PERTEMUAN IX

MORFOLOGI CITRA

1. Dilasi dan Erosi dalam Morfologi Citra

adalah dua operasi dasar dalam morfologi citra yang berfungsi untuk memanipulasi bentuk dan ukuran objek pada citra. Keduanya memiliki efek yang berlawanan dan merupakan dasar dari operasi morfologi lainnya, seperti opening dan closing. Operasi ini diterapkan pada citra biner atau grayscale, biasanya dengan tujuan untuk memperbaiki struktur atau kontur objek.

A. Dilasi (Dilation)

adalah operasi morfologi yang berfungsi untuk memperbesar atau memperluas area objek pada citra. Dilasi bertujuan untuk menambah piksel ke batas objek, sehingga objek tampak lebih tebal atau lebih besar. Operasi ini sangat berguna untuk mengisi lubang kecil atau celah di dalam objek dan menghubungkan objek yang berdekatan. Dilasi dilakukan dengan menggunakan elemen struktural. Elemen ini ditempatkan pada setiap piksel dalam citra, dan jika ada tumpang tindih antara piksel objek dan elemen struktural, piksel di posisi pusat elemen tersebut akan dianggap sebagai bagian dari objek dalam hasil dilasi. Dalam pengolahan citra, hal ini menyebabkan piksel objek "meluas" atau "menyebar" mengikuti bentuk elemen struktural.

Code Sample :

```
#
============================================================
==========
# OPERASI DILASI (DILATION) CITRA SIMPEL MENGGUNAKAN OPENCV
#
============================================================
==========
# Kode ini mendemonstrasikan:
# 1. Konsep dasar dilasi
# 2. Pengaruh ukuran kernel
# 3. Pengaruh bentuk kernel (Rectangle, Ellipse, Cross)
# 4. Pengaruh jumlah iterasi
```

```
# 5. Aplikasi praktis dilasi
#
============================================================
===========

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

#
============================================================
===========
# FUNGSI PEMBANTU
#
============================================================
===========

def imshow(title, image, cmap=None, figsize=(8, 6)):
    """Menampilkan gambar dengan matplotlib"""
    plt.figure(figsize=figsize)
    if len(image.shape) == 3:
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    else:
        plt.imshow(image, cmap=cmap)
    plt.title(title, fontsize=12)
    plt.axis('off')
    plt.tight_layout()
    plt.show()

def show_comparison(original, processed, title1="Asli", title2="Dilasi",
suptitle="Perbandingan"):
```

```
    """Menampilkan perbandingan gambar asli dan hasil dilasi"""
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

    # Gambar asli
    if len(original.shape) == 3:
        ax1.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    else:
        ax1.imshow(original, cmap='gray')
    ax1.set_title(title1)
    ax1.axis('off')

    # Gambar hasil dilasi
    if len(processed.shape) == 3:
        ax2.imshow(cv2.cvtColor(processed, cv2.COLOR_BGR2RGB))
    else:
        ax2.imshow(processed, cmap='gray')
    ax2.set_title(title2)
    ax2.axis('off')

    plt.suptitle(suptitle, fontsize=14)
    plt.tight_layout()
    plt.show()

def create_dilation_demo_image():
    """Membuat gambar demo untuk demonstrasi dilasi"""
    # Buat citra biner 200x300
    img = np.zeros((200, 300), dtype=np.uint8)

    # Objek utama: Persegi panjang
    cv2.rectangle(img, (50, 50), (150, 100), 255, -1)
```

```
    # Objek kecil: Lingkaran
    cv2.circle(img, (220, 75), 15, 255, -1)

    # Objek tipis: Garis horizontal
    cv2.line(img, (50, 140), (200, 140), 255, 5)

    # Objek tipis: Garis vertikal
    cv2.line(img, (250, 50), (250, 150), 255, 5)

    # Tambahkan sedikit noise
    noise = np.random.randint(0, 10, img.shape)
    img[noise > 8] = 255

    return img

def create_structuring_element(shape='rect', size=3):
    """Membuat structuring element untuk dilasi"""
    if shape == 'rect':
        return cv2.getStructuringElement(cv2.MORPH_RECT, (size, size))
    elif shape == 'ellipse':
        return cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (size, size))
    elif shape == 'cross':
        return cv2.getStructuringElement(cv2.MORPH_CROSS, (size, size))
    else:
        return cv2.getStructuringElement(cv2.MORPH_RECT, (size, size))

print("="*60)
print("OPERASI DILASI CITRA SIMPEL")
print("="*60)
```

```
#
==========
# BAGIAN 1: PEMBUATAN GAMBAR DEMO
#
==========

print("\n--- 1. Pembuatan Gambar Demo ---")

# Buat gambar demo biner
demo_img = create_dilation_demo_image()

print("Gambar Demo Dibuat:")
print("  • Persegi panjang (50x50 px)")
print("  • Lingkaran kecil (diameter 30 px)")
print("  • Garis horizontal (panjang 150 px)")
print("  • Garis vertikal (panjang 100 px)")
print("  • Noise acak (10 titik kecil)")

imshow("Gambar Demo Asli", demo_img, cmap='gray')

# Hitung statistik awal
original_area = np.sum(demo_img == 255)
print(f"\n Statistik Awal:")
print(f"  • Total piksel putih (objek): {original_area} px")
print(f"  • Jumlah objek: {len(cv2.connectedComponents(demo_img)[1]) - 1}")

#
==========
```

```
# BAGIAN 2: KONSEP DASAR DILASI
#
============================================================
==========

print("\n--- 2. Konsep Dasar Dilasi ---")

print("""
🎯 KONSEP DILASI (Dilation):
• Dilasi = 'Pembesaran' objek dalam citra
• Setiap piksel putih (255) yang bersentuhan dengan kernel menjadi putih
• Kernel (structuring element) = 'cap' yang 'mengembang' objek
• Efek: Objek membesar, celah mengecil, noise bisa tersambung

 FUNGSI OPENCV:
cv2.dilate(image, kernel, iterations=1)
""")

# Dilasi dasar dengan kernel 3x3 rectangle
kernel_basic = create_structuring_element('rect', 3)
dilation_basic = cv2.dilate(demo_img, kernel_basic, iterations=1)

show_comparison(demo_img, dilation_basic, "Asli", "Dilasi 3x3 (1 iterasi)", "Dilasi
Dasar")

# Analisis hasil
dilated_area = np.sum(dilation_basic == 255)
print(f"\n Analisis Dilasi Dasar:")
print(f"  • Area asli: {original_area} px")
print(f"  • Area setelah dilasi: {dilated_area} px")
```

```
print(f"   • Peningkatan area: {dilated_area - original_area} px (+{((dilated_area-original_area)/original_area*100):.1f}%)")
print(f"  • Lebar objek bertambah: ~{3-1} px di setiap sisi")

#
==========
# BAGIAN 3: PENGARUH UKURAN KERNEL
#
==========

print("\n--- 3. Pengaruh Ukuran Kernel ---")

print("""
🔧UKURAN KERNEL:
• Kernel 3x3: Efek ringan, detail terjaga
• Kernel 5x5: Efek sedang, pembesaran signifikan
• Kernel 7x7: Efek kuat, objek sangat membesar
• Ukuran GENJIL (3,5,7,9...) untuk anchor di tengah
""")

# Test berbagai ukuran kernel
kernel_sizes = [3, 5, 7]
dilation_sizes = {}

plt.figure(figsize=(15, 4))
for i, size in enumerate(kernel_sizes):
    kernel = create_structuring_element('rect', size)
    result = cv2.dilate(demo_img, kernel, iterations=1)
    dilation_sizes[f"{size}x{size}"] = result
```

```
    # Hitung area
    area = np.sum(result == 255)
    increase = area - original_area

    plt.subplot(1, len(kernel_sizes), i+1)
    plt.imshow(result, cmap='gray')
    plt.title(f"Kernel {size}x{size}\nArea: {area} px\n(+{increase} px)", fontsize=10)
    plt.axis('off')

plt.suptitle("Pengaruh Ukuran Kernel pada Dilasi", fontsize=14)
plt.tight_layout()
plt.show()

# Tabel analisis ukuran kernel
print(f"\n Analisis Ukuran Kernel:")
print(f"{'Ukuran':<8} {'Area (px)':<10} {'Peningkatan':<12} {'% Peningkatan':<12}")
print("-" * 45)
for size in kernel_sizes:
    result = dilation_sizes[f"{size}x{size}"]
    area = np.sum(result == 255)
    increase = area - original_area
    percent = (increase / original_area) * 100
    print(f"{size}x{size:<6} {area:<10} {increase:<12} {percent:<11.1f}%")

#
============================================================
===========
# BAGIAN 4: PENGARUH BENTUK KERNEL
```

```
#
==========================================================================
==========

print("\n--- 4. Pengaruh Bentuk Kernel ---")

print("""
BENTUK KERNEL:
• Rectangle: Efek persegi, cocok untuk garis H/V
• Ellipse: Efek bulat, cocok untuk objek melingkar
• Cross: Efek silang, cocok untuk sambungan diagonal
""")

# Test berbagai bentuk kernel (ukuran 5x5)
kernel_shapes = {
    "Rectangle": create_structuring_element('rect', 5),
    "Ellipse": create_structuring_element('ellipse', 5),
    "Cross": create_structuring_element('cross', 5)
}

dilation_shapes = {}
plt.figure(figsize=(15, 4))

for i, (shape_name, kernel) in enumerate(kernel_shapes.items()):
    result = cv2.dilate(demo_img, kernel, iterations=1)
    dilation_shapes[shape_name] = result

    area = np.sum(result == 255)
    plt.subplot(1, 3, i+1)
    plt.imshow(result, cmap='gray')
    plt.title(f"{shape_name}\nArea: {area} px", fontsize=10)
```

```
    plt.axis('off')

plt.suptitle("Pengaruh Bentuk Kernel (5x5) pada Dilasi", fontsize=14)
plt.tight_layout()
plt.show()

# Visualisasi kernel
fig, axes = plt.subplots(1, 3, figsize=(12, 3))
for i, (shape_name, kernel) in enumerate(kernel_shapes.items()):
    axes[i].imshow(kernel, cmap='gray')
    axes[i].set_title(f"Kernel {shape_name}", fontsize=10)
    axes[i].axis('off')

plt.suptitle("Visualisasi Structuring Element (Kernel)", fontsize=14)
plt.tight_layout()
plt.show()

# Analisis entuk kernel
print(f"\nAnalisis Bentuk Kernel (5x5):")
print(f"{'Bentuk':<10} {'Area (px)':<10} {'Peningkatan':<12}")
print("-" * 35)
for shape_name in kernel_shapes.keys():
    result = dilation_shapes[shape_name]
    area = np.sum(result == 255)
    increase = area - original_area
    print(f"{shape_name:<10} {area:<10} {increase:<12}")

print(f"\n Observasi:")
print(f"  • Rectangle: Paling besar (bentuk persegi)")
print(f"  • Ellipse: Sedang (bentuk bulat)")
print(f"  • Cross: Paling kecil (hanya 4 arah)")
```

```
#
==========
# BAGIAN 5: PENGARUH JUMLAH ITERASI
#
==========

print("\n--- 5. Pengaruh Jumlah Iterasi ---")

print("""
 ITERASI DILASI:
• 1 iterasi: Efek dasar
• 2 iterasi: Efek sedang (2x kernel)
• 3 iterasi: Efek kuat (3x kernel)
• Setiap iterasi = aplikasi kernel berulang
""")

# Test berbagai iterasi
iterations_list = [1, 2, 3]
dilation_iterations = {}

plt.figure(figsize=(15, 4))
for i, n_iter in enumerate(iterations_list):
    result = cv2.dilate(demo_img, kernel_basic, iterations=n_iter)
    dilation_iterations[f"{n_iter} Iterasi"] = result

    # Hitung area efektif
    effective_size = 3 + 2*(n_iter-1)  # Ukuran efektif kernel
    area = np.sum(result == 255)
```

```
    plt.subplot(1, 3, i+1)
    plt.imshow(result, cmap='gray')
    plt.title(f"{n_iter} Iterasi\nKernel efektif: {effective_size}x{effective_size}\nArea: {area}
px",
              fontsize=9)
    plt.axis('off')

plt.suptitle("Pengaruh Jumlah Iterasi pada Dilasi", fontsize=14)
plt.tight_layout()
plt.show()

# Analisis iterasi
print(f"\nAnalisis Iterasi (Kernel 3x3):")
print(f"{'Iterasi':<8} {'Kernel Efektif':<14} {'Area (px)':<10} {'Peningkatan':<12}")
print("-" * 50)
for n_iter in iterations_list:
    result = dilation_iterations[f"{n_iter} Iterasi"]
    effective_size = 3 + 2*(n_iter-1)
    area = np.sum(result == 255)
    increase = area - original_area
    print(f"{n_iter:<7} {effective_size}x{effective_size:<11} {area:<10} {increase:<12}")

#
==========================================================================
==========
# BAGIAN 6: APLIKASI PRAKTIS DILASI
#
==========================================================================
==========
```

```
print("\n--- 6. Aplikasi Praktis Dilasi ---")

# 6.1 Mengisi Celah Kecil
print("\n 6.1 Mengisi Celah Kecil dalam Objek")

# Buat objek dengan celah
img_with_gaps = demo_img.copy()
# Buat celah acak pada persegi panjang
gap_y, gap_x = np.random.randint(55, 95, 2)
cv2.line(img_with_gaps, (gap_x, 50), (gap_x, 100), 0, 3)

# Isi celah dengan dilasi
kernel_gap = create_structuring_element('rect', 3)
filled_gaps = cv2.dilate(img_with_gaps, kernel_gap, iterations=2)

show_comparison(img_with_gaps, filled_gaps, "Dengan Celah", "Celah Terisi", "Aplikasi:
Gap Filling")

print(f"\nAnalisis Gap Filling:")
print(f"  • Area sebelum: {np.sum(img_with_gaps == 255)} px")
print(f"  • Area setelah: {np.sum(filled_gaps == 255)} px")
print(f"  • Celah terisi: {np.sum(filled_gaps == 255) - np.sum(img_with_gaps == 255)} px")

# 6.2 Menyambung Objek Terpisah
print("\n6.2 Menyambung Objek yang Terpisah")

# Buat dua objek terpisah
img_separated = np.zeros((150, 250), dtype=np.uint8)
cv2.rectangle(img_separated, (30, 30), (80, 80), 255, -1)   # Objek kiri
cv2.rectangle(img_separated, (120, 30), (170, 80), 255, -1) # Objek kanan
```

```
# Sambung dengan dilasi
kernel_connect = create_structuring_element('rect', 7)
connected_objects = cv2.dilate(img_separated, kernel_connect, iterations=1)

show_comparison(img_separated, connected_objects, "Objek Terpisah", "Objek Tersambung", "Aplikasi: Object Connection")

# Hitung komponen terhubung
components_original = len(cv2.connectedComponents(img_separated)[1]) - 1
components_connected = len(cv2.connectedComponents(connected_objects)[1]) - 1

print(f"\nAnalisis Object Connection:")
print(f"  • Komponen asli: {components_original}")
print(f"  • Komponen setelah dilasi: {components_connected}")
print(f"  • Objek {'✓Tersambung' if components_connected < components_original else 'Masih terpisah'}")

# 6.3 Preprocessing untuk Edge Detection
print("\n 6.3 Preprocessing untuk Deteksi Tepi")

# Gambar dengan noise untuk edge detection
noisy_edges = demo_img.copy()
# Tambahkan noise
noise = np.random.randint(0, 255, noisy_edges.shape)
noisy_edges[noise > 240] = 255

# Preprocessing: Dilasi untuk memperkuat objek
preprocessed = cv2.dilate(noisy_edges, kernel_basic, iterations=1)

# Edge detection sebelum dan sesudah
edges_original = cv2.Canny(noisy_edges, 50, 150)
```

```
edges_preprocessed = cv2.Canny(preprocessed, 50, 150)

fig, axes = plt.subplots(2, 2, figsize=(10, 8))
axes[0,0].imshow(noisy_edges, cmap='gray')
axes[0,0].set_title('Gambar Ber-noise')
axes[0,0].axis('off')

axes[0,1].imshow(edges_original, cmap='gray')
axes[0,1].set_title('Edge Detection (Tanpa Preprocessing)')
axes[0,1].axis('off')

axes[1,0].imshow(preprocessed, cmap='gray')
axes[1,0].set_title('Setelah Dilasi (Preprocessing)')
axes[1,0].axis('off')

axes[1,1].imshow(edges_preprocessed, cmap='gray')
axes[1,1].set_title('Edge Detection (Dengan Preprocessing)')
axes[1,1].axis('off')

plt.suptitle("Dilasi sebagai Preprocessing untuk Edge Detection", fontsize=14)
plt.tight_layout()
plt.show()

print(f"\n Analisis Edge Detection:")
print(f"  • Tepi tanpa preprocessing: {np.sum(edges_original > 0)} px")
print(f"  • Tepi dengan preprocessing: {np.sum(edges_preprocessed > 0)} px")
print(f"  • Preprocessing {'✓ Memperbaiki' if np.sum(edges_preprocessed > 0) > np.sum(edges_original > 0) else 'Tidak efektif'} deteksi tepi")
```

```
#
==========
# BAGIAN 7: RINGKASAN DAN INTERPRETASI
#
==========

print("\n" + "="*60)
print("7. RINGKASAN OPERASI DILASI")
print("="*60)

print("""
🎯 RINGKASAN DILASI:

KONSEP DASAR:
• Dilasi = Pembesaran objek menggunakan structuring element
• Setiap piksel putih yang bersentuhan dengan kernel → menjadi putih
• Efek: Objek membesar, celah mengecil, noise bisa tersambung

PARAMETER PENTING:
• Kernel Size: 3x3 (ringan), 5x5 (sedang), 7x7+ (kuat)
• Kernel Shape: Rectangle (persegi), Ellipse (bulat), Cross (silang)
• Iterations: 1 (dasar), 2-3 (kuat), >3 (sangat kuat)

APLİKASI UTAMA:
•  Mengisi celah dalam teks/karakter
•  Menyambung objek yang terputus
• Preprocessing untuk edge detection
• Object growth dalam segmentasi
• Noise reduction (kombinasi dengan erosi)
```

KELEBIHAN:

- *✓Cepat dan efisien*
- *✓Tidak memerlukan parameter kompleks*
- *✓Robust terhadap noise*
- *✓Mudah dikontrol dengan kernel*

KETERBATASAN:

- *✗Bisa menyambung objek yang seharusnya terpisah*
- *✗Memperbesar noise jika tidak dikombinasikan*
- *✗Kurang presisi untuk bentuk kompleks*

""")

```
# Tabel ringkasan efek parameter
summary_data = {
    'Parameter': ['Kernel 3x3', 'Kernel 5x5', 'Kernel 7x7', 'Rectangle', 'Ellipse', 'Cross', '1 Iterasi', '2 Iterasi', '3 Iterasi'],
    'Efek': ['Pembesaran ringan', 'Pembesaran sedang', 'Pembesaran kuat', 'Bentuk persegi', 'Bentuk bulat', '4 arah saja', 'Efek dasar', 'Efek sedang', 'Efek kuat'],
    'Aplikasi': ['Detail halus', 'Objek sedang', 'Objek besar', 'Garis H/V', 'Objek bulat', 'Sambungan diagonal', 'Preprocessing', 'Gap filling', 'Object growth']
}

import pandas as pd
df_summary = pd.DataFrame(summary_data)
print("\nTabel Ringkasan Parameter Dilasi:")
print(df_summary.to_string(index=False))

print(f"\nTIPS IMPLEMENTASI:")
print("  • Mulai dengan kernel 3x3 untuk testing")
```

```
print("  • Gunakan Rectangle untuk teks/garis")
print("  • Gunakan Ellipse untuk objek bulat")
print("  • Kombinasikan dengan Erosi (Opening/Closing) untuk hasil optimal")
print("  • Test iterasi 1-2 untuk kebanyakan aplikasi")
print("  • Visualisasikan always before/after untuk evaluasi")

print(f"\n🔧 CONTOH KODE CEPAT:")
print("""
# Dilasi dasar
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
result = cv2.dilate(image, kernel, iterations=1)

# Gap filling
kernel_fill = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
filled = cv2.dilate(image_with_gaps, kernel_fill, iterations=2)

# Preprocessing edge detection
kernel_edge = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
preprocessed = cv2.dilate(noisy_image, kernel_edge, iterations=1)
""")

print("\n" + "="*60)
print("✅ OPERASI DILASI CITRA SIMPEL SELESAI!")
print("Konsep yang berhasil dipahami:")
print("  • Dasar dilasi dan cara kerjanya")
print("  • Pengaruh ukuran, bentuk, dan iterasi kernel")
print("  • Aplikasi: gap filling, object connection, preprocessing")
print("  • Analisis kuantitatif (area, komponen)")
print("="*60)
```

OUTPUT ??

2. Morfologi Closing

adalah operasi morfologi yang terdiri dari dua langkah: pertama, dilakukan dilasi pada citra menggunakan elemen struktural, diikuti oleh langkah kedua yaitu erosi pada hasil dilasi. Secara keseluruhan, closing bertujuan untuk mengisi lubang atau celah kecil dalam objek dan menghubungkan objek yang terpisah. Fungsi Closing sebagai berikut:

- Menutup lubang atau celah kecil dalam objek.
- Menghubungkan objek yang terpisah (misalnya objek yang tidak terhubung sepenuhnya).
- Memperhalus objek dan menghilangkan bagian yang terpotong.
- Mengisi area kosong kecil dalam objek yang lebih besar.

Code Sample :

```
# ============================================================================
# OPERASI CLOSING MORFOLOGI PADA CITRA
# ============================================================================
# Closing = Dilation diikuti Erosion
# Fungsi: Menutup hole kecil, menyambungkan objek yang dekat, menghaluskan boundaries
# ============================================================================

import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import data, morphology
from scipy import ndimage

# ============================================================================
# 1. FUNGSI PEMBANTU DAN SETUP
# ============================================================================

def imshow_grid(images, titles, cmap='gray', figsize=(15, 8)):
    """Menampilkan multiple gambar dalam grid"""
    n = len(images)
```

```
    rows = (n + 2) // 3
    cols = min(n, 3)

    plt.figure(figsize=figsize)
    for i in range(n):
        plt.subplot(rows, cols, i + 1)
        if len(images[i].shape) == 3 and images[i].shape[2] == 3:
            plt.imshow(cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB))
        else:
            plt.imshow(images[i], cmap=cmap)
        plt.title(titles[i], fontsize=12, fontweight='bold')
        plt.axis('off')
    plt.tight_layout()
    plt.show()

def create_sample_with_holes():
    """Membuat citra sampel dengan holes dan celah"""
    img = np.ones((300, 400), dtype=np.uint8) * 255  # Background putih

    # 1. Objek dengan holes (lingkaran)
    cv2.circle(img, (100, 100), 50, 0, -1)
    cv2.circle(img, (100, 100), 20, 255, -1)  # Hole di tengah
    cv2.circle(img, (80, 80), 10, 255, -1)    # Hole kecil
    cv2.circle(img, (120, 80), 10, 255, -1)   # Hole kecil

    # 2. Objek terputus (garis putus-putus)
    for i in range(0, 150, 20):
        cv2.rectangle(img, (200 + i, 50), (210 + i, 100), 0, -1)

    # 3. Objek dengan celah kecil
    cv2.rectangle(img, (150, 150), (250, 250), 0, -1)
    # Buat celah di tengah
```

```
    cv2.rectangle(img, (190, 190), (210, 210), 255, -1)

    # 4. Text dengan celah
    cv2.putText(img, 'HOLE', (280, 200), cv2.FONT_HERSHEY_SIMPLEX, 1, 0, 3)

    return img

def create_noisy_image():
    """Membuat citra dengan pepper noise (hole-like noise)"""
    img = np.ones((200, 300), dtype=np.uint8) * 255

    # Objek utama
    cv2.rectangle(img, (50, 50), (150, 150), 0, -1)
    cv2.circle(img, (220, 100), 40, 0, -1)

    # Tambahkan pepper noise (hole-like)
    noise_mask = np.random.random(img.shape) < 0.1
    img[noise_mask] = 255  # Pepper noise

    return img

# ============================================================================
# 2. MEMUAT DAN PREPROCESSING CITRA
# ============================================================================

print("="*70)
print("OPERASI CLOSING MORFOLOGI - MENUTUP HOLE DAN CELAH")
print("="*70)

print("\nMEMUAT CITRA SAMPEL...")

# Buat citra sampel
```

```
image_holes = create_sample_with_holes()
image_noisy = create_noisy_image()

# Konversi ke biner
_, binary_holes = cv2.threshold(image_holes, 127, 255, cv2.THRESH_BINARY_INV)
_, binary_noisy = cv2.threshold(image_noisy, 127, 255, cv2.THRESH_BINARY_INV)

print("✓ Citra sampel berhasil dibuat")
print(f"▪ Dimensi citra: {binary_holes.shape}")
print(f"▪ Tipe citra: {binary_holes.dtype}")

# ============================================================================
# 3. MEMBUAT STRUCTURING ELEMENTS
# ============================================================================

print("\n" + "="*50)
print("MEMBUAT STRUCTURING ELEMENTS")
print("="*50)

# Buat berbagai kernel
kernel_rect_3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
kernel_rect_5 = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
kernel_rect_7 = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
kernel_cross_3 = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
kernel_ellipse_5 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))

kernels = [kernel_rect_3, kernel_rect_5, kernel_rect_7, kernel_cross_3, kernel_ellipse_5]
kernel_names = ['Rectangle 3x3', 'Rectangle 5x5', 'Rectangle 7x7', 'Cross 3x3', 'Ellipse 5x5']

# Visualisasi kernel
plt.figure(figsize=(15, 3))
for i, (kernel, name) in enumerate(zip(kernels, kernel_names)):
```

```
    plt.subplot(1, 5, i + 1)
    plt.imshow(kernel, cmap='gray', vmin=0, vmax=1)
    plt.title(name, fontweight='bold')
    plt.axis('off')
plt.tight_layout()
plt.show()

# ============================================================================
# 4. OPERASI CLOSING DASAR
# ============================================================================

print("\n" + "="*50)
print("OPERASI CLOSING DASAR")
print("="*50)

print("Melakukan operasi closing dengan kernel 5x5...")

# Closing dengan kernel 5x5
closing_basic = cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel_rect_5)

# Tampilkan proses step-by-step
dilation_step = cv2.dilate(binary_holes, kernel_rect_5, iterations=1)
closing_step = cv2.erode(dilation_step, kernel_rect_5, iterations=1)

images_basic = [binary_holes, dilation_step, closing_step, closing_basic]
titles_basic = [
    'Original\n(Ada holes & celah)',
    'Step 1: Dilation\n(Mengisi holes sementara)',
    'Step 2: Erosion\n(Mengembalikan ukuran)',
    'Final: Closing\n(Holes tertutup)'
]
```

```
imshow_grid(images_basic, titles_basic, cmap='gray')

# Analisis hasil
original_holes = np.sum(binary_holes == 0)  # Hitung area hitam (holes)
after_closing = np.sum(closing_basic == 0)
holes_filled = original_holes - after_closing

print(f"\nANALISIS HASIL CLOSING:")
print(f"▪ Area holes sebelum closing: {original_holes} px")
print(f"▪ Area holes setelah closing: {after_closing} px")
print(f"▪ Holes yang berhasil ditutup: {holes_filled} px")
print(f"▪ Efektivitas: {(holes_filled/original_holes)*100:.1f}%")

# ============================================================================
# 5. PENGARUH UKURAN KERNEL PADA CLOSING
# ============================================================================

print("\n" + "="*50)
print(" PENGARUH UKURAN KERNEL")
print("="*50)

print(" Menguji berbagai ukuran kernel...")

# Closing dengan berbagai ukuran kernel
closing_results = [binary_holes]
closing_titles = ['Original']

kernel_sizes = [3, 5, 7, 9, 11]

for size in kernel_sizes:
    kernel_temp = cv2.getStructuringElement(cv2.MORPH_RECT, (size, size))
    closing_temp = cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel_temp)
```

```
    closing_results.append(closing_temp)
    closing_titles.append(f'Closing {size}x{size}')

# Tampilkan hasil
imshow_grid(closing_results, closing_titles, cmap='gray', figsize=(16, 10))

# Analisis pengaruh ukuran kernel
print("\nPENGARUH UKURAN KERNEL:")
print("Size | Area Holes | Holes Terisi | Efektivitas")
print("-" * 50)

original_area = np.sum(binary_holes == 0)

for i, size in enumerate(kernel_sizes):
    holes_area = np.sum(closing_results[i+1] == 0)
    holes_filled = original_area - holes_area
    effectiveness = (holes_filled / original_area) * 100

    print(f"{size}x{size} | {holes_area:>10} | {holes_filled:>12} | {effectiveness:>10.1f}%")

# ============================================================================
# 6. PENGARUH BENTUK KERNEL PADA CLOSING
# ============================================================================

print("\n" + "="*50)
print("🎯 PENGARUH BENTUK KERNEL")
print("="*50)

print("Membandingkan berbagai bentuk kernel...")

# Closing dengan berbagai bentuk kernel (size 5x5)
kernel_shapes = [
```

```
    cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5)),
    cv2.getStructuringElement(cv2.MORPH_CROSS, (5, 5)),
    cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
]

shape_names = ['Rectangle', 'Cross', 'Ellipse']

shape_results = [binary_holes]
shape_titles = ['Original']

for kernel, name in zip(kernel_shapes, shape_names):
    closing_temp = cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel)
    shape_results.append(closing_temp)
    shape_titles.append(f'Closing {name}')

imshow_grid(shape_results, shape_titles, cmap='gray')

print("\nPERBANDINGAN BENTUK KERNEL:")
print("Shape    | Area Holes | Efektivitas")
print("-" * 40)

for i, name in enumerate(shape_names):
    holes_area = np.sum(shape_results[i+1] == 0)
    effectiveness = ((original_area - holes_area) / original_area) * 100
    print(f"{name:<8} | {holes_area:>10} | {effectiveness:>10.1f}%")

# ============================================================================
# 7. APLIKASI: DENOISING PEPPER NOISE
# ============================================================================

print("\n" + "="*50)
print(" APLIKASI: MENGHILANGKAN PEPPER NOISE")
```

```
print("="*50)

print("Membersihkan noise menggunakan closing...")

# Closing pada citra ber-noise
closing_noisy = cv2.morphologyEx(binary_noisy, cv2.MORPH_CLOSE, kernel_rect_5)

# Bandingkan dengan operasi lain
opening_noisy = cv2.morphologyEx(binary_noisy, cv2.MORPH_OPEN, kernel_rect_5)

noise_results = [binary_noisy, opening_noisy, closing_noisy]
noise_titles = [
    'Original Noisy\n(Pepper noise)',
    'Opening\n(Hilangkan noise kecil)',
    'Closing\n(Tutup hole noise)'
]

imshow_grid(noise_results, noise_titles, cmap='gray')

# Analisis denoising
original_noise = np.sum(binary_noisy == 255)
after_closing_noise = np.sum(closing_noisy == 255)
noise_reduction = original_noise - after_closing_noise

print(f"\n HASIL DENOISING:")
print(f"▪ Area objek sebelum cleaning: {original_noise} px")
print(f"▪ Area objek setelah closing: {after_closing_noise} px")
print(f"▪ Noise yang tereduksi: {noise_reduction} px")
print(f"▪ Efektivitas: {(noise_reduction/original_noise)*100:.1f}%")

# ============================================================================
# 8. APLIKASI: MENYAMBUNGKAN OBJEK TERPUTUS
```

```
# ============================================================================

print("\n" + "="*50)
print("🔗 APLIKASI: MENYAMBUNGKAN OBJEK TERPUTUS")
print("="*50)

print("🎯 Menggunakan closing untuk menyambungkan objek...")

# Buat citra dengan objek terputus
disconnected_img = np.ones((200, 300), dtype=np.uint8) * 255

# Buat garis putus-putus
for i in range(0, 250, 30):
    cv2.rectangle(disconnected_img, (i, 80), (i+15, 120), 0, -1)

# Buat titik-titik terpisah
points = [(50, 150), (100, 150), (150, 150), (200, 150), (250, 150)]
for point in points:
    cv2.circle(disconnected_img, point, 8, 0, -1)

binary_disconnected = cv2.threshold(disconnected_img, 127, 255, cv2.THRESH_BINARY_INV)[1]

# Closing untuk menyambungkan
closing_connected = cv2.morphologyEx(binary_disconnected, cv2.MORPH_CLOSE, kernel_rect_7)

connection_results = [binary_disconnected, closing_connected]
connection_titles = [
    'Original\n(Objek terputus-putus)',
    'After Closing\n(Objek tersambung)'
]

imshow_grid(connection_results, connection_titles, cmap='gray')
```

```
# Hitung connected components
def count_objects(binary_img):
    num_labels, labels = cv2.connectedComponents(binary_img)
    return num_labels - 1  # Kurangi 1 untuk background

original_objects = count_objects(binary_disconnected)
after_closing_objects = count_objects(closing_connected)

print(f"\nANALISIS PENYAMBUNGAN:")
print(f"▪ Jumlah objek sebelum closing: {original_objects}")
print(f"▪ Jumlah objek setelah closing: {after_closing_objects}")
print(f"▪ Pengurangan objek terpisah: {original_objects - after_closing_objects}")

# ============================================================================
# 9. CLOSING PADA CITRA GRAYSCALE
# ============================================================================

print("\n" + "="*50)
print(" CLOSING PADA CITRA GRAYSCALE")
print("="*50)

print(" Menerapkan closing pada citra grayscale...")

# Buat citra grayscale dengan variasi intensitas
gray_image = np.zeros((150, 200), dtype=np.uint8)

# Buat gradient
for i in range(200):
    gray_image[:, i] = i // 2

# Tambahkan objek dengan holes
```

```
cv2.circle(gray_image, (50, 50), 30, 200, -1)
cv2.circle(gray_image, (50, 50), 15, 100, -1)  # Hole

cv2.rectangle(gray_image, (120, 30), (180, 80), 150, -1)
cv2.rectangle(gray_image, (140, 40), (160, 70), 50, -1)  # Hole

# Closing grayscale
gray_closing = cv2.morphologyEx(gray_image, cv2.MORPH_CLOSE, kernel_rect_5)

gray_results = [gray_image, gray_closing]
gray_titles = [
    'Original Grayscale\n(Dengan area gelap/holes)',
    'Gray Closing\n(Area gelap/holes tertutup)'
]

imshow_grid(gray_results, gray_titles, cmap='viridis')

print("✓ Closing grayscale berhasil diaplikasikan")
print("▪ Pada grayscale, closing mengisi area dengan intensitas rendah")
print("▪ Efektif untuk menghaluskan tekstur gelap")

# ============================================================================
# 10. OPTIMALISASI: MEMILIH PARAMETER TERBAIK
# ============================================================================

print("\n" + "="*50)
print("⚡ OPTIMALISASI PARAMETER CLOSING")
print("="*50)

print("🎯 Mencari parameter optimal untuk aplikasi spesifik...")

# Test case: Menutup holes pada objek lingkaran
```

```
test_image = np.ones((100, 100), dtype=np.uint8) * 255
cv2.circle(test_image, (50, 50), 40, 0, -1)
# Tambahkan holes berbagai ukuran
cv2.circle(test_image, (50, 50), 15, 255, -1)  # Hole besar
cv2.circle(test_image, (30, 30), 5, 255, -1)   # Hole kecil
cv2.circle(test_image, (70, 30), 3, 255, -1)   # Hole sangat kecil

binary_test = cv2.threshold(test_image, 127, 255, cv2.THRESH_BINARY_INV)[1]

# Test berbagai parameter
optimal_results = []
optimal_titles = []

# Kernel size optimal
for size in [3, 5, 7]:
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (size, size))
    result = cv2.morphologyEx(binary_test, cv2.MORPH_CLOSE, kernel)
    optimal_results.append(result)
    optimal_titles.append(f'Kernel {size}x{size}')

# Iterasi optimal
kernel_fixed = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
for iterations in [1, 2, 3]:
    # Manual closing dengan multiple iterations
    temp = binary_test.copy()
    for _ in range(iterations):
        temp = cv2.dilate(temp, kernel_fixed)
        temp = cv2.erode(temp, kernel_fixed)
    optimal_results.append(temp)
    optimal_titles.append(f'{iterations} Iterations')

# Tambahkan original
```

```
optimal_results.insert(0, binary_test)
optimal_titles.insert(0, 'Original\n(Multiple holes)')

imshow_grid(optimal_results, optimal_titles, cmap='gray')

print("\nREKOMENDASI PARAMETER OPTIMAL:")
print("1. Untuk holes kecil (1-3 px): Kernel 3x3, 1 iterasi")
print("2. Untuk holes sedang (3-7 px): Kernel 5x5, 1 iterasi")
print("3. Untuk holes besar (>7 px): Kernel 7x7 atau multiple iterasi")
print("4. Bentuk Ellipse: Hasil lebih natural untuk objek melengkung")
print("5. Bentuk Rectangle: Hasil lebih tajam untuk objek persegi")

# ============================================================================
# 11. KESIMPULAN DAN BEST PRACTICES
# ============================================================================

print("\n" + "="*70)
print("✅ RINGKASAN OPERASI CLOSING MORFOLOGI")
print("="*70)

print(f"""
DEFINISI CLOSING:
• Operasi: Dilation diikuti Erosion
• Rumus: A • B = (A ⊕ B) ⊖ B
• Tujuan: Menutup holes kecil, menyambungkan objek terdekat

🎯 APLIKASI PRAKTIS:
1. Hole Filling: Mengisi lubang kecil dalam objek
2. Noise Removal: Menghilangkan pepper noise
3. Connection: Menyambungkan objek yang terputus
4. Smoothing: Menghaluskan boundaries objek
```

⚙PARAMETER PENTING:

• Kernel Size: Menentukan ukuran holes yang bisa ditutup

• Kernel Shape: Mempengaruhi bentuk hasil closing

• Iterations: Mengontrol kekuatan operasi

BEST PRACTICES:

• Mulai dengan kernel kecil (3x3-5x5)

• Gunakan bentuk kernel sesuai bentuk objek

• Test berbagai parameter untuk hasil optimal

• Kombinasikan dengan operasi lain untuk hasil terbaik

⚠ PERTIMBANGAN:

• Closing dapat mengubah ukuran objek asli

• Terlalu besar kernel dapat menyatukan objek yang tidak diinginkan

• Tidak efektif untuk holes yang sangat besar

CONTOH PENGGUNAAN:

```
# Closing dasar
closing = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)

# Closing dengan parameter spesifik
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))
result = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE, kernel)
""")

# Tampilkan perbandingan final
final_comparison = [
    binary_holes,
    cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel_rect_3),
    cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel_rect_5),
    cv2.morphologyEx(binary_holes, cv2.MORPH_CLOSE, kernel_rect_7)
]
```

```
final_titles = [
    'Original\n(Holes & Celah)',
    'Closing 3x3\n(Holes kecil tertutup)',
    'Closing 5x5\n(Holes sedang tertutup)',
    'Closing 7x7\n(Holes besar tertutup)'
]
print("\nPERBANDINGAN FINAL BERBAGAI UKURAN KERNEL:")
imshow_grid(final_comparison, final_titles, cmap='gray', figsize=(16, 8))
print("🎉 OPERASI CLOSING SELESAI DIPELAJARI!")
```

2.Dilation ??

adalah operasi morfologi yang berfungsi untuk memperbesar atau memperluas area objek pada citra. Dilasi bertujuan untuk menambah piksel ke batas objek, sehingga objek tampak lebih tebal atau lebih besar. Operasi ini sangat berguna untuk mengisi lubang kecil atau celah di dalam objek dan menghubungkan objek yang berdekatan. Dilasi dilakukan dengan menggunakan elemen struktural. Elemen ini ditempatkan pada setiap piksel dalam citra, dan jika ada tumpang tindih antara piksel objek dan elemen struktural, piksel di posisi pusat elemen tersebut akan dianggap sebagai bagian dari objek dalam hasil dilasi

Code :

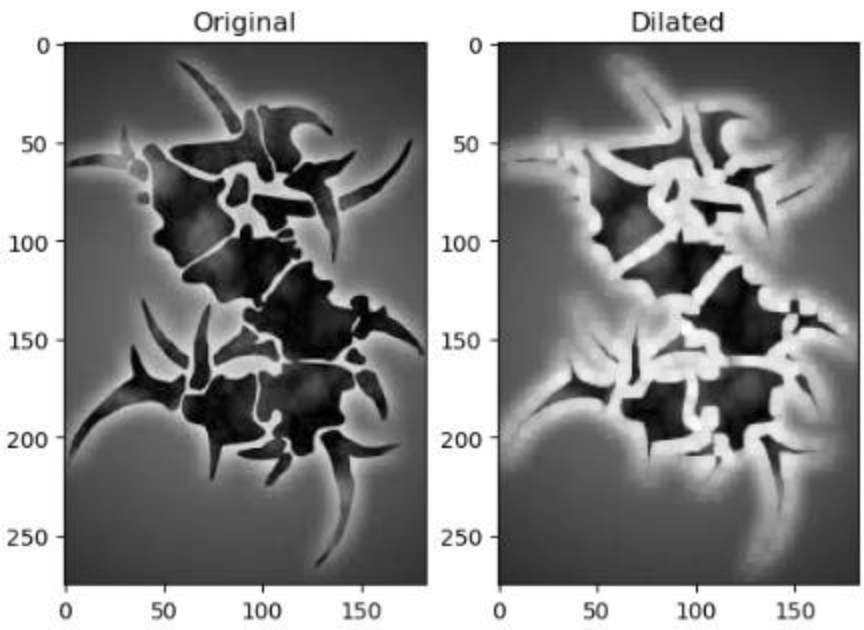
```
#DILATION CONTOH
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Membaca gambar dalam mode grayscale
img = cv2.imread('sepul1.jfif', 0)

# Membuat element structural berbentuk persegi dengan ukuran 5x5
kernel = np.ones((5,5), np.uint8)
```

```
# Melakukan operasi dilasi
dilated = cv2.dilate(img, kernel, iterations=1)

# Menampilkan hasil
plt.subplot(1, 2, 1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1, 2, 2), plt.imshow(dilated, cmap='gray'), plt.title('Dilated')
plt.show()
```

CODE CXPLAIN :

OpenCV (cv2): Digunakan untuk membaca dan memproses citra.

- NumPy: Digunakan untuk membuat elemen struktural.
- Matplotlib (pyplot): Digunakan untuk menampilkan gambar.
- cv2.imread('lingkaran.jpg', 0) membaca gambar dari file lingkaran.jpg dan mengonversinya ke grayscale. Jika gambar berwarna, grayscale akan memudahkan penerapan operasi morfologi.
- kernel = np.ones((5,5), np.uint8) membuat elemen struktural berbentuk persegi berukuran 5×55 \times 55×5. Elemen struktural ini digunakan oleh operasi dilasi untuk mempengaruhi bentuk objek dalam citra.
- dilated = cv2.dilate(img, kernel, iterations=1) melakukan operasi dilasi pada gambar. Dilasi akan memperbesar objek (misalnya, lingkaran) dan menambahkan piksel di sekeliling tepi objek, berdasarkan elemen struktural.
- Plt.subplot(1, 2, 1) menampilkan gambar asli di sebelah kiri.
- plt.subplot(1, 2, 2) menampilkan gambar yang telah dilasi di sebelah kanan.
- plt.show() akan menampilkan kedua gambar tersebut dalam satu jendela.



3. EROTION

Adalah kebalikan dari dilasi, di mana operasi ini menghilangkan atau mengurangi area objek. Dengan kata lain, erosi mengecilkan objek dengan menghapus piksel dari tepi objek, sehingga objek tampak lebih tipis atau lebih kecil. Erosi berguna untuk menghilangkan detail kecil atau noise pada citra. Pada operasi erosi, elemen struktural ditempatkan di setiap posisi dalam citra, dan objek pada posisi tersebut hanya akan dipertahankan jika semua piksel di bawah elemen struktural juga merupakan piksel objek.

Code :

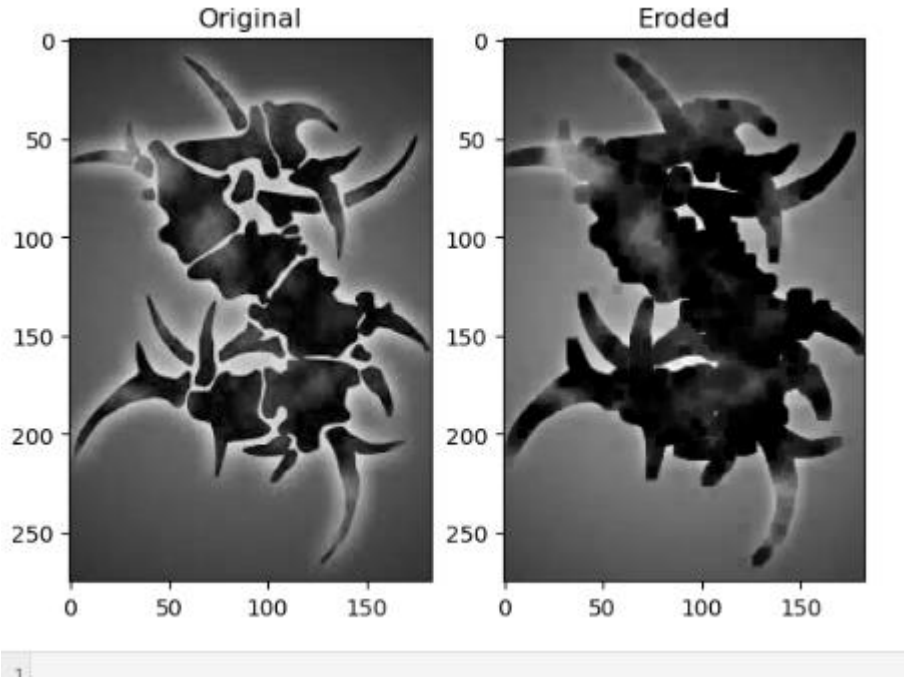
```
#EROTION
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Membaca gambar dalam mode grayscale
img = cv2.imread('sepul1.jfif', 0)

# Membuat element structural berbentuk persegi dengan ukuran 5x5
kernel = np.ones((5,5), np.uint8)
```

```
# Melakukan operasi erosi
eroded = cv2.erode(img, kernel, iterations=1)

# Menampilkan hasil
plt.subplot(1, 2, 1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1, 2, 2), plt.imshow(eroded, cmap='gray'), plt.title('Eroded')
plt.show()
```

CODE EXPLAIN:

- OpenCV (cv2): Digunakan untuk membaca gambar dan menerapkan operasi morfologi.
- NumPy: Digunakan untuk membuat elemen struktural yang digunakan dalam operasi morfologi.
- Matplotlib (pyplot): Digunakan untuk menampilkan gambar hasil pengolahan citra.
- img = cv2.imread('lingkaran.jpg', 0) membaca gambar lingkaran.jpg dan mengonversinya menjadi gambar grayscale. Citra grayscale akan lebih mudah diterapkan operasi morfologi.
- kernel = np.ones((5,5), np.uint8) membuat elemen struktural berbentuk persegi dengan ukuran 5×5, yang digunakan dalam operasi erosi. Elemen struktural ini mempengaruhi area piksel sekitar objek.
- eroded = cv2.erode(img, kernel, iterations=1) menerapkan operasi erosi pada citra. Erosi akan mengurangi atau mengecilkan objek dalam gambar dengan menghapus piksel dari tepi objek. Jika objeknya kecil atau tepinya tidak padat, operasi ini akan mengurangi ukuran objek lebih lanjut.
- plt.subplot(1, 2, 1) menampilkan gambar asli di sebelah kiri.
- plt.subplot(1, 2, 2) menampilkan gambar setelah diterapkan operasi erosi di sebelah kanan.
- plt.show() menampilkan kedua gambar tersebut dalam satu jendela.

OUTPUT:



4. OPENING

adalah operasi morfologi yang terdiri dari dua langkah: pertama, dilakukan erosi pada citra menggunakan elemen struktural, diikuti oleh langkah kedua yaitu dilasi pada hasil erosi. Secara keseluruhan, opening bertujuan untuk menghapus objek atau noise kecil dalam citra, dan memperhalus objek yang lebih besar.. Fungsi opening seperti berikut:

- Menghapus noise kecil atau objek yang tidak diinginkan dalam citra.
- Memisahkan objek yang berdekatan.
- Memperhalus objek yang lebih besar dan mengurangi detail-detail kecil.
- Memperbaiki citra dengan menyingkirkan objek kecil tanpa merusak struktur objek yang lebih besar.

CODE :

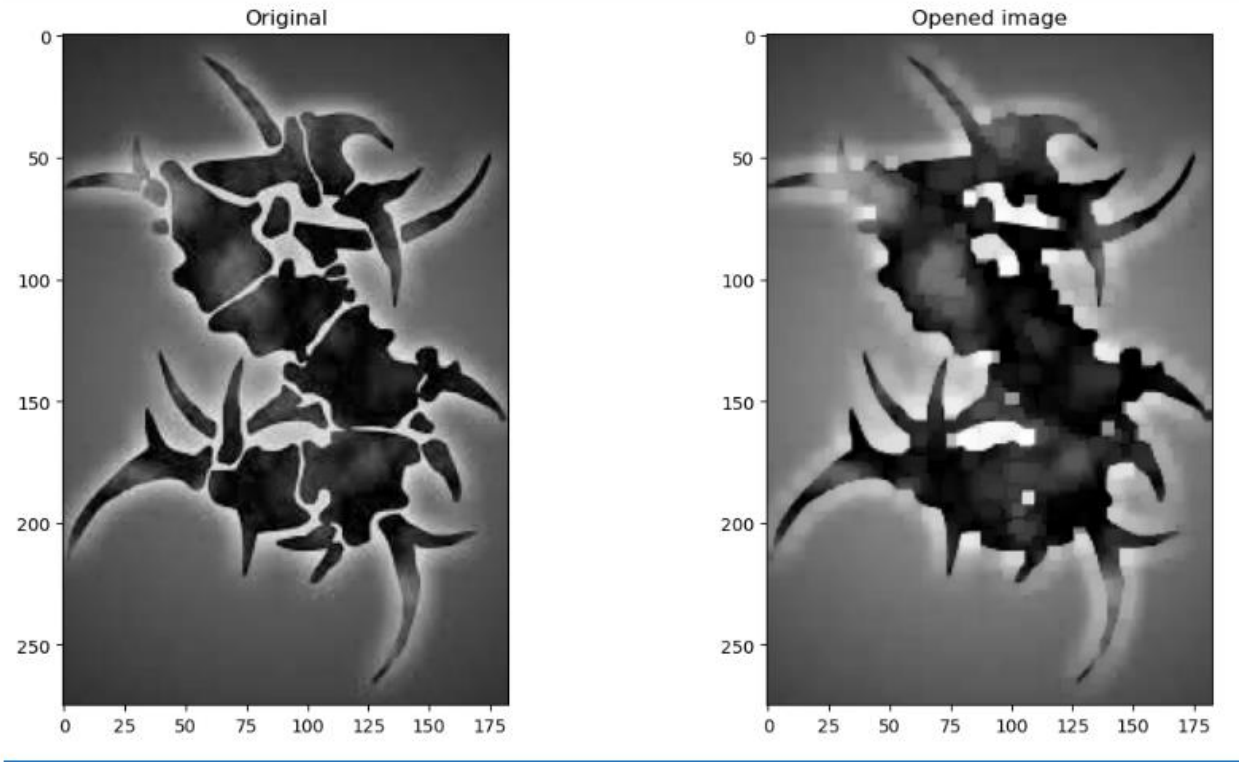
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Membaca gambar dalam mode grayscale
```

```
img = cv2.imread('lingkaran.jpg', 0)

# Membuat element structural berbentuk persegi dengan ukuran 5x5
kernel = np.ones((5,5), np.uint8)

# Melakukan operasi Opening: Erosi diikuti dilasi
opened = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)

# Menampilkan hasil
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1, 2, 2), plt.imshow(opened, cmap='gray'), plt.title('Opened image')
plt.tight_layout()
plt.show()
```

Penjelasan Script:

- cv2.imread('lingkaran.jpg', 0) membaca gambar dalam mode grayscale.
- kernel = np.ones((5,5), np.uint8) membuat elemen struktural berbentuk persegi dengan ukuran 5x5. Ini digunakan dalam operasi morfologi.
- cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel) melakukan operasi Opening, yaitu erosi diikuti oleh dilasi. Ini berguna untuk menghapus noise kecil dalam gambar.
- plt.subplot(1, 2, 1) menampilkan gambar asli.
- plt.subplot(1, 2, 2) menampilkan gambar setelah operasi Opening atau Closing diterapkan.

OUTPUT :


Original
Opened image
0
50
100
150
200
250
0
25
50
75
100
125
150
175