

MODUL 9

PENGOLAHAN CITRA DIGITAL
DETEKSI TEPI & KONTUR (FREEMAN CHAINCODE)

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN
INFORMATIKA POLITEKNIK NEGERI
BANDUNG

**TIM
JEBRET**

KELAS 2A



009_022_032 Arief Rahman A_M. Ardi Nur I_Wildan Setya N |
PENGOLAHAN CITRA DIGITAL | MARET, 26 2023

DAFTAR ISI

1. DETEKSI TEPI (OPERATOR ROBERTS)	2
a. Citra	2
b. Source Code	3
c. Hasil	4
2. PEMBUKTIAN MATEMATIS OPERATOR ROBERTS	5
3. PENGENALAN DIGIT MENGGUNAKAN KONTUR DAN FREEMAN CHAIN CODE	6

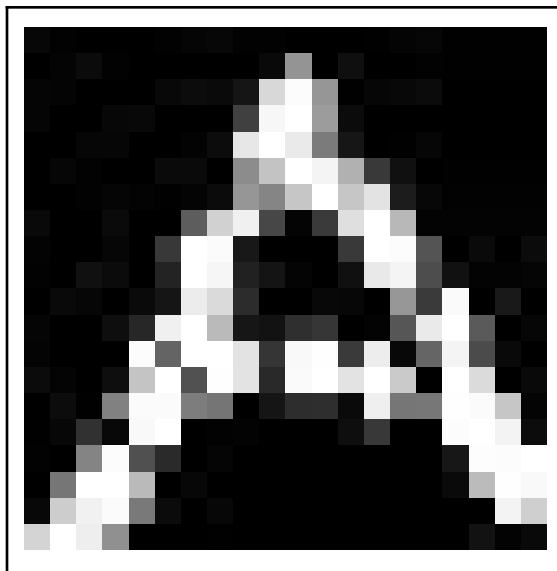
1. DETEKSI TEPI (OPERATOR ROBERTS)

a. Citra

- Citra 1 - 3000 x 3000 pixel



- Citra 2 - 20 x 20px



b. Source Code

```
# Created by Jebret Team

import streamlit as st
import numpy as np
import cv2

st.set_page_config(page_title="Roberts Edge Detection")

st.title("Edge Detection Robert 🤖")
# add subtitle
st.markdown("""
    ©Created by Jebret Team - 2023
    """)

uploaded_file = st.file_uploader("Choose an image", type=["jpg","jpeg","png"])

if uploaded_file is not None:
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    img = cv2.imdecode(file_bytes, 1)

    # Mengonversi gambar ke grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

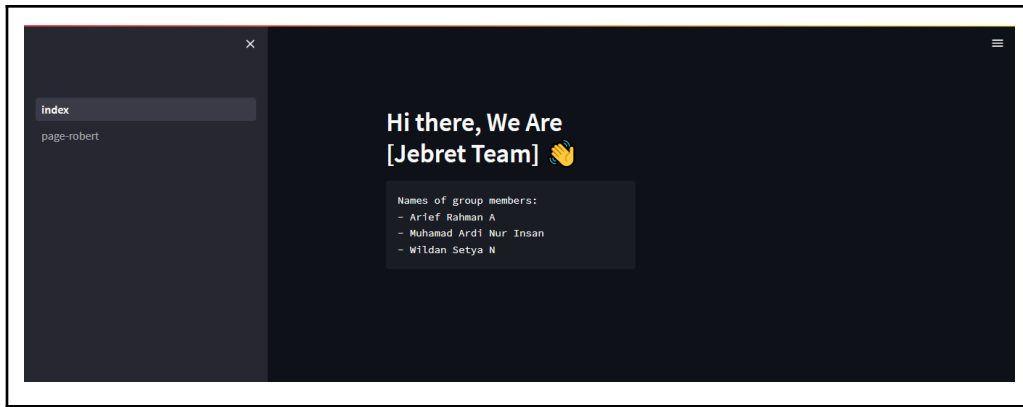
    # Menerapkan Operator Robert
    kernelx = np.array([[1, 0], [0, -1]])
    kernely = np.array([[0, 1], [-1, 0]])
    robertsx = cv2.filter2D(gray, -1, kernelx)
    robertsy = cv2.filter2D(gray, -1, kernely)
    roberts = cv2.add(robertsx, robertsy)

    col1, col2 = st.columns(2)

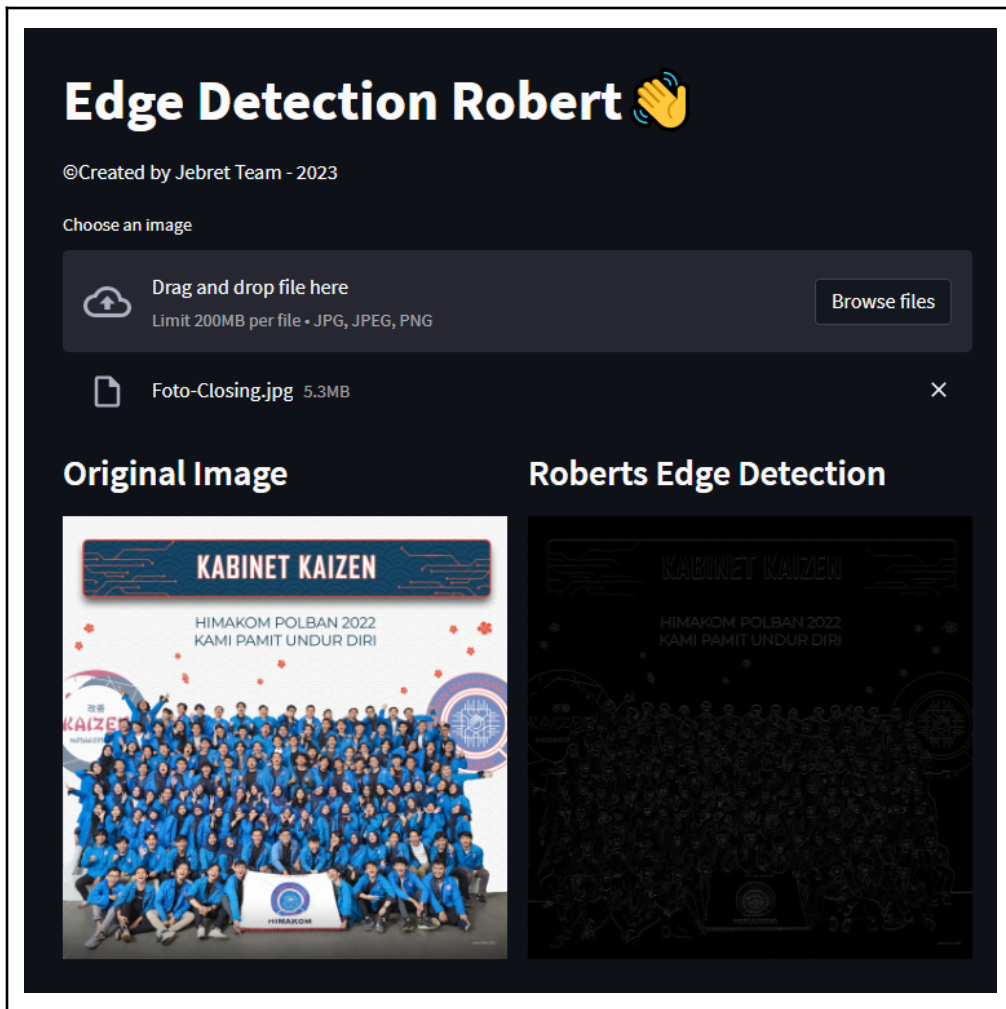
    with col1:
        st.subheader('Original Image')
        st.image(img, channels='BGR')

    with col2:
        st.subheader('Roberts Edge Detection')
        st.image(roberts, channels='GRAY')
```

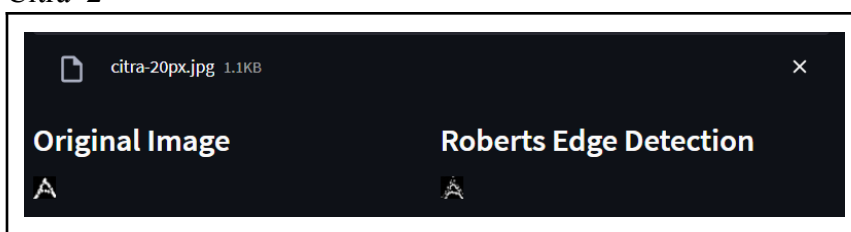
c. Hasil



• Citra 1



• Citra 2



2. PEMBUKTIAN MATEMATIS OPERATOR ROBERTS

Pada operator Roberts ini kita gunakan dua buah kernel 2 * 2 (Gx dan Gy)

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Kita gunakan operasi konvolusi

$$A_x = A * G_x$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	6	0	0	1	1	0	0	0	2	0	2	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	2	1	0	0	3	2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	7	0	2	0	0	1	0	0	0	0	0	2	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	1	3	1	0	0	0	0	0	0	1	4	1	0	4	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	252	254	1	1	0	0	0	2	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	247	3	0	0	4	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	3	5	0	1	0	0	0	4	0	1	0	2	2	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	4	0	3	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	2	3	0	2	4	3	0	1	3	0	0	0	0	0	1	1	2	0
11	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	0	0	0	252	3	0	0	4	1	2	1	0	1	1	0
12	0	0	0	0	0	0	0	0	0	0	0	3	0	255	255	250	1	254	2	0	2	0	251	0	0	0	2	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	250	251	0	0	0	1	0	0	1	0	0	0	1	0	0	0	6	4
14	0	0	0	0	0	0	0	0	0	0	0	3	0	2	0	1	2	1	1	0	1	1	0	255	0	0	2	2	0	1
15	0	0	0	0	0	0	0	0	0	7	0	0	252	254	0	0	0	0	0	0	0	2	0	251	0	3	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	1	254	0	1	7	0	0	0	0	0	0	250	252	0	0	0	0	2	6
17	0	0	0	0	0	0	0	0	0	0	1	250	0	0	1	0	2	0	0	0	4	0	0	0	0	0	0	1	1	0
18	0	0	0	0	2	0	0	0	0	0	2	3	0	251	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0
19	0	2	0	0	0	1	0	0	247	4	0	0	0	0	4	0	0	0	1	3	0	2	0	2	1	3	249	0	2	0
20	0	2	0	6	1	0	0	0	0	0	0	6	0	1	0	0	0	0	3	0	0	255	0	2	0	0	247	6	1	0
21	0	0	0	0	0	0	0	0	5	254	0	0	0	255	250	0	0	0	0	3	0	0	0	0	0	0	1	4	0	0
22	4	2	4	0	1	0	0	3	0	0	1	0	254	250	0	0	0	255	254	0	0	1	251	0	0	0	2	0	0	2
23	0	0	0	2	0	0	3	1	0	254	250	0	252	0	0	0	5	255	0	0	0	254	254	4	2	0	1	0	254	0
24	2	1	1	0	0	4	0	0	0	248	1	3	0	3	2	0	0	0	0	1	5	250	0	0	0	0	1	0	0	0
25	1	0	0	1	0	3	1	0	255	255	4	0	1	0	0	0	1	2	0	1	0	0	5	0	2	0	0	3	0	0
26	0	0	0	0	0	0	3	0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	2	0	0	0	1	0	247	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	2	
28	0	0	0	0	0	5	1	254	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
29	3	0	0	0	0	0	0	247	251	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	4	255
30	0	0	0	1	0	255	254	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	249	0

$$G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Kita gunakan operasi konvolusi

$$A_y = A * G_y$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	1	0	0	5	0	2	2	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	1	2	0	0	1	0	0	2	0	0	2	4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	4	0	6	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	2	3	1	0	0	0	3	0	0	0	6	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	4	1	0	3	0	0	0	0	5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	4	0	247	5	0	0	3	0	7	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	1	6	255	254	0	0	0	0	2	0	1	2	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	2	0	0	0	0	254	1	0	0	0	0	2	0	4	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	255	0	2	255	3	4	3	0	0	0	0	0	0	0	3	0	0	0
11	0	0	0	0	0	0	0	0	4	0	1	0	0	0	0	0	248	0	0	254	0	0	2	3	0	0	0	2	2	0
12	0	0	0	0	0	0	0	0	1	0	253	0	3	4	252	3	252	253	0	0	1	1	0	0	0	2	1	0	0	6
13	0	0	0	0	0	0	0	0	0	2	4	0	0	0	0	1	0	0	252	249	0	2	0	0	0	2	0	6	4	0
14	0	0	0	0	0	0	0	0	0	0	0	4	0	0	1	2	3	1	3	255	1	0	3	0	0	0	3	0	2	0
15	0	0	0	0	0	0	0	0	255	255	3	0	1	0	0	0	0	2	0	248	254	0	3	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	248	0	3	0	1	8	0	0	0	4	0	251	252	0	2	0	0	0	1	4	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	4	0	255	2	0	3	0	0	2	0	0	0
18	0	0	0	3	0	0	1	0	4	1	0	0	4	0	0	0	0	0	0	0	0	255	1	0	1	0	0	0	0	0
19	2	0	0	0	0	0	1	2	0	0	251	3	0	0	0	2	1	6	0	0	0	2	0	251	249	5	1	0	0	1
20	2	0	6	1	0	0	250	255	0	0	0	1	5	0	0	0	0	0	0	0	0	6	249	250	0	0	0	1	4	0
21	0	0	0	0	0	4	0	0	0	5	252	0	0	3	251	252	4	0	0	0	2	0	0	0	255	5	0	0	0	0
22	2	4	2	0	0	0	0	0	0	0	1	255	0	0	3	250	250	0	255	248	0	1	1	0	0	0	0	0	0	3
23	0	0	0	3	0	250	255	0	1	2	0	251	1	0	0	5	254	0	1	254	249	0	4	255	255	2	0	1	0	2
24	1	2	0	0	5	0	251	0	0	0	0	0	2	2	0	1	0	0	1	4	255	0	0	0	250	1	1	0	0	0
25	0	1	0	0	254	0	3	255	0	4	0	1	0	0	0	1	2	0	1	0	0	5	0	3	254	252	0	0	4	0
26	0	0	1	0	4	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	254	248	0	0	4
27	2	0	3	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	255	0	0	2	0
28	0	0	251	4	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	253	251	0	0
29	0	3	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	1	252	254	5
30	0	0	0	0	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	250	0	0

Dengan hasil mengkonvolusikan citra berdasarkan kernel Gx dan Gy, maka kita dapat menghitung magnitudo gradiennya dengan cara.

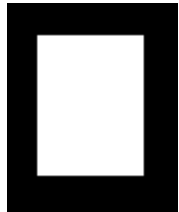

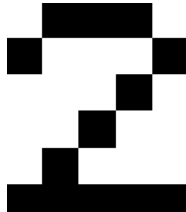
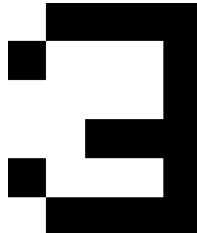
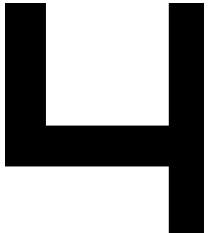
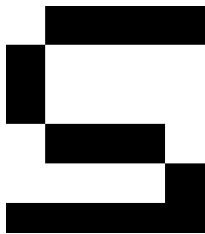
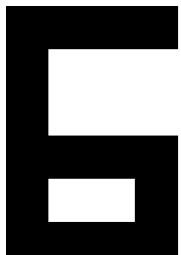

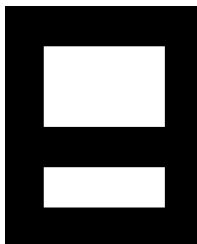
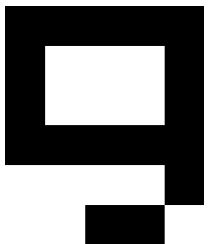
$$G = \sqrt{I * G_x^2 + I * G_y^2}$$

Maka hasil citra penerapan operator Roberts adalah sebagai berikut.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
1	0	0	0	0	0	0	0	0	0	0	0	2	2	1	4	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	2	0	0	1	4	1	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	2	4	3	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0	2	0	0	0	0	5	0	0	2	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0	0	2	2	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	2	1	124	2	0	0	128	127	4	0	0	0	0	1	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	2	0	2	2	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	4	130	127	0	0	0	0	2	1	0	0	2	2	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	128	0	0	0	0	0	2	2	2	2	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	1	128	2	1	128	4	4	2	0	2	0	0	0	0	0	2	1	0	0	
11	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	124	0	126	128	0	0	2	2	2	0	0	0	2	1	
12	0	0	0	0	0	0	0	0	0	0	0	126	2	2	130	254	126	126	254	1	0	2	0	126	0	0	1	2	0	0	3
13	0	0	0	0	0	0	0	0	0	1	2	0	125	126	0	0	0	0	126	124	0	1	0	0	0	1	0	3	5	2	
14	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	2	2	1	2	128	1	0	2	128	0	0	2	1	1	0	
15	0	0	0	0	0	0	0	0	0	128	131	2	0	126	127	0	0	0	1	124	127	1	2	126	0	2	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	124	0	2	127	0	4	4	0	0	2	0	126	126	125	127	0	0	0	3	3	
17	0	0	0	0	0	0	0	0	0	0	0	125	0	0	0	1	1	0	0	2	2	128	0	2	0	0	2	0	0	0	
18	0	0	0	2	1	0	0	0	2	0	1	2	2	126	0	0	0	0	0	0	0	0	128	0	0	0	128	0	0	0	
19	1	1	0	0	0	1	1	0	124	128	2	0	0	0	3	0	3	0	0	2	1	1	126	126	3	2	124	0	1	0	
20	1	1	3	4	0	0	125	128	0	0	0	4	2	0	0	0	0	0	2	0	0	128	3	126	125	0	124	3	1	2	
21	0	0	0	0	0	2	0	0	2	130	126	0	0	0	129	250	126	2	0	0	2	1	0	0	0	128	3	0	2	0	0
22	3	3	3	0	0	0	0	2	0	0	1	128	127	125	2	125	125	128	254	124	0	1	126	0	0	0	1	0	0	2	
23	0	0	0	2	0	125	129	0	0	128	125	126	126	0	0	2	130	128	0	127	124	127	129	130	128	1	0	0	127	1	
24	2	2	0	0	2	2	126	0	0	124	0	2	1	2	1	0	0	0	0	2	130	125	0	0	125	0	1	0	0	0	
25	0	0	0	0	127	2	2	128	128	130	2	0	0	0	0	0	2	1	0	0	0	2	2	2	128	126	0	2	2	0	
26	0	0	0	0	2	0	2	4	128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	127	124	0	0	2	
27	1	1	2	0	0	2	0	124	124	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	128	2	1	2	
28	0	0	126	2	0	2	4	127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	126	126	0	0	0	
29	2	2	0	0	0	0	124	126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4	0	126	129	130	124	
30	0	0	0	0	1	128	127	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	125	124	

3. PENGENALAN DIGIT MENGGUNAKAN KONTUR DAN FREEMAN CHAIN CODE

Untuk mengetahui chain code dari setiap digit dari 0-9, kita perlu membuat citra dari setiap digit tersebut. Dimana pada praktikum kali ini, kelompok kami menggunakan citra berukuran 5 * 6. Citra-citra tersebut antara lain:

Digit 0	Digit 1	Digit 2	Digit 3	Digit 4
				
Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
				

Dari citra digit 0-9 yang telah dibuat, untuk menentukan chain code nya kita perlu yang namanya mengubah citra menjadi citra biner. Agar nilai piksel dari citra tersebut hanya 0 dan 255 saja. Hal ini dikarenakan akan lebih memudahkan dalam melakukan kontur.

Untuk mengubah menjadi citra biner kita dapat melakukannya dengan code seperti berikut.

```
def binarize_image(img):  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)  
    return binary
```

Kode diatas adalah sebuah fungsi binarize_image yang digunakan untuk mengubah citra berwarna menjadi citra biner (hitam-putih) dengan menggunakan metode thresholding Otsu.

- Prosesnya dimulai dengan mengubah citra input img dari skala warna RGB menjadi skala abu-abu (grayscale) menggunakan fungsi **cv2.cvtColor** dari library **OpenCV** dengan parameter **cv2.COLOR_BGR2GRAY**.
- Selanjutnya, fungsi **cv2.threshold** digunakan untuk melakukan proses thresholding pada citra grayscale dengan nilai threshold 0 dan mode thresholding **cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU**. Mode ini menggabungkan dua jenis thresholding, yaitu **cv2.THRESH_BINARY_INV** untuk menghasilkan citra biner dengan piksel putih untuk nilai pixel yang lebih rendah dari threshold dan piksel hitam untuk nilai pixel yang lebih besar dari threshold, dan **cv2.THRESH_OTSU** untuk menghitung nilai threshold secara

otomatis berdasarkan histogram citra.

- Hasil akhirnya adalah citra biner yang ditampung pada variabel binary dan kemudian dikembalikan oleh fungsi `binarize_image`.

Setelah melakukan perubahan citra menjadi citra biner, langkah selanjutnya kita cari konturnya dari setiap digit menggunakan kode seperti berikut.

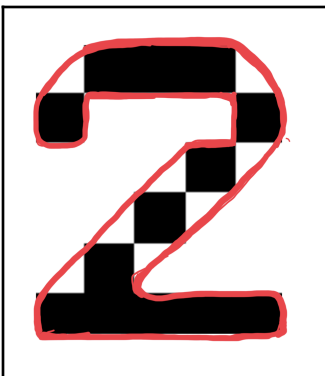
```
def get_contours(binary):  
    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)  
    if len(contours) == 0:  
        print("Error: No contour found")  
        return None  
    return contours[0]
```

Fungsi `get_contours` digunakan untuk mengambil kontur objek pada citra biner.

- Citra biner dimasukkan sebagai parameter `binary`. Kemudian, fungsi `cv2.findContours` digunakan untuk mencari kontur pada citra biner tersebut.
- Pada pemanggilan fungsi tersebut, parameter `cv2.RETR_EXTERNAL` digunakan untuk mendapatkan hanya kontur eksternal dari objek (tidak termasuk kontur dalam objek), dan parameter `cv2.CHAIN_APPROX_NONE` digunakan untuk menyimpan semua titik pada kontur.
- Jika tidak ada kontur yang ditemukan, fungsi akan mengembalikan nilai `None`. Jika terdapat satu atau lebih kontur yang ditemukan, maka fungsi akan mengembalikan kontur pertama pada daftar kontur tersebut (`contours[0]`).

Hasil dari contour tersebut adalah sebagai berikut

- Misalkan citra dengan digit 2



Garis yang berwarna merah itu kira-kira contour nya.

Jika di dalam codingan hasil dari contournya itu adalah sebagai berikut.

	0	1	2	3	4
0	0	255	255	255	0
1	255	0	0	0	255
2	0	0	0	255	0
3	0	0	255	0	0
4	0	255	0	0	0
5	255	255	255	255	255

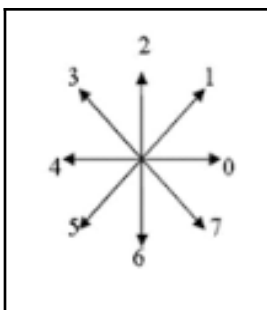
```

[[[1 0]]
 [[0 1]]
 [[1 0]]
 [[2 0]]
 [[3 0]]
 [[4 1]]
 [[3 2]]
 [[2 3]]
 [[1 4]]
 [[0 5]]
 [[1 5]]
 [[2 5]]
 [[3 5]]
 [[4 5]]
 [[3 5]]
 [[2 5]]
 [[1 4]]
 [[2 3]]
 [[3 2]]
 [[4 1]]
 [[3 0]]
 [[2 0]]]

```

Setelah mendapatkan contour dari setiap digit citra, kita tinggal cari chain code nya.

Chain code ini dimana arah kode nya adalah sebagai berikut.



Misalkan citra dengan digit 2

	0	1	2	3	4
0	0	255	255	255	0
1	255	0	0	0	255
2	0	0	0	255	0
3	0	0	255	0	0
4	0	255	0	0	0
5	255	255	255	255	255

chaincode: [5, 1, 0, 0, 7, 5, 5, 5, 5, 0, 0, 0, 0, 4, 4, 3, 1, 1, 1, 3, 4, 4]

arah dari chain code itu sendiri berdasarkan contour nya.

Yang perlu diperhatikan dalam chain code ini adalah

```
# Representasi chain code:
# 3 2 1
# \ | /
# 4--P--0
# / | \
# 5 6 7
#
# Representasi chain code berdasarkan ketetanggaan terhadap titik P
# Deltax Deltay code
# 0 +1 6
# +1 +1 7
# +1 0 0
# +1 -1 1
# 0 -1 2
# -1 -1 3
# -1 0 4
# -1 +1 5
#
```

Sehingga dengan memperhatikan ketetanggaan terhadap titik P, kode yang dapat dibuat untuk menentukan chain code adalah sebagai berikut.

```

chaincode = []
for i in range(len(contour)):
    if i == len(contour)-1:
        next = 0
    else:
        next = i+1
    x = contour[next][0][0] - contour[i][0][0]
    y = contour[next][0][1] - contour[i][0][1]
    if x == 1 and y == 0:
        chaincode.append(0)
    elif x == 1 and y == -1:
        chaincode.append(1)
    elif x == 0 and y == -1:
        chaincode.append(2)
    elif x == -1 and y == -1:
        chaincode.append(3)
    elif x == -1 and y == 0:
        chaincode.append(4)
    elif x == -1 and y == 1:
        chaincode.append(5)
    elif x == 0 and y == 1:
        chaincode.append(6)
    elif x == 1 and y == 1:
        chaincode.append(7)
return chaincode

```

Dari hasil-hasil chain code tersebut kita dapat menentukan template chain code untuk setiap digit, sehingga dari template tersebut lah komputer dapat mengenali digit berapa yang ada pada citra.

```

def recognize_digit(chaincode):
    # 0
    if chaincode == [6, 6, 6, 6, 6, 0, 0, 0, 0, 2, 2, 2, 2, 4, 4, 4]:
        return '0'
    # 1
    elif chaincode == [5, 1, 7, 6, 6, 6, 6, 2, 2, 2, 2, 4]:
        return '1'
    # 2
    elif chaincode == [5, 1, 0, 0, 7, 5, 5, 5, 5, 0, 0, 0, 0, 4, 4, 3, 1, 1, 3, 4, 4]:
        return '2'
    # 3
    elif chaincode == [5, 1, 0, 0, 7, 6, 5, 4, 0, 7, 5, 4, 4, 3, 7, 0, 0, 0, 2, 2, 2, 2, 4, 4, 4]:
        return '3'
    # 4
    elif chaincode == [6, 6, 6, 0, 0, 0, 7, 6, 2, 2, 2, 2, 2, 6, 6, 5, 4, 4, 3, 2, 2]:
        return '4'
    # 5
    elif chaincode == [5, 6, 7, 0, 0, 7, 5, 4, 4, 4, 0, 0, 0, 0, 2, 3, 4, 4, 3, 2, 1, 0, 0, 0, 4, 4, 4]:
        return '5'
    # 6
    elif chaincode == [6, 6, 6, 6, 6, 0, 0, 0, 2, 2, 4, 4, 3, 2, 1, 0, 0, 4, 4, 4]:
        return '6'
    # 7
    elif chaincode == [0, 0, 7, 6, 5, 6, 6, 2, 2, 1, 2, 2, 4, 4, 4]:
        return '7'
    # 8
    elif chaincode == [6, 6, 6, 6, 6, 0, 0, 0, 0, 2, 2, 2, 2, 2, 4, 4, 4, 4]:
        return '8'
    # 9
    elif chaincode == [6, 6, 6, 0, 0, 0, 7, 5, 4, 0, 1, 2, 2, 2, 2, 4, 4, 4, 4]:
        return '9'
    else:
        return None

```

Untuk kode keseluruhannya dalam mencari recognize digit adalah sebagai berikut

```
st.title("Digit Recognizer")

uploaded_file = st.file_uploader("Upload Image", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
    st.image(img, channels="BGR")

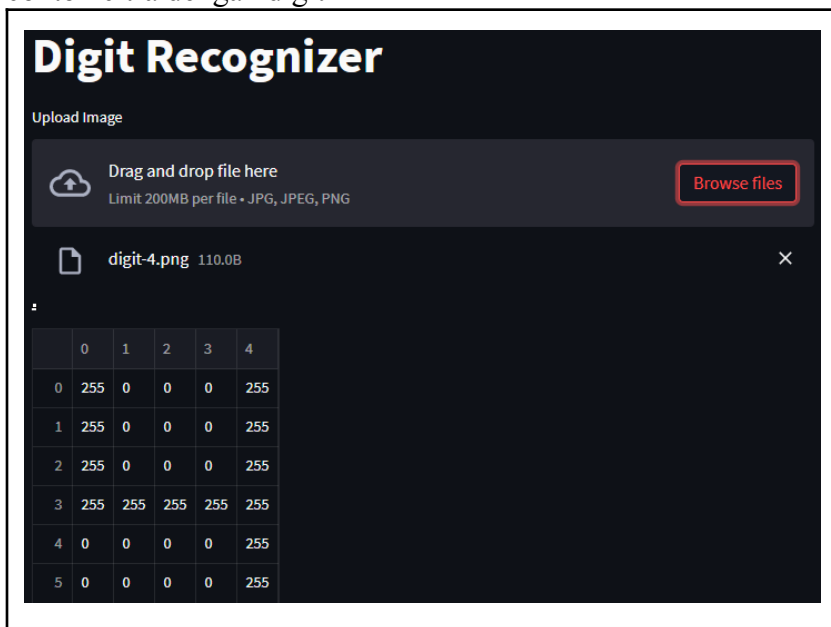
    binary = binarize_image(img)
    st.write(binary)
    contour = get_contours(binary)
    chaincode = get_freeman_chaincode(contour)
    digit = recognize_digit(chaincode)
    # digit
    st.write(f"chaincode: {chaincode}")

    if digit is not None:
        st.write(f"Recognized Digit: {digit}")
```

Jadi dalam urutan main codenya adalah sebagai berikut:

1. Image akan diupload menggunakan st.file_uploader dengan ekstensi image nya adalah
 - a. JPG
 - b. JPEG
 - c. PNG
2. Kemudian citra yang telah di upload akan dijadikan citra biner menggunakan fungsi yang telah dijelaskan di atas, untuk hasilnya akan seperti ini.

contoh citra dengan digit 4



3. citra biner tersebut akan dilakukan pencarian contour, menggunakan fungsi `get_contours`
 4. Setelah mendapatkan contour nya, maka dapat dicari chain code nya menggunakan fungsi `get_freeman_chaincode`
 - Hasil dari chain code akan berupa seperti berikut
- Contoh jika citra nya dengan digit 4

	0	1	2	3	4
0	255	0	0	0	255
1	255	0	0	0	255
2	255	0	0	0	255
3	255	255	255	255	255
4	0	0	0	0	255
5	0	0	0	0	255

chaincode: [6, 6, 6, 0, 0, 0, 7, 6, 2, 2, 2, 2, 2, 6, 6, 5, 4, 4, 3, 2, 2]

5. Dari hasil chain code tersebut, maka kita tinggal memanggil fungsi `recognize_digit`, dengan membawa nilai chain code yang sudah dilakukan. Nilai chain code tersebut akan disamakan dengan template, jika chain code memenuhi yang ada pada template maka akan mengembalikan nilai digitnya.
 - Hasil dari pemanggilan fungsi tersebut adalah sebagai berikut

	0	1	2	3	4
0	255	0	0	0	255
1	255	0	0	0	255
2	255	0	0	0	255
3	255	255	255	255	255
4	0	0	0	0	255
5	0	0	0	0	255

Recognized Digit: 4

- Untuk Hasil Streamlit secara keseluruhan adalah sebagai berikut

