Tugas Kelompok Visi Komputer
IF-A Sore

Kelompok Strive

Anggota:
Albert Suhargo (211110370)
David (211110419)
Jimmy (211110476)

Link ke folder kode di OneDrive: Tugas 1-Strive
Link dataset: Dataset

Dependencies/Library pada python yang harus diinstall terlebih dahulu:

- CV2
- Matplotlib
- Numpy
- openpyxl
- pandas

**Perhitungan dilakukan dalam file excel yang dapat diakses dari OneDrive

Deskripsi Dataset: Dataset diambil dari Kaggle sebanyak 51 gambar di mana masing-masing gambar terdiri dari 1 kucing yang berbeda-beda posisi, corak, dan warnanya.

Teknik yang digunakan yaitu:

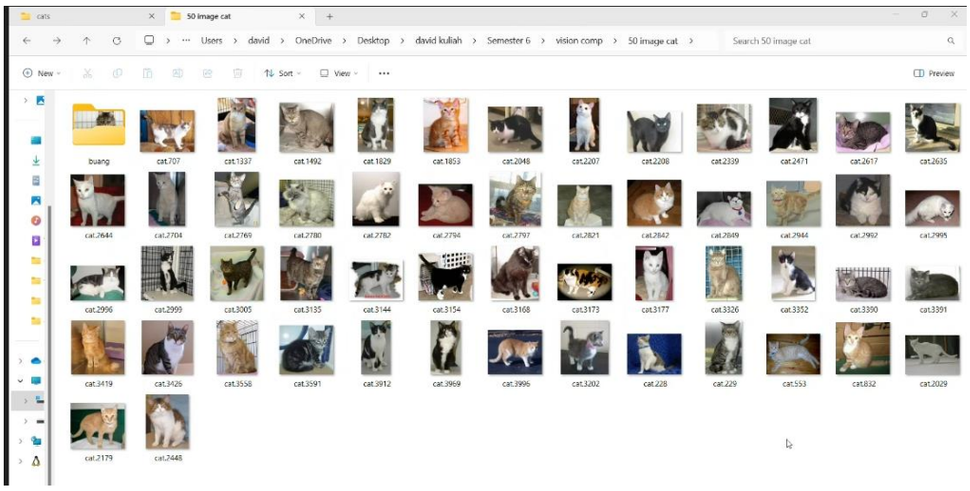
- Sharpening
- Edge Detection
- SIFT
- Gaussian Blur
- Sobel
- Prewitt
- Filling Holes
- Dilasi, Erosi, Opening, dan Closing

Hasil eksperimen dilampirkan di bagian 5 yang meliputi:

- Membandingkan hasil dari sharpening dengan ImageJ dan kode manual yang hasilnya lebih kurang sama
- Perbandingan gambar menggunakan SIFT pada image kucing dengan satu kucing dan satu anjing menghasilkan persentase kemiripan untuk kucing 99.57% dan untuk anjing 56.89%
- Penggunaan Fill Holes pada dataset mungkin kurang efektif karena kebanyakan gambar hanya terfokus pada satu gambar di mana gambar tersebut tidak memiliki banyak lubang kecil
- Threshold sangat berefek ke gambar yang ingin dijadikan binary, misalnya threshold yang terlalu tinggi membuat objek kucing tubuhnya semakin hilang atau menyatu ke background (hitam) sebaliknya jika nilai threshold terlalu rendah, maka objek kucing semakin tidak terdeteksi sama sekali dan menyisakan bercak-bercak hitam yang tidak beraturan dan menyatu ke background (putih)
- Permasalahan utamanya yaitu di bagian pendeteksian warna putih atau warna yang tidak terdapat pada gambar utama yang dijadikan pembanding sehingga sebagian pendeteksian tubuh kucingnya menyatu ke background.
- Segmentasi menggunakan erode lebih cocok dalam kasus kami karena memperjelas bagian mana yang kepala dan badan kucing karena mempersempit area gambar secara keseluruhan
- Hasil dari edge detection lumayan karena terlihat outline dari tubuh kucing berwarna putih yang memisahkannya dari background
- Matching menggunakan SIFT dari suatu gambar yang dijadikan pembanding utama terhadap 50 image lainnya dalam dataset yang kami ambil sebagian besar berhasil mendeteksi bahwa gambar tersebut memiliki kucing di dalamnya
- Kernel yang digunakan sangat berefek ke hasil gambar yang diproses

1. Data Computer Vision

a. Memilih 51 citra dari dataset



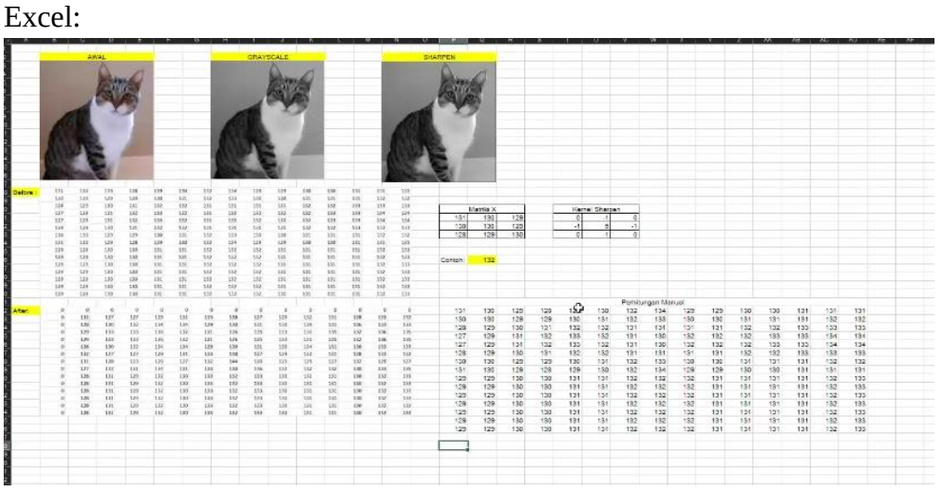
b. Dilakukan Penajaman Citra Menggunakan Code berikut:

- Sharpening

```
import cv2
import numpy as np
import pandas as pd
from openpyxl import Workbook
from openpyxl.drawing.image import Image as ExcelImage

def sharpening_with_border_handling(image, kernel):
    height, width = image.shape[:2]
    k_height, k_width = kernel.shape[:2]

    output = np.zeros((height, width), dtype=np.uint8)
    border_x = k_width // 2
    border_y = k_height // 2

    for y in range(border_y, height - border_y):
        for x in range(border_x, width - border_x):
            roi = image[y - border_y:y + border_y + 1, x - border_x:x +
border_x + 1]
            conv_result = np.sum(roi * kernel)
            output[y, x] = np.clip(conv_result, 0, 255)

    return output

image_sharpen_3x3 = np.array([
    [0,-1,0],
    [-1,5,-1],
    [0,-1,0]
])

image = cv2.imread('Soal 1\\dataset\\cat.png', cv2.IMREAD_GRAYSCALE)

if image is not None:
    contrast_enhanced_image = sharpening_with_border_handling(image,
image_sharpen_3x3)
    cv2.imwrite('grayscale.png', image)
    print("Gambar grayscale berhasil disimpan sebagai 'grayscale.png'.")
```

```
    cv2.imwrite('download_contrast_enhanced.png', contrast_enhanced_image)
    print("Gambar berhasil ditingkatkan kontras dan disimpan sebagai
'download_contrast_enhanced.png'.")

    wb = Workbook()
    ws = wb.active

    img = ExcelImage('download_contrast_enhanced.png')
    ws.add_image(img, 'A1')

    wb.save('hasil.xlsx')
    print("Hasil gambar telah diekspor ke dalam Excel sebagai 'hasil.xlsx'.")
    result = sharpening_with_border_handling(image, image_sharpen_3x3)

    print("\nMatriks hasil sharpening:")
    print(result.astype(np.uint8))

    df_before = pd.DataFrame(image)
    df_after = pd.DataFrame(result)

    with pd.ExcelWriter('output.xlsx') as writer:
        df_before.to_excel(writer, sheet_name='Before sharpening',
index=False, header=False)
        df_after.to_excel(writer, sheet_name='After sharpening', index=False,
header=False)
else:
    print("Gagal membaca gambar 'cat.png'. Pastikan gambar tersedia di path
yang benar.")
```

c. Membuat Proses Perhitungan 15x15, dan memasukkan gambar hasil ke dalam Excel:



2. Technique Computer Vision

a. Melakukan Edge Detection Dengan Code Berikut:

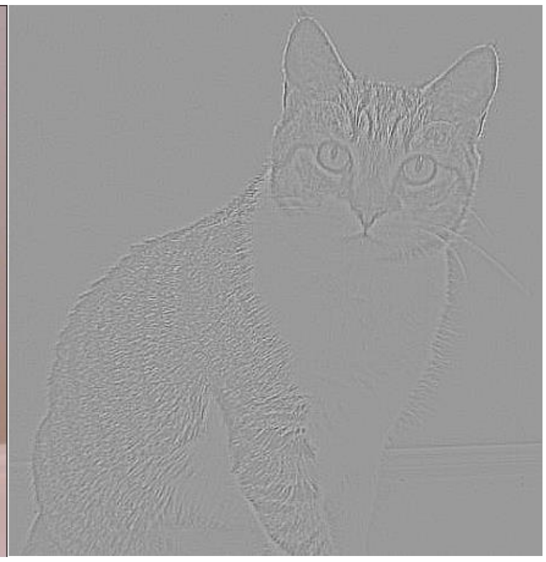
- Edge Detection

```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def convolution_with_border_handling(image, kernel):
    height, width = image.shape[:2]
    k_height, k_width = kernel.shape[:2]

    output = np.zeros((height, width))

    border_x = k_width // 2
    border_y = k_height // 2

    for y in range(border_y, height - border_y):
        for x in range(border_x, width - border_x):
            roi = image[y - border_y:y + border_y + 1, x - border_x:x +
border_x + 1]
            conv_result = np.sum(roi * kernel)
            output[y, x] = min(conv_result, 255)

    return output

edge_detection_3x3 = np.array([
   [-1,-1,-1],
   [-1,8,-1],
   [-1,-1,-1]
])

image = cv2.imread('C:\\Users\\frynn\\OneDrive\\Desktop\\Kode Comp Vis\\Soal
2\\cat.png', cv2.IMREAD_GRAYSCALE)

if image is None:
    print("Gagal membaca gambar!")
else:
    print("\nMatriks gambar sebelum edge detection:")
    print(image)

    result = convolution_with_border_handling(image, edge_detection_3x3)

    print("\nMatriks hasil edge detection:")
    print(result.astype(np.uint8))

    df_before = pd.DataFrame(image)
    df_after = pd.DataFrame(result)

    with pd.ExcelWriter('output.xlsx') as writer:
        df_before.to_excel(writer, sheet_name='Before Edge Detection',
index=False, header=False)
        df_after.to_excel(writer, sheet_name='After Edge Detection',
index=False, header=False)
```

```
    plt.imshow(result, cmap='gray')
    plt.title('Hasil Edge Detection')
    plt.axis('off')
    plt.show()
```

b. Hasil Edge Detection:

Before



After

c. Perhitungan Edge Detection dalam Excel:

131	130	129	128	129	130	132	134	129	129	130	130	131	131	131
130	130	129	129	130	131	132	133	130	130	131	131	131	132	132
128	129	130	131	132	132	131	131	131	131	132	132	133	133	133
127	129	131	132	133	132	131	130	132	132	132	133	133	134	134
127	129	131	132	133	132	131	130	132	132	132	133	133	134	134
128	129	130	131	132	132	131	131	131	131	132	132	133	133	133
130	130	129	129	130	131	132	133	130	130	131	131	131	132	132
131	130	129	128	129	130	132	134	129	129	130	130	131	131	131
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133
129	129	130	130	131	131	132	132	132	131	131	131	131	132	133

Kernel Edge Detection		
-1	-1	-1
-1	8	-1
-1	-1	-1

Hasil Edge Detection														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	-4	-6	-2	0	2	14	-8	-3	3	-2	-5	1	0
0	-2	0	2	6	4	-4	-2	-1	-2	4	0	5	2	0
0	0	5	3	8	1	-1	-9	7	2	-1	4	-1	5	0
0	0	5	3	8	1	-1	-9	7	2	-1	4	-1	5	0
0	-2	0	2	6	4	-4	-2	-1	-2	4	0	5	2	0
0	4	-4	-6	-2	0	2	14	-8	-3	3	-2	-5	1	0
0	3	-3	-13	-8	-8	1	20	-19	-12	-4	-7	-1	-5	0
0	-5	5	2	8	0	2	1	6	3	4	2	-1	3	0
0	-3	3	-3	3	-3	3	0	3	-3	0	0	-3	0	0
0	-3	3	-3	3	-3	3	0	3	-3	0	0	-3	0	0
0	-3	3	-3	3	-3	3	0	3	-3	0	0	-3	0	0
0	-3	3	-3	3	-3	3	0	3	-3	0	0	-3	0	0
0	-3	3	-3	3	-3	3	0	3	-3	0	0	-3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

d. Menggunakan Teknik SIFT untuk mendapatkan titik titik penting dengan code berikut:

```
import cv2
import pandas as pd

# Load gambar
image = cv2.imread('C:\\Users\\frynn\\OneDrive\\Desktop\\Kode Comp Vis\\Soal 4\\cat.png')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Inisialisasi detektor SIFT
sift = cv2.SIFT_create()

# Temukan titik kunci dan deskripsinya
keypoints, descriptors = sift.detectAndCompute(gray, None)

# Membuat DataFrame untuk menyimpan posisi titik kunci
keypoints_data = []
for kp in keypoints:
    keypoints_data.append([kp.pt[0], kp.pt[1]])

df_keypoints = pd.DataFrame(keypoints_data, columns=['X', 'Y'])

# Simpan DataFrame ke dalam file Excel
df_keypoints.to_excel('keypoints_data.xlsx', index=False)

print("Data titik kunci telah disimpan ke dalam 'keypoints_data.xlsx'.")

# Gambar titik kunci pada gambar input
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)

# Tampilkan gambar dengan titik kunci
cv2.imshow("Image with Keypoints", image_with_keypoints)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

e. Hasil SIFT Detection:

f. Hasil SIFT Detection dalam Excel:

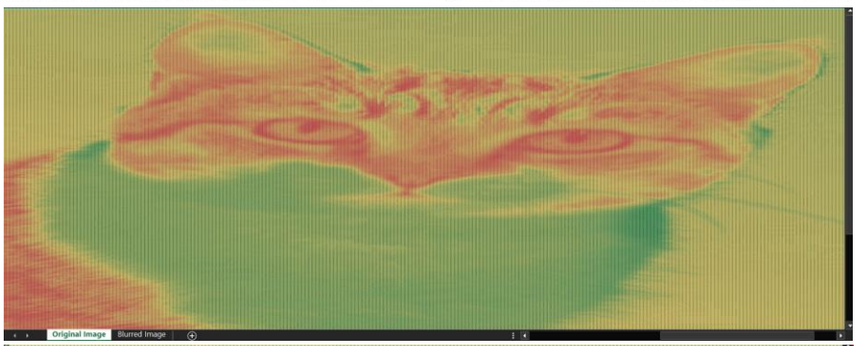
X	Y
20,2485	429,837
22,6249	417,104
25,143	410,943
25,4783	459,653
26,0982	445,194
26,5251	431,759
27,3378	473,009
28,985	453,586
29,0811	365,73
29,687	319,024
30,1219	348,193
32,1618	407,426
32,2145	383,935
32,6009	413,745
33,978	413,257
33,978	413,257
37,1378	301,229
38,8158	338,734
38,9531	360,012
40,2562	303,811
40,9309	458,702
41,4338	324,525
42,3845	304,53
45,0908	308,884
45,1954	283,662
46,2056	444,103
46,4455	293,066
47,557	296,605
52,2214	300,476
52,2634	389,052
53,7201	409,475
53,7201	409,475
57,8745	306,064
58,932	281,393

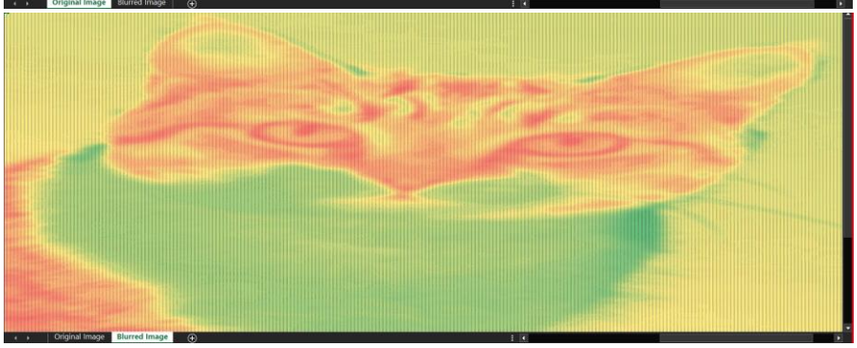
3. Konvolusi

a. Menggunakan Teknik Konvolusi Gaussian Blur untuk menghilangkan noise:

```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def apply_gaussian_blur(image, kernel_size=(5, 5), sigma_x=0):
    blurred_image = cv2.GaussianBlur(image, kernel_size, sigma_x)
    return blurred_image

# Load gambar
image = cv2.imread('C:\\Users\\frynn\\OneDrive\\Desktop\\Kode Comp Vis\\Soal
3\\cat.png')

# Cek apakah gambar berhasil dimuat
if image is None:
    print("Gagal membaca gambar!")
else:
    # Konversi gambar ke grayscale jika perlu
    if len(image.shape) > 2:
        grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        grayscale_image = image

    # Terapkan Gaussian blur
    blurred_image = apply_gaussian_blur(grayscale_image)

     # Simpan hasil perhitungan ke dalam file Excel
    df_original = pd.DataFrame(grayscale_image)
    df_blurred = pd.DataFrame(blurred_image)

    with pd.ExcelWriter('gaussian_blur_output.xlsx') as writer:
        df_original.to_excel(writer, sheet_name='Original Image',
index=False, header=False)
        df_blurred.to_excel(writer, sheet_name='Blurred Image', index=False,
header=False)

    print("Hasil perhitungan telah disimpan ke dalam file
'gaussian_blur_output.xlsx'.")

    # Tampilkan gambar asli dan hasil blur
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(grayscale_image, cmap='gray')
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(blurred_image, cmap='gray')
    plt.title('Blurred Image')
    plt.axis('off')

    plt.show()
```

b. Hasil Gaussian Blur:



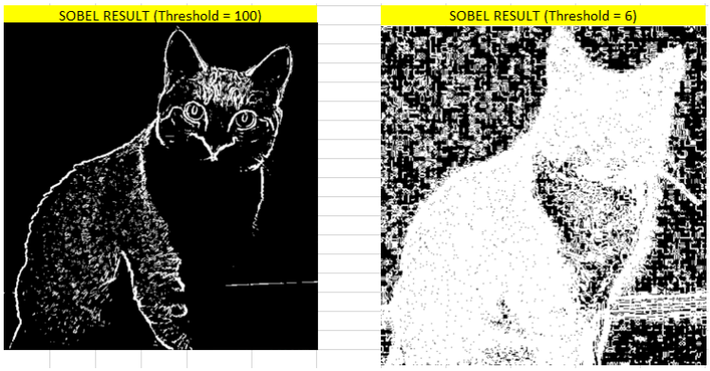


c. Menggunakan kernel Sobel untuk meningkatkan citra:

```python
import cv2
import numpy as np

# Read the image
image = cv2.imread('Soal 3\\cat.png', cv2.IMREAD_GRAYSCALE)

# Apply Sobel operator
sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)

# Compute the gradient magnitude
gradient_magnitude = np.sqrt(sobelx**2 + sobely**2)

# Set a threshold to binarize the gradient magnitude image
threshold = 6
edges = np.uint8(gradient_magnitude > threshold) * 255

# Display the result
cv2.imshow('Original Image', image)
cv2.imshow('Edges (Sobel)', edges)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

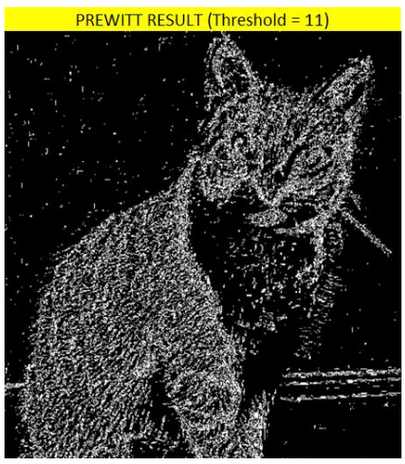
d. Hasil Sobel:



e. Menggunakan kernel Prewit untuk meningkatkan citra:

```
import cv2
import numpy as np

image = cv2.imread('Soal 3\\cat.png', cv2.IMREAD_GRAYSCALE)

kernel_x = np.array([[-1, 0, 1],
                     [-1, 0, 1],
                     [-1, 0, 1]])

kernel_y = np.array([[-1, -1, -1],
                     [ 0,  0,  0],
                     [ 1,  1,  1]])

prewittx = cv2.filter2D(image, -1, kernel_x)
prewitty = cv2.filter2D(image, -1, kernel_y)

gradient_magnitude = np.sqrt(prewittx**2 + prewitty**2)

threshold = 11
edges = np.uint8(gradient_magnitude > threshold) * 255

cv2.imshow('Original Image', image)
cv2.imshow('Edges (Prewitt)', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

f. Hasil Prewit:



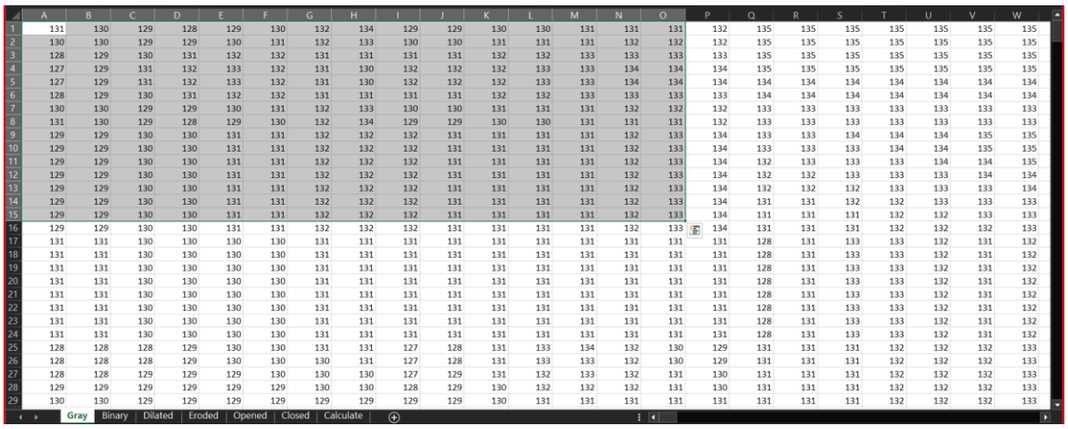
4. Morfologi

a. Melakukan proses Morfologi (Dilasi, Erosi, Opening, dan Closing):

```
import numpy as np
import cv2
import pandas as pd

def grayscale(image):
    img = cv2.imread(image)
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, binary_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY)
    return gray_img, binary_img

def dilate(image, kernel):
    result = cv2.dilate(image, kernel, iterations=1)
    return result

def erode(image, kernel):
    result = cv2.erode(image, kernel, iterations=1)
    return result

def opening(image, kernel):
    result = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
    return result

def closing(image, kernel):
    result = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
    return result

def convert_to_binary(image):
    return np.where(image == 255, 1, 0)

img_path = 'Soal 4\\cat.png'
gray_img, binary_img = grayscale(img_path)

kernel_size = (3, 3)
kernel = np.ones(kernel_size, dtype=np.uint8)

# Process the images
dilated_img = dilate(binary_img, kernel)
eroded_img = erode(binary_img, kernel)
opened_img = opening(binary_img, kernel)
closed_img = closing(binary_img, kernel)

# Convert images to binary
binary_img_binary = convert_to_binary(binary_img)
dilated_img_binary = convert_to_binary(dilated_img)
eroded_img_binary = convert_to_binary(eroded_img)
opened_img_binary = convert_to_binary(opened_img)
closed_img_binary = convert_to_binary(closed_img)

# Convert binary arrays to DataFrame
df_gray = pd.DataFrame(gray_img)
df_binary = pd.DataFrame(binary_img_binary)
df_dilated = pd.DataFrame(dilated_img_binary)
df_eroded = pd.DataFrame(eroded_img_binary)
```

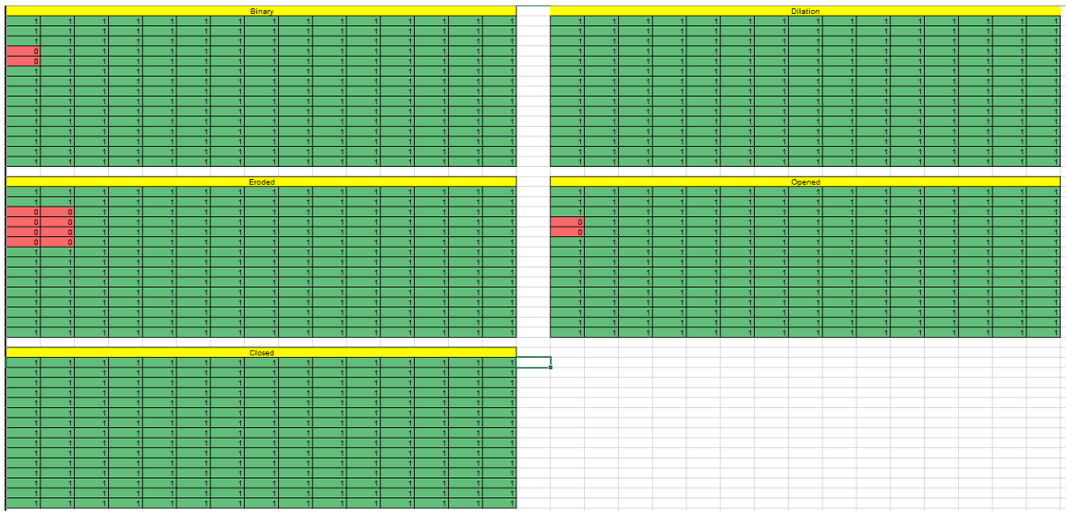
```
df_opened = pd.DataFrame(opened_img_binary)
df_closed = pd.DataFrame(closed_img_binary)

# Write DataFrames to Excel file
with pd.ExcelWriter('segmentation.xlsx') as writer:
    df_gray.to_excel(writer, sheet_name='Gray', index=False, header=False)
    df_binary.to_excel(writer, sheet_name='Binary', index=False,
header=False)
    df_dilated.to_excel(writer, sheet_name='Dilated', index=False,
header=False)
    df_eroded.to_excel(writer, sheet_name='Eroded', index=False,
header=False)
    df_opened.to_excel(writer, sheet_name='Opened', index=False,
header=False)
    df_closed.to_excel(writer, sheet_name='Closed', index=False,
header=False)

print("Excel file 'segmentation.xlsx' saved with binary calculations and
original grayscale values.")
```

b. Hasil Morfologi Opening:



Grayscale (Angka awal kiri atas)

Hasil kalkulasi Dilation, Eroded, Opened, Closed

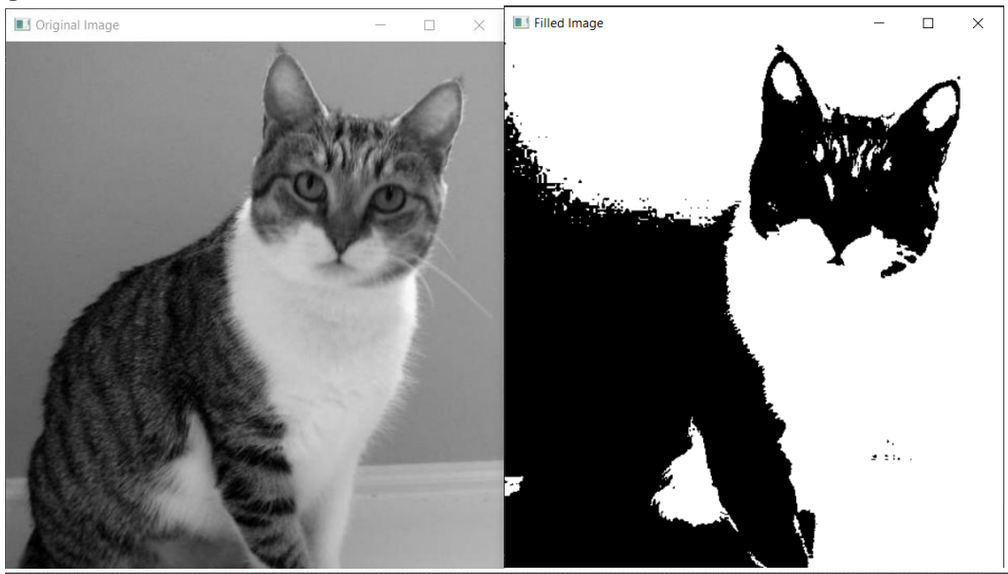


c. Melakukan Teknik Filling Holes untuk menghilangkan lubang lubang kecil pada citra

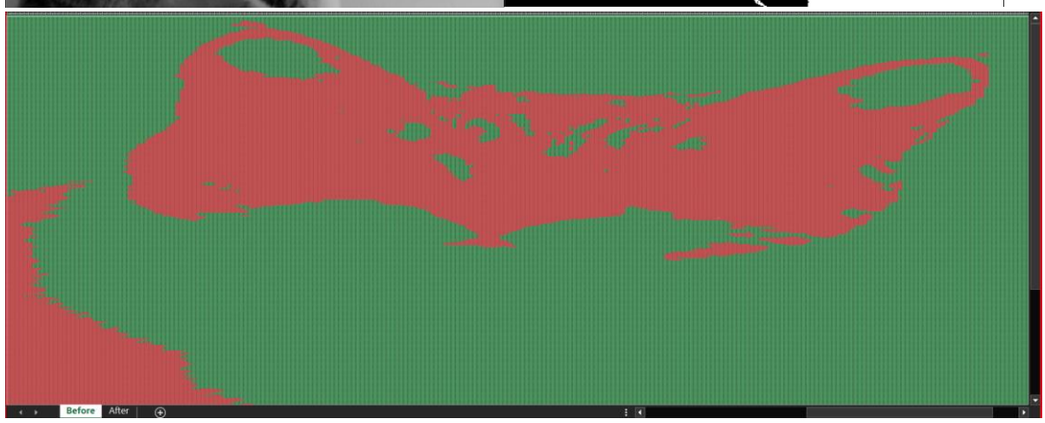
```
import cv2
import numpy as np
import pandas as pd

def fill_holes(binary_image):
    # Inversi citra biner (putih menjadi hitam dan sebaliknya)
    inverted_image = cv2.bitwise_not(binary_image)

    # Temukan kontur dari citra terinversi
    contours, _ = cv2.findContours(inverted_image, cv2.RETR_CCOMP,
cv2.CHAIN_APPROX_SIMPLE)

    # Jika tidak ada kontur yang ditemukan, kembalikan citra biner asli
    if not contours:
        return binary_image

    # Buat citra kosong untuk menyimpan area yang diisi (kontur diisi)
    filled_image = np.zeros_like(binary_image)

    # Gambar ulang semua kontur yang ditemukan untuk mengisi area dalam citra
    cv2.drawContours(filled_image, contours, -1, 255, thickness=cv2.FILLED)

    # Inversi kembali citra untuk mendapatkan hasil pengisian lubang
    filled_image = cv2.bitwise_not(filled_image)

    return filled_image

# Baca citra (pastikan citra adalah citra biner, misalnya hasil dari dilasi
dan erosi)
input_image = cv2.imread('Soal 4\\cat.png', cv2.IMREAD_GRAYSCALE)

# Lakukan thresholding untuk mengubah citra ke citra biner
_, binary_image = cv2.threshold(input_image, 127, 255, cv2.THRESH_BINARY)

# Lakukan pengisian lubang (filling holes)
filled_image = fill_holes(binary_image)
```

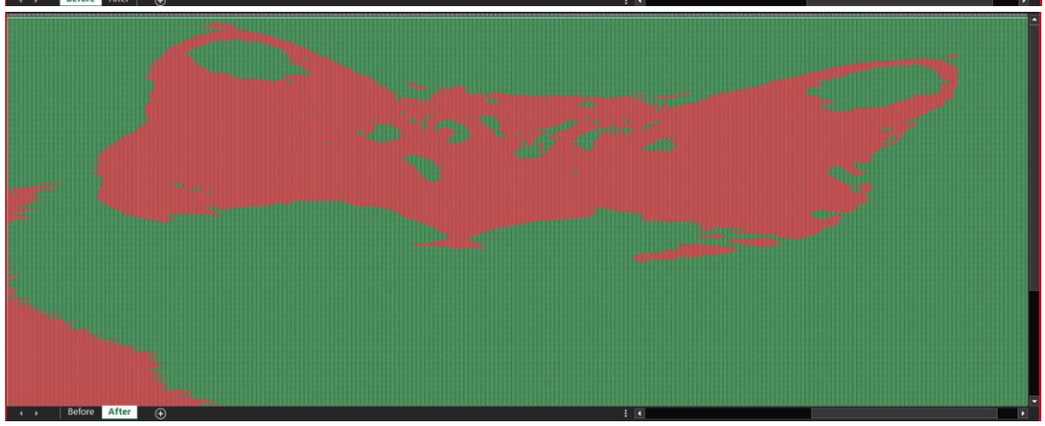
```
# Convert images to DataFrame
df_before = pd.DataFrame(binary_image)
df_after = pd.DataFrame(filled_image)

# Write DataFrames to Excel file
with pd.ExcelWriter('fill_holes.xlsx') as writer:
    df_before.to_excel(writer, sheet_name='Before', index=False,
header=False)
    df_after.to_excel(writer, sheet_name='After', index=False, header=False)

print("Excel file 'fill_holes.xlsx' saved with before and after images.")
```

Hasil Filling Holes:



Threshold 127





5. Deteksi Fitur dan Pencocokan

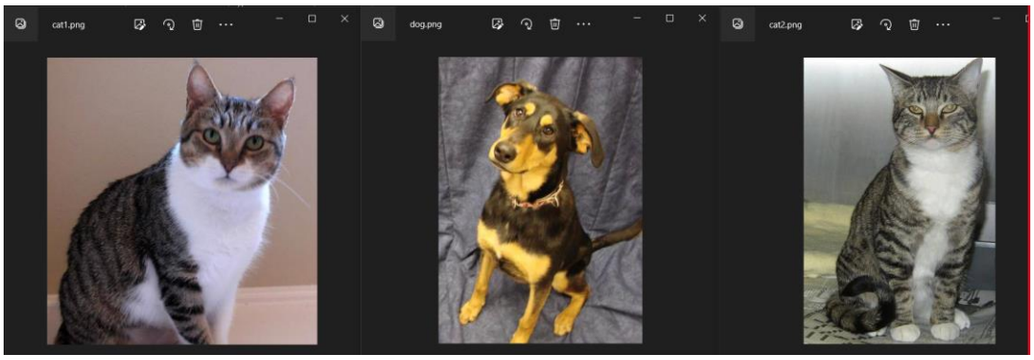
a. Menggunakan SIFT untuk melakukan perbandingan antara Citra yang satu dengan yang lain

```
import cv2
import numpy as np
import pandas as pd
from sklearn.metrics import precision_score, recall_score, f1_score

def compare_images(reference_image, image_list):
    similarities = []
    precision_scores = []
    recall_scores = []
    f1_scores = []

    # Load reference image and extract SIFT features
    gray_ref = cv2.cvtColor(reference_image, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create()
    keypoints_ref, descriptors_ref = sift.detectAndCompute(gray_ref,
None)

    keypoints_data_all = {'X': [], 'Y': [], 'Image': []}

    # Prepare a list to hold images with keypoints drawn
    images_with_keypoints = []

    # Process reference image
    image_with_keypoints = cv2.drawKeypoints(reference_image,
keypoints_ref, None)
    images_with_keypoints.append(image_with_keypoints)

    for idx, image in enumerate(image_list):
        # Load the current image
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # Extract SIFT features for the current image
        keypoints, descriptors = sift.detectAndCompute(gray, None)

        # Perform feature matching using descriptors
        bf = cv2.BFMatcher()
        matches = bf.knnMatch(descriptors_ref, descriptors, k=2)

        # Filter matches using Lowe's ratio test
        good_matches = []
        for m, n in matches:
            if m.distance < 0.75 * n.distance:
                good_matches.append(m)

        # Calculate precision, recall, and F1-score
        matches_mask = np.zeros(len(good_matches))
        for i, match in enumerate(good_matches):
            matches_mask[i] = 1 if match.distance < 0.75 * n.distance
else 0

        precision = precision_score(np.ones(len(matches_mask)),
matches_mask)
```

```
        recall = recall_score(np.ones(len(matches_mask)), matches_mask)
        f1 = f1_score(np.ones(len(matches_mask)), matches_mask)

        precision_scores.append(precision)
        recall_scores.append(recall)
        f1_scores.append(f1)

        # Calculate similarity percentage
        similarity_percentage = len(good_matches) / len(keypoints_ref)
* 100
        similarities.append(similarity_percentage)

        # Save SIFT keypoints of the current image
        for kp in keypoints:
            keypoints_data_all['X'].append(kp.pt[0])
            keypoints_data_all['Y'].append(kp.pt[1])
            keypoints_data_all['Image'].append(f'Image{idx + 1}')

        # Draw SIFT keypoints on the current image
        image_with_keypoints = cv2.drawKeypoints(image, keypoints,
None)
        images_with_keypoints.append(image_with_keypoints)

    # Display all images with keypoints drawn
    for idx, img_with_keypoints in enumerate(images_with_keypoints):
        cv2.imshow(f"Image{idx + 1} with Keypoints",
img_with_keypoints)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # Save SIFT keypoints of all images to Excel
    df_keypoints_all = pd.DataFrame(keypoints_data_all)
    with pd.ExcelWriter('keypoints_all_images.xlsx') as writer:
        for idx, image_name in
enumerate(df_keypoints_all['Image'].unique()):
            df_keypoints_all[df_keypoints_all['Image'] ==
image_name].to_excel(writer, sheet_name=f'Image{idx + 1}', index=False)

    # Save metrics to Excel file
    data = {'Image': [f'Image{i + 1}' for i in range(len(image_list) +
1)],
            'Similarity (%)': similarities,
            'Precision': precision_scores,
            'Recall': recall_scores,
            'F1-score': f1_scores}
    df_metrics = pd.DataFrame(data)
    df_metrics.to_excel('image_metrics.xlsx', index=False)

# Load reference image
reference_image = cv2.imread('Soal 5\\cat1.png')

# Load other images to compare with the reference image
image2 = cv2.imread('Soal 5\\cat2.png')
image3 = cv2.imread('Soal 5\\dog.png')
```

```
# Compare images
compare_images(reference_image, [image2, image3])
```

b. Hasil Pencocokan



Titik-titik hasil pendeteksian SIFT

X	Y	Image
2,338377	439,5488	Image1
3,519588	415,4006	Image1
4,715528	482,4943	Image1
4,992142	470,3138	Image1
4,992142	470,3138	Image1
5,720012	489,5215	Image1
6,277241	477,3922	Image1
7,644722	444,6082	Image1
7,896303	485,6238	Image1
7,896303	485,6238	Image1
8,269421	404,0545	Image1
8,603991	448,7083	Image1
8,603991	448,7083	Image1
8,632886	479,1611	Image1
8,932993	447,0649	Image1
9,0334	407,1591	Image1
10,09299	437,5269	Image1
10,09299	437,5269	Image1
11,39378	462,4961	Image1

X	Y	Image
2,488863	69,17744	Image2
2,660163	42,10822	Image2
2,793549	61,87799	Image2
2,842324	57,70458	Image2
3,00267	49,43956	Image2
3,066212	100,2168	Image2
3,151961	147,7722	Image2
3,371328	107,3321	Image2
3,586629	113,1228	Image2
4,293559	150,5435	Image2
4,33252	120,6029	Image2
5,870138	52,65678	Image2
5,871977	94,34959	Image2
6,045863	84,1978	Image2
6,420413	131,9765	Image2
6,755392	177,1763	Image2
6,834649	33,49517	Image2
6,875071	148,4453	Image2
6,875071	148,4453	Image2

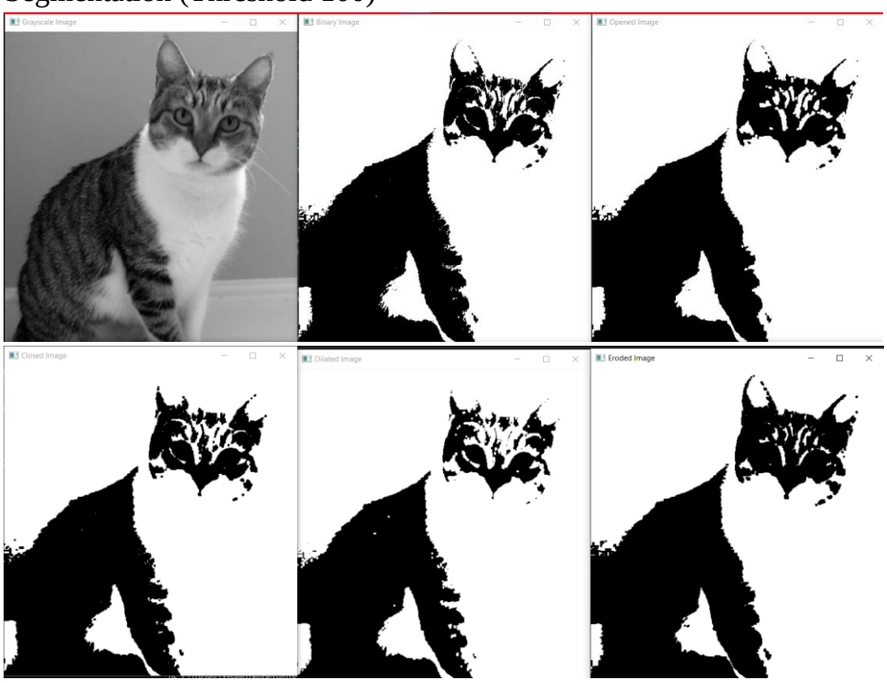
Image	Similarity (%)	Precision	Recall	F1-score		
Image1	0,99573257	1	0,71429	0,83333	->	Gambar Kucing Sejenis
Image2	0,56899004	1	0,5	0,66667	->	Gambar Anjing

Hasil Uji Coba pada beberapa Gambar:

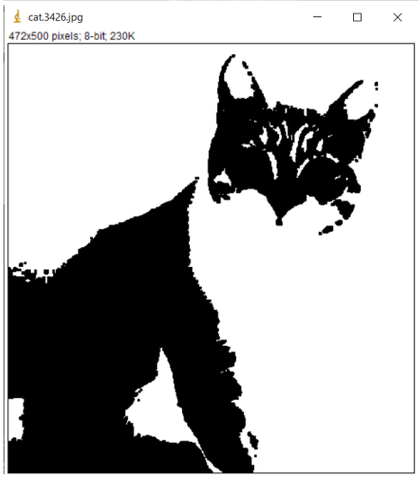
- Fill Holes (Threshold 100)

Fill Holes untuk bagian muka dan sebagian tubuh sudah bagus, tetapi tubuhnya yang berwarna putih menyatu sebagian ke background

- Segmentation (Threshold 100)



Hasil binary, opening, closing, dilation, dan erode menggunakan kode manual

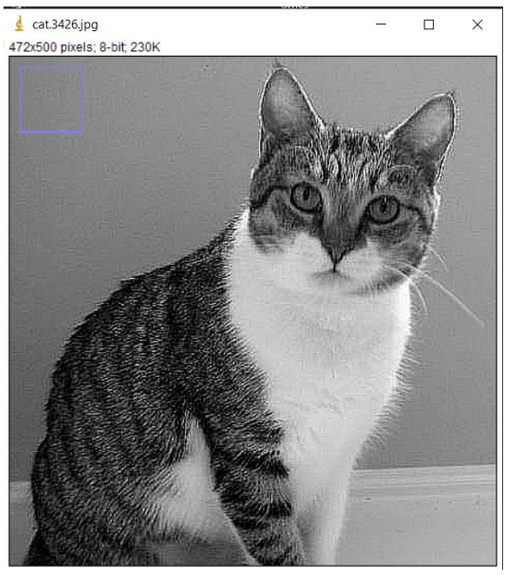


Hasil dilation menggunakan ImageJ

Untuk segmentation, karena warna kucingnya ada yang putih secara default, maka algoritma erode lebih bagus untuk menyempitkan area yang berdekatan sehingga membuat keseluruhan gambar lebih menyatu sebagai contoh, telinga kucing semakin terlihat dan muka kucing semakin jelas.

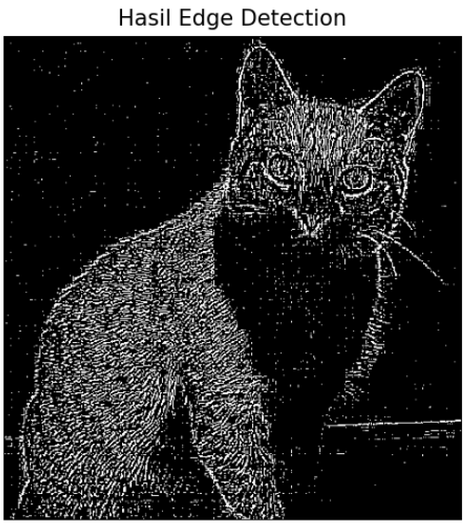
- Sharpening (Kernel High Pass)

Gambarnya sangat bagus kalau tidak terlalu pixelated. Sepertinya banyak noise pada gambar. Hasil dari kode manual

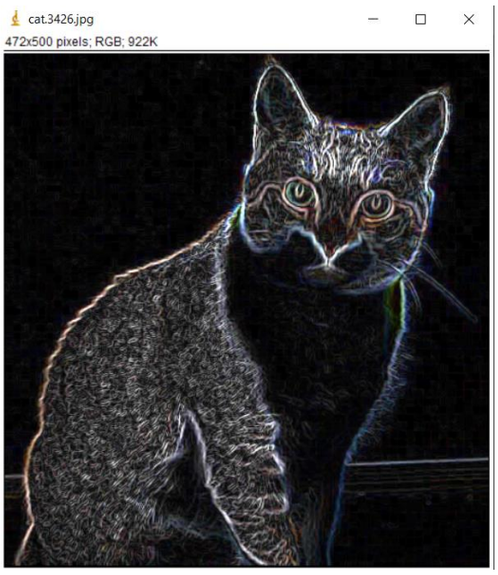


Hasil sharpening ImageJ

- Edge Detection (Threshold = 160)



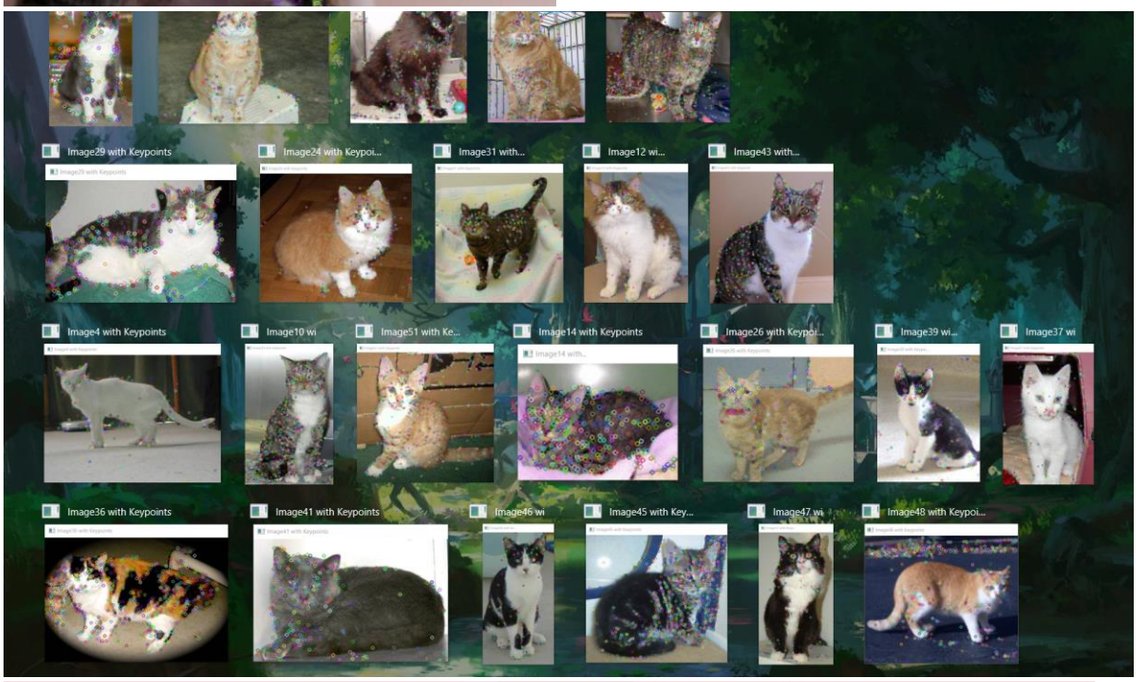
Hasil menggunakan kode manual dengan kernel
[-1,-1,-1],
[-1,8,-1],
[-1,-1,-1]



Hasil edge detection menggunakan ImageJ

- Matching dengan SIFT

<- Gambar Utama



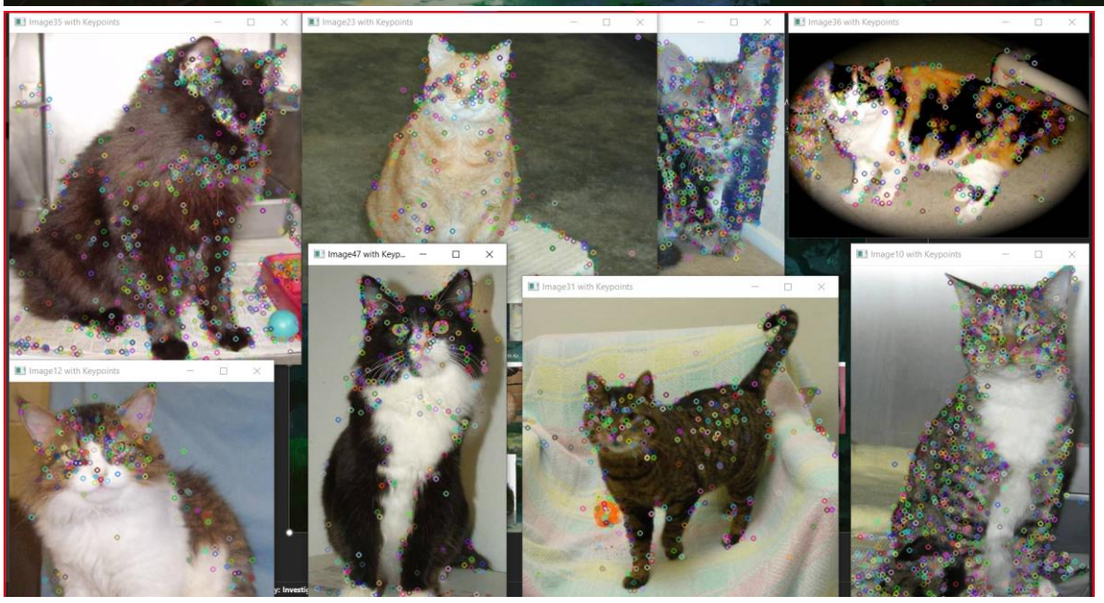
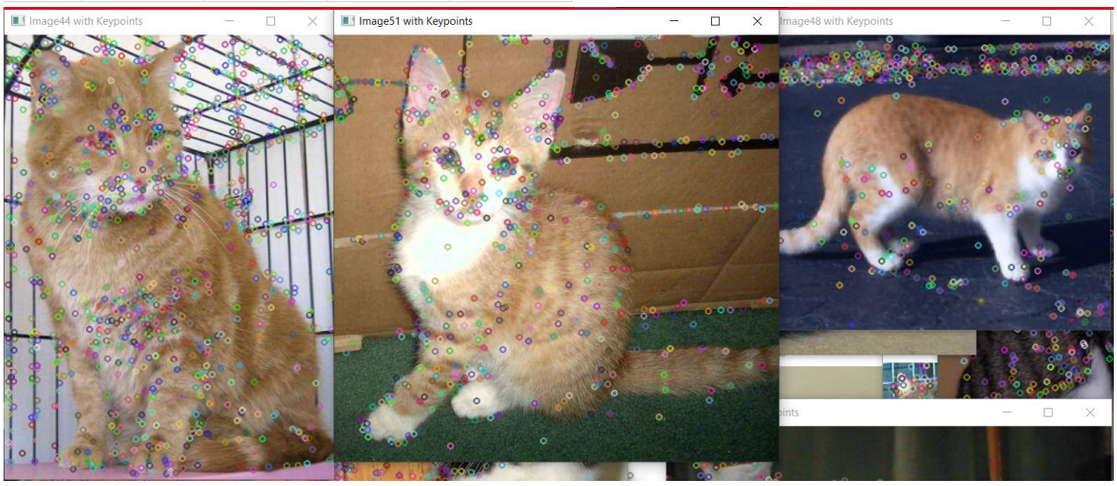


Image	Similarity (%)	Precision	Recall	F1-score
Image1	28%	0	0	0
Image2	228%	1	0,8125	0,896551724
Image3	142%	1	0,5	0,666666667
Image4	128%	1	0,333333333	0,5
Image5	228%	1	0,5	0,666666667
Image6	71%	0	0	0
Image7	85%	1	0,5	0,666666667
Image8	213%	1	0,466666667	0,636363636
Image9	142%	1	0,5	0,666666667
Image10	100%	1	0,714285714	0,833333333
Image11	85%	1	0,166666667	0,285714286
Image12	71%	1	0,4	0,571428571
Image13	213%	1	0,733333333	0,846153846
Image14	57%	0	0	0
Image15	114%	1	0,5	0,666666667
Image16	185%	1	0,538461538	0,7
Image17	85%	1	0,166666667	0,285714286
Image18	43%	1	0,666666667	0,8
Image19	213%	1	0,8	0,888888889
Image20	71%	1	0,8	0,888888889
Image21	85%	1	0,166666667	0,285714286
Image22	85%	1	0,333333333	0,5
Image23	114%	1	0,5	0,666666667
Image24	71%	1	0,4	0,571428571
Image25	128%	1	0,444444444	0,615384615
Image26	100%	1	0,142857143	0,25
Image27	171%	1	0,583333333	0,736842105
Image28	100%	1	0,428571429	0,6
Image28	100%	1	0,428571429	0,6
Image29	100%	1	0,714285714	0,833333333
Image30	114%	1	0,125	0,222222222
Image31	100%	1	0,285714286	0,444444444
Image32	57%	1	0,25	0,4
Image33	171%	1	0,333333333	0,5
Image34	142%	1	0,3	0,461538462
Image35	28%	1	0,5	0,666666667
Image36	57%	1	1	1
Image37	142%	1	0,5	0,666666667
Image38	57%	1	0,25	0,4
Image39	213%	1	0,466666667	0,636363636
Image40	71%	0	0	0
Image41	43%	1	0,666666667	0,8
Image42	142%	1	0,2	0,333333333
Image43	10000%	1	1	1
Image44	284%	1	0,5	0,666666667
Image45	71%	0	0	0
Image46	213%	1	0,4	0,571428571
Image47	114%	1	0,5	0,666666667
Image48	128%	1	0,333333333	0,5
Image49	28%	0	0	0
Image50	14%	0	0	0
Image51	28%	1	0,5	0,666666667

Hasil melebihi 100% dikarenakan jumlah titik yang ditemukan melebihi jumlah titik awal yang dibandingkan atau bug

Hasil 10000% similarity disebabkan itu gambar yang dijadikan patokan dalam perbandingan antar gambar



Hasil di atas merupakan beberapa pendeteksian yang kurang akurat

Kesimpulan

Menggunakan tools seperti ImageJ dengan program yang dibuat secara manual hasilnya tidak jauh berbeda. Penggunaan teknik Sobel lumayan efektif dalam edge detection dimana bisa kita lihat object kucingnya terseleksi dengan cukup baik. Angka threshold sangat berpengaruh pada hasil binary yang ingin didapat. Angka threshold yang baik tidak selalu ditengah (nilai tengah 255/2 = 127), misalnya pada citra kami, kami menggunakan threshold 160 dan hasilnya objek kucingnya terlihat lebih baik, ketimbang ketika kami menggunakan threshold 127 yang menyebabkan sebagian bagian tubuh kucing menyatu dengan background. Penggunaan teknik SIFT untuk membandingkan dua gambar hasilnya sangat baik dimana ketika kami memasukkan kucing yang warnanya mirip dengan kucing yang dijadikan gambar pembanding utama, dan nilai kemiripannya mencapai 99,5% sedangkan ketika dibandingkan dengan gambar lain yaitu anjing, tingkat kemiripannya hanya di angka 56,8%. Namun juga perlu diperhatikan, dalam penggunaan SIFT ini dapat menghasilkan angka persentase kemiripan diatas 100 ketika titik titik pada gambar yang ingin dicek lebih banyak daripada titik titik pada gambar utama yang dijadikan pembanding.