LAPORAN TUGAS
PENGOLAHAN CITRA DIGITAL
DOMAIN SPACIAL

OLEH :

Ahmad Afil

F 551 22 050

KELAS B

PROGRAM STUDI S1 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TADULAKO

PALU 2024

I. TEORI DASAR

Domain spasial adalah pendekatan dalam pemrosesan gambar dan pengolahan sinyal di mana operasi dilakukan secara langsung pada piksel gambar atau elemen sinyal berdasarkan lokasi spasial mereka. Pendekatan ini melibatkan manipulasi nilai piksel dan distribusi dalam gambar atau sinyal untuk mencapai tujuan tertentu, seperti peningkatan kualitas gambar, deteksi tepi, atau penyaringan kebisingan. Dalam domain spasial, teknik-teknik seperti operasi filter dan konvolusi digunakan untuk memodifikasi atau mengekstrak informasi dari gambar dengan mempertimbangkan hubungan spasial antar piksel.

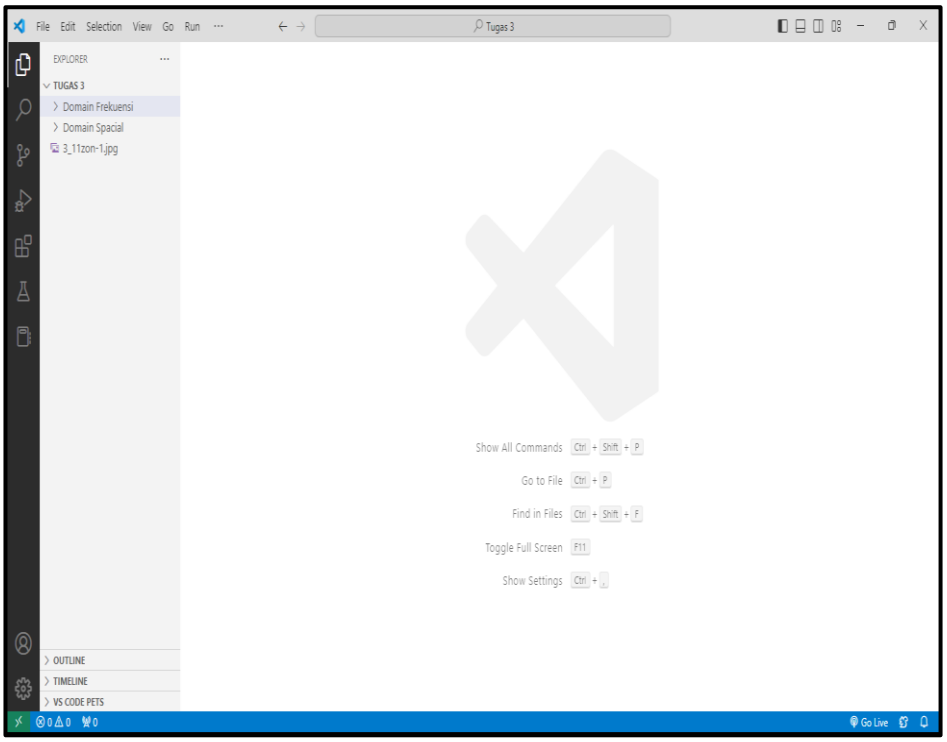
Pada pendekatan domain spasial, hasil manipulasi bergantung pada nilai-nilai piksel di sekitarnya dan cara mereka diolah dengan filter atau operator tertentu. Misalnya, dalam operasi konvolusi, gambar diproses dengan menggunakan sebuah kernel atau filter yang terdiri dari nilai-nilai tertentu, dan hasilnya adalah penjumlahan produk dari nilai piksel gambar dan kernel. Dengan cara ini, kita bisa menerapkan operasi penyaringan, pendeteksian tepi, atau transformasi lainnya.

Salah satu contoh umum penggunaan domain spasial adalah dalam teknik peningkatan gambar, seperti penghalusan (smoothing), penajaman (sharpening), atau deteksi tepi (edge detection). Teknik-teknik seperti filter rata-rata (average filter) dan median filter digunakan untuk mengurangi kebisingan dalam gambar. Pada saat yang sama, teknik lain seperti filter Sobel atau Laplacian digunakan untuk menyoroti tepi dan detail yang tajam dalam gambar. Meskipun sederhana, operasi domain spasial sangat berguna dalam banyak aplikasi, mulai dari fotografi hingga analisis medis dan pengenalan pola.

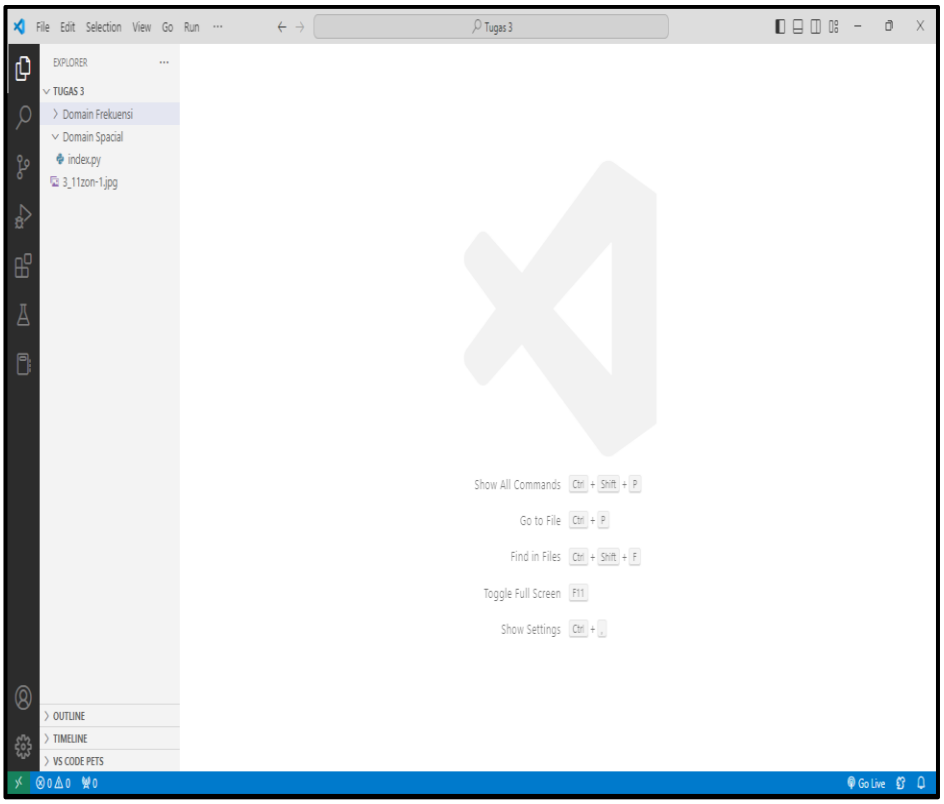
Penting untuk memahami bahwa pendekatan domain spasial berbeda dari domain frekuensi, di mana transformasi dilakukan dalam ranah frekuensi menggunakan teknik seperti Transformasi Fourier. Meskipun domain spasial lebih intuitif karena berurusan langsung dengan piksel gambar, domain frekuensi bisa menawarkan keunggulan dalam analisis frekuensi dan operasi lain yang memerlukan pemrosesan global gambar.

II. LANGKAH KERJA

A. Jalankan aplikasi *VSCode*.



B. Selanjutnya menambahkan file dengan nama ‘index.py’



C. Masukkan Kode Program Berikut.

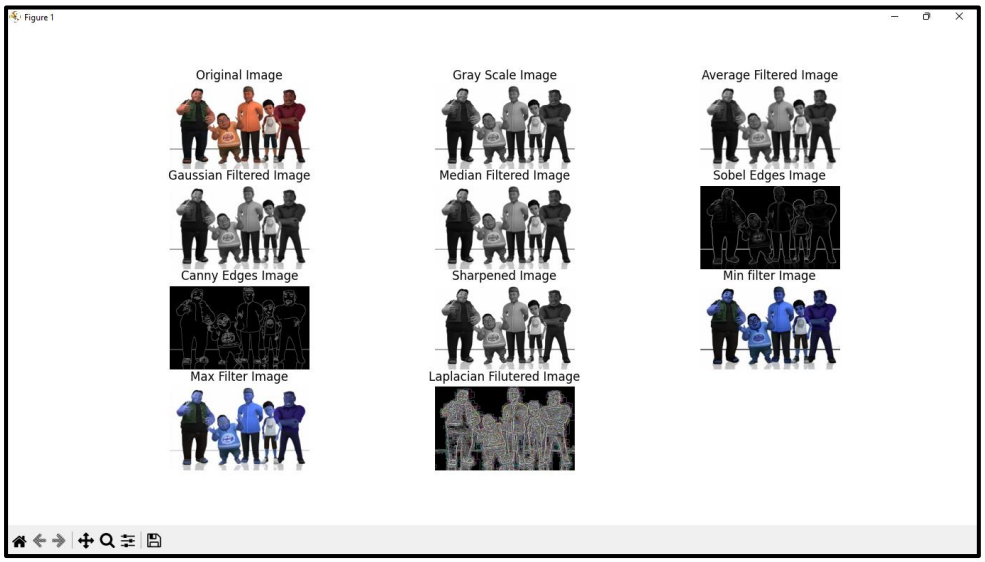
Domain Spacial > index.py > ...

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

# Fungsi untuk mengubah citra menjadi grayscale
def grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Fungsi untuk menerapkan filter rata-rata pada citra
def average_filter(image, kernel_size):
    return cv2.blur(image, (kernel_size, kernel_size))

# Fungsi untuk menerapkan filter Gaussian pada citra
def gaussian_filter(image, kernel_size):
    return cv2.GaussianBlur(image, (kernel_size, kernel_size), 0)

# Fungsi untuk menerapkan filter median pada citra
def median_filter(image, kernel_size):
    return cv2.medianBlur(image, kernel_size)

# Fungsi untuk memperoleh tepi citra menggunakan operator Sobel
def sobel_edge_detection(image):
    sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    gradient_magnitude = np.sqrt(sobel_x**2 + sobel_y**2)
    gradient_magnitude *= 255.0 / gradient_magnitude.max()
    return gradient_magnitude.astype(np.uint8)

# Fungsi untuk memperoleh tepi citra menggunakan operator Canny
def canny_edge_detection(image, threshold1, threshold2):
    return cv2.Canny(image, threshold1, threshold2)

# Fungsi untuk menerapkan filter sharpening pada citra
def sharpening_filter(image, kernel_size, alpha):
    blurred = cv2.GaussianBlur(image, (kernel_size, kernel_size), 0)
    sharpened = cv2.addWeighted(image, 1 + alpha, blurred, -alpha, 0)
    return sharpened

# Fungsi untuk menerapkan filter min pada citra
def min_filter(image, kernel_size):
    kernel = np.ones((kernel_size, kernel_size), np.uint8)
    return cv2.erode(image, kernel)

# Fungsi untuk menerapkan filter max pada citra
def max_filter(image, kernel_size):
    kernel = np.ones((kernel_size, kernel_size), np.uint8)
    return cv2.dilate(image, kernel)

# Fungsi untuk menerapkan filter laplacian pada citra
def laplacian_filter(image):
    return cv2.Laplacian(image, cv2.CV_64F)


# Load citra
image = cv2.imread('G:\MK\SEMESTER 4\PCD\Tugas 3\\3_11zon-1.jpg')

# Mengubah citra menjadi grayscale
gray_image = grayscale(image)

# Menerapkan filter rata-rata pada citra grayscale
average_filtered_image = average_filter(gray_image, kernel_size=5)

# Menerapkan filter Gaussian pada citra grayscale
gaussian_filtered_image = gaussian_filter(gray_image, kernel_size=5)

```

```
# Menerapkan filter median pada citra grayscale
median_filtered_image = median_filter(gray_image, kernel_size=5)

# Mendeteksi tepi citra menggunakan operator Sobel pada citra grayscale
sobel_edges = sobel_edge_detection(gray_image)

# Mendeteksi tepi citra menggunakan operator Canny pada citra grayscale
canny_edges = canny_edge_detection(gray_image, threshold1=100,
threshold2=200)

# Menerapkan filter sharpening pada citra grayscale
sharpened_image = sharpening_filter(gray_image, kernel_size=5, alpha=1.
0)

# Menerapkan filter min pada citra grayscale
min_filtered_image = min_filter(image, kernel_size=3)

# Menerapkan filter max pada citra grayscale
max_filtered_image = max_filter(image, kernel_size=3)

# Menerapkan filter laplacian pada citra grayscale
laplacian_filtered_image = laplacian_filter(image)

# Menampilkan citra-citra hasil pemrosesan
plt.subplot(4, 3, 1)
plt.imshow(image[:,:,::-1])
plt.title('Original Image')
plt.axis('off')

plt.subplot(4, 3, 2)
plt.imshow(gray_image, cmap='gray')
plt.title('Gray Scale Image')
plt.axis('off')

plt.subplot(4, 3, 3)
plt.imshow(average_filtered_image, cmap='gray')
plt.title('Average Filtered Image')
plt.axis('off')

plt.subplot(4, 3, 4)
plt.imshow(gaussian_filtered_image, cmap='gray')
plt.title('Gaussian Filtered Image')
plt.axis('off')

plt.subplot(4, 3, 5)
plt.imshow(median_filtered_image, cmap='gray')
plt.title('Median Filtered Image')
plt.axis('off')

plt.subplot(4, 3, 6)
plt.imshow(sobel_edges, cmap='gray')
plt.title('Sobel Edges Image')
plt.axis('off')

plt.subplot(4, 3, 7)
plt.imshow(canny_edges, cmap='gray')
plt.title('Canny Edges Image')
plt.axis('off')

plt.subplot(4, 3, 8)
plt.imshow(sharpened_image, cmap='gray')
plt.title('Sharpened Image')
plt.axis('off')

```

```
128    plt.subplot(4, 3, 9)
129    plt.imshow(min_filtered_image, cmap='gray')
130    plt.title('Min filter Image')
131    plt.axis('off')
132
133    plt.subplot(4, 3, 10)
134    plt.imshow(max_filtered_image, cmap='gray')
135    plt.title('Max Filter Image')
136    plt.axis('off')
137
138    plt.subplot(4, 3, 11)
139    plt.imshow(laplacian_filtered_image, cmap='gray')
140    plt.title('Laplacian Filutered Image')
141    plt.axis('off')
142
143    plt.show()
```

III. HASIL PERCOBAAN



IV. ANALISIS

Program di atas merupakan contoh implementasi pemrosesan citra menggunakan library OpenCV dan NumPy. Pada awal program, beberapa library yang diperlukan diimpor, yaitu cv2 untuk pemrosesan citra, numpy untuk operasi matriks, dan matplotlib.pyplot untuk menampilkan gambar.

Selanjutnya, program mendefinisikan beberapa fungsi untuk menerapkan filter dan operasi pemrosesan citra. Fungsi-fungsi tersebut antara lain adalah fungsi untuk mengubah citra menjadi grayscale, menerapkan filter rata-rata, filter Gaussian, filter median, mendeteksi tepi citra menggunakan operator Sobel, mendeteksi tepi citra menggunakan operator Canny, menerapkan filter

sharpening, menerapkan filter min, menerapkan filter max, dan menerapkan filter Laplacian.

Setelah definisi fungsi-fungsi, program memuat citra menggunakan fungsi cv2.imread dengan path file citra yang diberikan. Citra tersebut kemudian diubah menjadi citra grayscale menggunakan fungsi grayscale.

Selanjutnya, program menerapkan berbagai filter dan operasi pemrosesan citra pada citra grayscale. Filter yang diterapkan antara lain filter rata-rata, filter Gaussian, filter median, operator Sobel untuk mendeteksi tepi citra, operator Canny untuk mendeteksi tepi citra, filter sharpening, filter min, filter max, dan filter Laplacian.

Hasil pemrosesan citra kemudian ditampilkan menggunakan library matplotlib.pyplot. Program menggunakan plt.subplot untuk menampilkan citra-citra hasil pemrosesan dalam bentuk subplot yang terdiri dari beberapa baris dan kolom. Setiap citra ditampilkan dengan menggunakan plt.imshow dengan menerapkan cmap='gray' untuk menampilkan citra grayscale. Program juga memberikan judul pada setiap subplot dan menghilangkan sumbu x dan y pada gambar.

Terakhir, hasil pemrosesan citra ditampilkan dengan menggunakan plt.show() untuk menampilkan semua subplot yang telah dibuat.

Dengan demikian, program ini mengilustrasikan berbagai operasi pemrosesan citra seperti filter, deteksi tepi, dan operator Laplacian yang dapat diterapkan pada citra grayscale menggunakan library OpenCV dan NumPy.