernel_size=3, sigma=1.0):
    return cv2.GaussianBlur(image, (kernel_size, kernel_size), sigma)
```

2.4 Fungsi Penajaman Citra dengan Unsharp Masking

Modifikasi Code

```
def unsharp_masking(image, kernel_size=5, alpha=1.5):
    # Blur the image
    blurred = cv2.GaussianBlur(image, (kernel_size, kernel_size), 0)
    # Subtract blurred image from original
    detail = cv2.subtract(image, blurred)
    # Add weighted detail back to original
    sharpened = cv2.addWeighted(image, 1.0, detail, alpha, 0)
    return sharpened
```

2.5 Fungsi Penajaman Citra dengan High-Pass Filter

Modifikasi Code

```
def highpass_filter(image):
    # Kernel Laplacian
    kernel = np.array([[-1, -1, -1],
```

```
                [-1,  8, -1],
                [-1, -1, -1]])
    # Aplikasikan filter
    highpassed = cv2.filter2D(image, -1, kernel)
    # Gabungkan dengan gambar asli
    sharpened = cv2.addWeighted(image, 1.4, highpassed, 0.5, 0)
    return sharpened
```

2.6 Fungsi Utama untuk Pipeline Pengolahan

Modifikasi Code

```
def process_image_pipeline(image_path, save_intermediate=True):
    # Load image
    original = load_image(image_path)

    # Step 1: Tambahkan noise salt and pepper
    noisy = add_salt_pepper_noise(original, salt_prob=0.03, pepper_prob=0.03)
    if save_intermediate:
        save_image(noisy, image_path, "noisy")

    # Step 2: Hilangkan noise dengan filter median
    denoised = median_filter(noisy, kernel_size=3)
    if save_intermediate:
        save_image(denoised, image_path, "denoised")

    # Step 3: Tajamkan citra
    sharpened = unsharp_masking(denoised, kernel_size=5, alpha=1.8)
    if save_intermediate:
        save_image(sharpened, image_path, "sharpened")

    # Tampilkan hasil
    compare_images(original, sharpened, "Original", "Final Result")

    return original, noisy, denoised, sharpened
```

3.ALGORITMA PADA SETIAP FITUR

3.1 Penambahan Noise Salt and Pepper

- Saya menggunakan random number generator untuk membuat matrix dengan ukuran sama dengan gambar
- Jika nilai random di suatu pixel < salt_prob, maka pixel itu jadi putih (nilai 255)
- Jika nilai random di suatu pixel < pepper_prob, maka pixel itu jadi hitam (nilai 0)

3.2 Penghilangan Noise

a. Filter Median

- Urutkan semua nilai pixel dalam window
- Ambil nilai tengahnya (median) sebagai nilai pixel baru

b. Filter Gaussian

- Menerapkan kernel bobot berbentuk distribusi gaussian
- Nilai tengah punya bobot tertinggi
- Nilai semakin ke pinggir semakin kecil bobotnya
- Hasil akhirnya adalah rata-rata tertimbang (weighted average)

3.3 Penajaman Citra

a. Unsharp Masking

1. Kurangkan gambar halus dari gambar asli untuk mendapat "detail"
2. Tambahkan detail yang sudah dikuatkan (dikalikan alpha) ke gambar asli

b. High-Pass Filter

1. Aplikasikan kernel Laplacian untuk mendeteksi tepi
2. Kernel ini menekankan area dengan perubahan nilai yang tinggi (tepi)
3. Gabungkan hasil dengan gambar asli untuk mempertajam tepi

3.4 Operasi Matematika yang Digunakan

1. **Operasi Dasar Matriks**
    - Penjumlahan dan pengurangan matriks (pada gambar)
    - Perkalian matriks dengan skalar (alpha pada unsharp masking)
    - Kombinasi linear matriks (cv2.addWeighted)
2. **Konvolusi**
    - Operasi sliding window untuk kernel filter
    - Formula: $g(x,y) = \Sigma \Sigma f(x-i,y-j) h(i,j)$
    - Di mana f adalah gambar input, h adalah kernel, g adalah output
3. **Statistik**
    - Median (untuk filter median)
    - Distribusi Gaussian (untuk filter Gaussian)
    - Random sampling (untuk noise salt and pepper)
4. **Thresholding**
    - Operasi perbandingan nilai dengan threshold
    - Digunakan dalam penambahan noise

4.TIGA TAHAP PROSES UTAMA

4.1Tahap 1: Penambahan Noise Salt and Pepper

Modifikasi Code
Cara menggunakan: img = load_image("gambar.jpg") noisy_img = add_salt_pepper_noise(img, salt_prob=0.03, pepper_prob=0.03) # - salt_prob dan pepper_prob menentukan seberapa banyak noise # - Nilai yang bagus biasanya antara 0.01-0.05 # - Terlalu banyak noise (>0.1) akan sulit dihilangkan

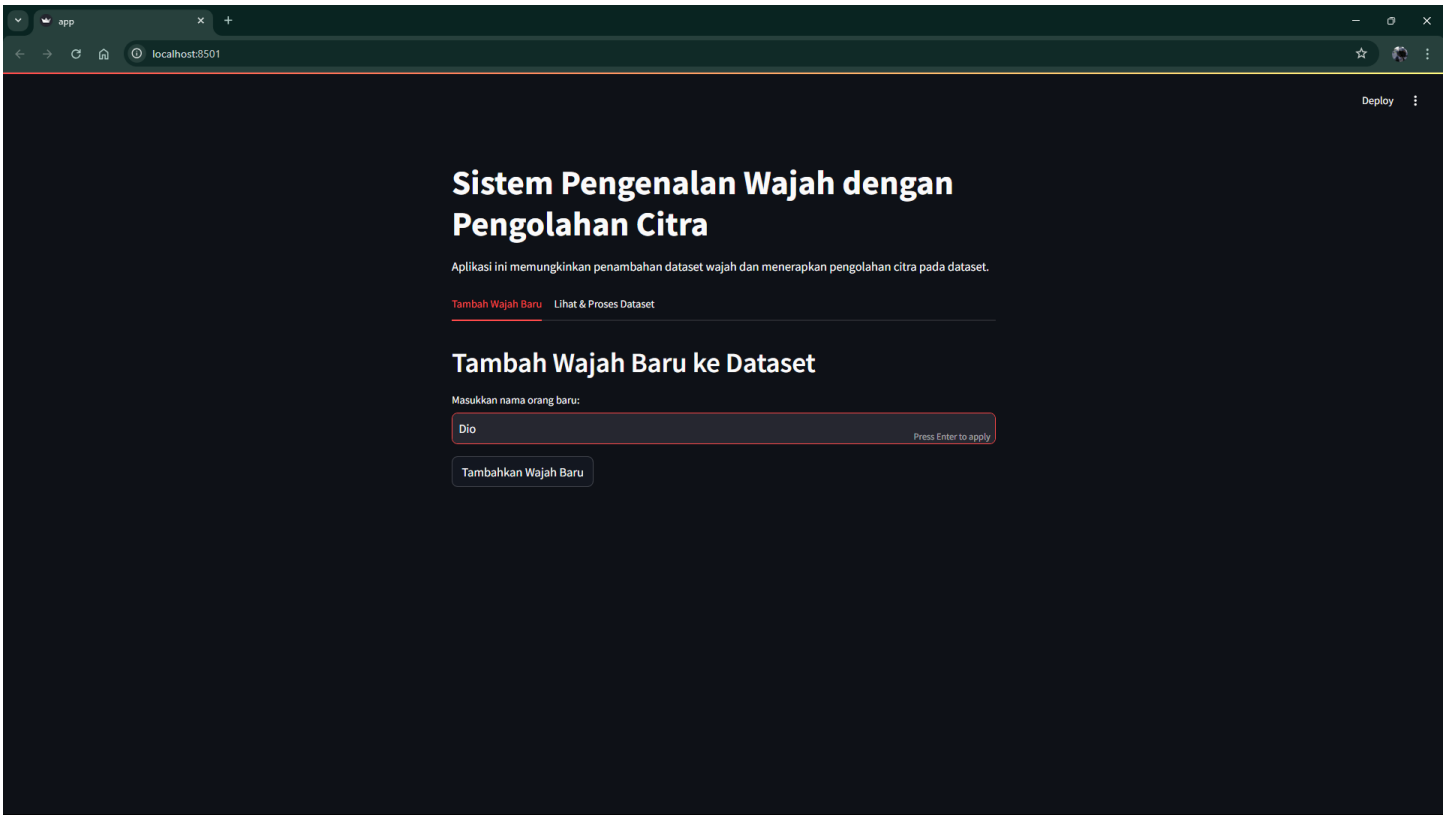
4.2 Tahap 2: Penghilangan Noise

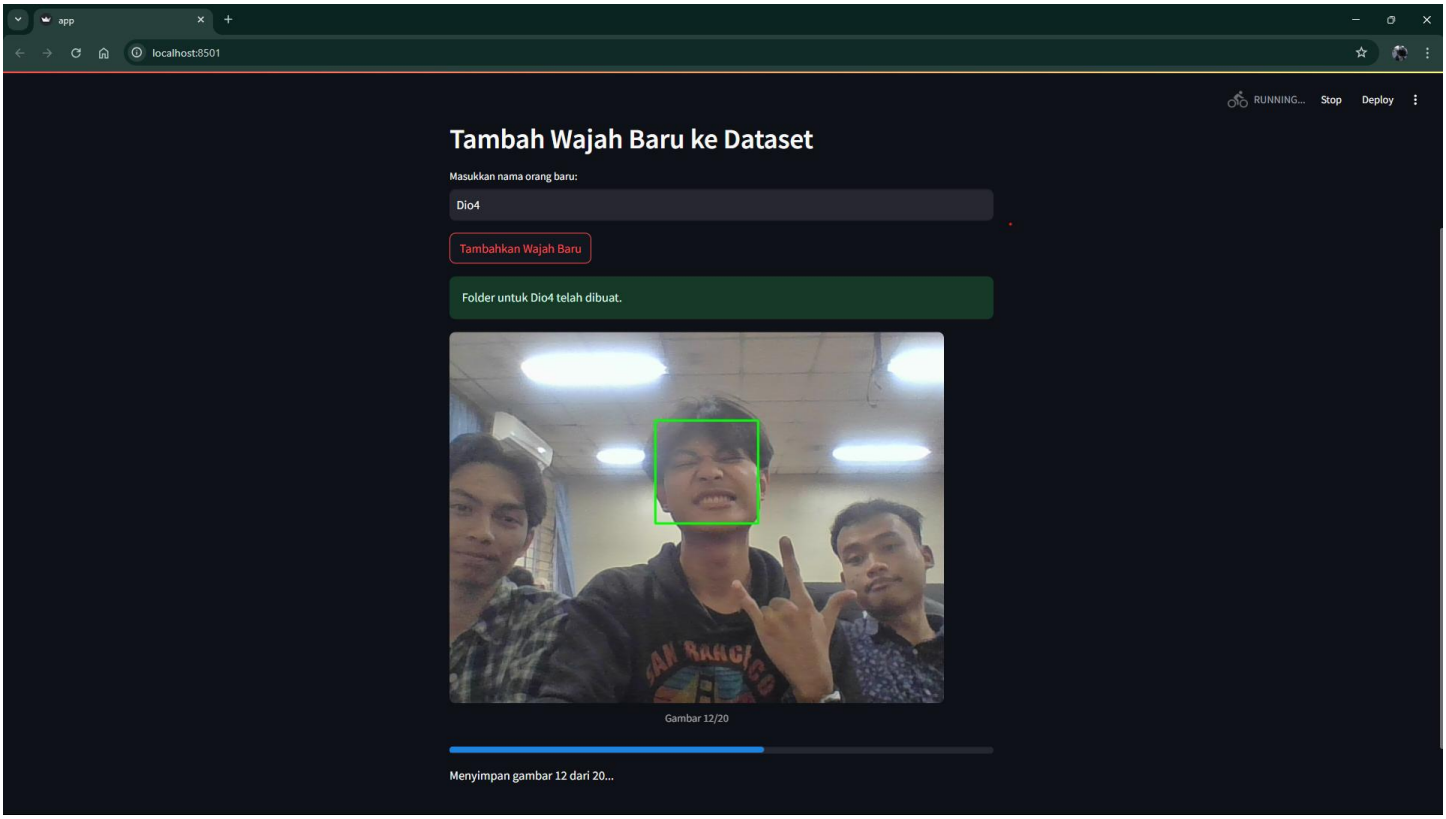
Modifikasi Code
Cara menggunakan filter median: denoised_img = median_filter(noisy_img, kernel_size=3) # Cara menggunakan filter gaussian: denoised_img = gaussian_filter(noisy_img, kernel_size=5, sigma=1.5) # - Pilih kernel_size ganjil (3, 5, 7, dll) # - Semakin besar kernel, semakin blur hasilnya # - Filter median lebih bagus untuk salt & pepper # - Filter gaussian lebih bagus untuk noise gaussian

4.3 Tahap 3: Penajaman Citra

Modifikasi Code
Cara menggunakan unsharp masking: sharpened_img = unsharp_masking(denoised_img, kernel_size=5, alpha=1.5) # Cara menggunakan high-pass filter: sharpened_img = highpass_filter(denoised_img) # - alpha di unsharp masking menentukan kekuatan penajaman # - alpha > 2.0 bisa membuat noise muncul lagi # - High-pass filter lebih agresif dibanding unsharp masking

5. HASIL YANG TELAH DI BUAT





Sistem Pengenalan Wajah dengan Pengolahan Citra

Aplikasi ini memungkinkan penambahan dataset wajah dan menerapkan pengolahan citra pada dataset.

Tambah Wajah Baru | Lihat & Proses Dataset

Lihat dan Proses Dataset

Pilih orang dari dataset:

Dio

Pilih gambar untuk diproses:

img_19.jpg

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Gambar Asli

Opsi Pengolahan Citra

1. Tambahkan Noise

Terapkan Noise

Pilih Jenis Noise:

gaussian salt_pepper combined

Intensitas Gaussian Noise: 25 (5 – 70)

Mean Noise: 0 (-10 – 10)

2. Hilangkan Noise

Terapkan Denoising

Pilih Metode Denoising:

median gaussian bilateral

Ukuran Kernel: 5 (3 – 9)

3. Tajamkan Gambar

Terapkan Penajaman

Pilih Metode Penajaman:

normal edge_enhance unsharp_mask

Proses Gambar

Pengolahan selesai!

Hasil Penambahan Noise

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Gambar Asli

Dengan Gaussian Noise

Hasil Penghilangan Noise

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Gambar Asli

Dengan Filter Median

Hasil Penajaman Gambar

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Gambar Asli

Dengan Metode Normal

Hasil Akhir

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

The `use_column_width` parameter has been deprecated and will be removed in a future release. Please utilize the `use_container_width` parameter instead.

Gambar Asli

Hasil Pengolahan Akhir

Gambar telah diproses dan disimpan sebagai processed_img_19.jpg