

UJIAN AKHIR SEMESTER
MATA KULIAH
PRAKTIKUM PENGOLAHAN GAMBAR

Topik
Segmentasi Gambar atau Foto



PENYUSUN LAPORAN



Nama Mahasiswa

Yudha Mulia

NIM

062340833203

Kelas

1MIN

PROGRAM STUDI MANAJEMEN INFORMATIKA
JURUSAN MANAJEMEN INFORMATIKA
POLITEKNIK NEGERI SRWIJAYA
2024

Semester	: 1	Tanggal	: 08 januari 2024
Dosen Pengampu	: Sulistiyanto, MTI	Kelas	: 1MIN
CP	: Mahasiswa/i dapat mengerjakan UAS dengan prosedur yang telah ditentukan.		

Nama Mahasiswa : Yudha Mulia

NIM : 062340833203

TUJUAN

1. Memberikan pemahaman singkat dan padat tentang bagaimana program bekerja dalam membaca, menganalisis tepi, dan melakukan segmentasi warna merah pada gambar.
2. Memberikan gambaran yang jelas dan singkat tentang apa yang dicapai oleh kode yang digunakan dan bagaimana cara kerjanya.

MATERIAL

1. Laptop
2. Aplikasi Visual Studio Code, Bahasa Pemrograman Python, PIP Numpy, PIP Matplotlib, dan PIP Open Cv.

PROJECT

1. Setiap capture gambar, diberi penjelasan
 - a). Gambar 1.1 Mendefinisikan fungsi `process_image` yang menerima path gambar sebagai argumen.
 - b). Gambar 1.2 Membaca gambar asli dari path yang diberikan.
 - c). Gambar 1.3 :
 - Menggunakan deteksi tepi Canny untuk menemukan tepi pada gambar.
 - Mengonversi gambar ke skala abu-abu sebelum deteksi tepi.
 - Mengonversi gambar tepi ke format RGB.
 - d). Gambar 1.4 :
 - Mengonversi gambar ke ruang warna HSV.
 - Membuat mask dengan menentukan range warna dari gambar yang digunakan.
 - Mengaplikasikan mask pada gambar asli menggunakan operasi bitwise AND.
 - e). Gambar 1.5 :
 - Membuat jendela dengan ukuran 10x5 inch.
 - Menampilkan gambar asli, gambar deteksi tepi, dan gambar hasil segmentasi dalam satu jendela dengan tiga subplot.
 - Menonaktifkan sumbu pada setiap subplot.
 - Menampilkan jendela.

f). Gambar 1.6 Memanggil fungsi process_image dengan memberikan path gambar 'nama_gambar.jpg' sebagai argumen.

2. Tulislah dokumentasi kode python untuk segmentasi dan deteksi gambar

a. Process Image

```
def process_image(image_path):
```

Gambar 1.1

b. Membaca Gambar

```
original_image = cv2.imread(image_path)
```

Gambar 1.2

c. Deteksi Tepi

```
edges = cv2.Canny(cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY), 50, 150)
edge_image = cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)
```

Gambar 1.3

d. Segmentasi

```
hsv = cv2.cvtColor(original_image, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv, np.array([0, 0, 0]), np.array([10, 400, 255]))
segmented_image = cv2.bitwise_and(original_image, original_image, mask=mask)
```

Gambar 1.4

e. Jendela Output

```
plt.figure(figsize=(10, 5))

plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title('Gambar Asli')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title('Deteksi Gambar')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title('Segmentasi')
plt.axis('off')

plt.show()
```

Gambar 1.5

f. Pemanggilan Fungsi Path

```
process_image('apel.jpg')
```

DOKUMENTASI KODE & HASIL PROGRAM

1. Citra Wajib “Pisang.jpg”

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

# function to perform edge detection on the image
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# function to remove the background from the image
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = np.array([20,100,100])
    upper_yellow = np.array([30,255,255])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    result = cv2.bitwise_and(image, image, mask = mask)
    return result

# load the original image
original_image = cv2.imread('pisang.jpg')

# perform edge detection on the image
edge_image = edge_detection(original_image)




# display the three images in one window
plt.figure(figsize=(10, 5))

# original image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("original image")
plt.axis("off")

# image with edges detected
plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title("edge detected image")
plt.axis("off")

# segmented image with background removed
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title("segmented image")
plt.axis("off")

plt.show()
```

original image	edge detected image	segmented image
		

2.Citra Wajib “lemon.jpg”

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

# function to perform edge detection on the image
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# function to remove the background from the image
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = np.array([20,100,100])
    upper_yellow = np.array([30,255,255])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    result = cv2.bitwise_and(image, image, mask = mask)
    return result

# load the original image
original_image = cv2.imread('lemon.jpg')

# perform edge detection on the image
edge_image = edge_detection(original_image)

# remove the background from the image
segmented_image = remove_background(original_image)

# display the three images in one window
plt.figure(figsize=(10, 5))


# original image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("original image")
plt.axis("off")

# image with edges detected
plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title("edge detected image")
plt.axis("off")

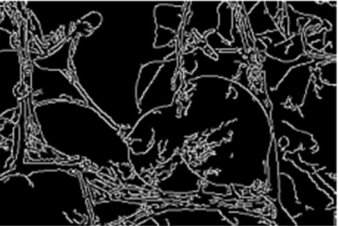
# segmented image with background removed
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title("segmented image")
plt.axis("off")

plt.show()
```

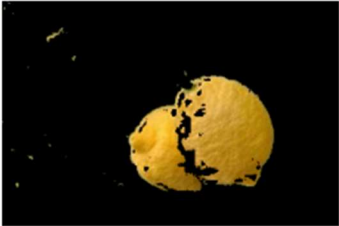
original image



edge detected image



segmented image



3. Citra Wajib “apel2.jpg”

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

# function to perform edge detection on the image
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# function to remove the background from the image
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0,25,0])
    upper_red = np.array([10,400,255])
    mask = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(image, image, mask = mask)
    return result

# load the original image
original_image = cv2.imread('apel2.jpg')

# perform edge detection on the image
edge_image = edge_detection(original_image)

# remove the background from the image
segmented_image = remove_background(original_image)

# display the three images in one window
plt.figure(figsize=(10, 5))


# original image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("original image")
plt.axis("off")

# image with edges detected
plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title("edge detected image")
plt.axis("off")


# segmented image with background removed
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title("segmented image")
plt.axis("off")

plt.show()
```

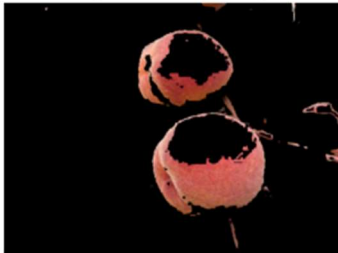
original image



edge detected image



segmented image



4. Citra Bebas “Cangkir.jpg”

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

# function to perform edge detection on the image
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# function to remove the background from the image
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0,100,0])
    upper_red = np.array([10,400,255])
    mask = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(image, image, mask = mask)
    return result

# load the original image
original_image = cv2.imread('Cangkir.jpg')

# perform edge detection on the image
edge_image = edge_detection(original_image)
```

```
[35] # remove the background from the image
segmented_image = remove_background(original_image)

# display the three images in one window
plt.figure(figsize=(10, 5))

# original image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("original image")
plt.axis("off")

# image with edges detected
plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title("edge detected image")
plt.axis("off")

# segmented image with background removed
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title("segmented image")
plt.axis("off")

plt.show()
```

original image



edge detected image



segmented image



5. Citra Bebas “Bunga.jpg”


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

# function to perform edge detection on the image
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# function to remove the background from the image
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_orange = np.array([0, 100, 45])
    upper_orange = np.array([225, 250, 255])
    mask = cv2.inRange(hsv, lower_orange, upper_orange)
    result = cv2.bitwise_and(image, image, mask = mask)
    return result

# load the original image
original_image = cv2.imread('Bunga.jpg')

# perform edge detection on the image
edge_image = edge_detection(original_image)

# remove the background from the image
segmented_image = remove_background(original_image)

# display the three images in one window
plt.figure(figsize=(10, 5))

# original image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
plt.title("original image")
plt.axis("off")

# image with edges detected
plt.subplot(1, 3, 2)
plt.imshow(edge_image)
plt.title("edge detected image")
plt.axis("off")

# segmented image with background removed
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
plt.title("segmented image")
plt.axis("off")

plt.show()
```

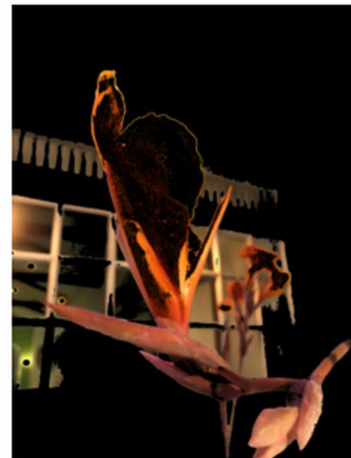
original image



edge detected image



segmented image



ANALISIS

Kode ini melakukan tugas-tugas berikut:

1. Muat gambar (pisang.jpg) menggunakan `cv2.imread()`.
2. Lakukan deteksi tepi pada gambar asli menggunakan fungsi `edge_detection()`. Di dalam fungsi ini, gambar diubah terlebih dahulu menjadi skala abu-abu menggunakan `cv2.cvtColor()`. Kemudian diterapkan algoritma deteksi tepi Canny untuk mencari tepi pada gambar menggunakan `cv2.Canny()`. Hasilnya adalah gambar hitam putih yang bagian tepinya disorot dengan warna putih.
3. Hapus latar belakang dari gambar asli menggunakan fungsi `hapus_latar_belakang()`. Di dalam fungsi ini, gambar terlebih dahulu dikonversi ke ruang warna HSV menggunakan `cv2.cvtColor()`. Kemudian, masker warna dibuat menggunakan fungsi `cv2.inRange()` untuk menentukan rentang warna kuning. Terakhir, fungsi `cv2.bitwise_and()` digunakan untuk menerapkan mask ke gambar asli, sehingga secara efektif menghilangkan latar belakang.
4. Tampilkan gambar asli, gambar dengan tepi terdeteksi, dan gambar tersegmentasi dengan latar belakang dihapus secara berdampingan menggunakan `plt.subplot()`.
5. Dalam fungsi ini, gambar diubah terlebih dahulu menjadi skala abu-abu menggunakan `cv2.cvtColor()`. Kemudian diterapkan algoritma deteksi tepi Canny untuk mencari tepi pada gambar menggunakan `cv2.Canny()`. Fungsi `cv2.Canny()` mengambil tiga argumen: gambar skala abu-abu, ambang batas bawah untuk deteksi tepi, dan ambang batas atas untuk deteksi tepi. Dalam hal ini, ambang batas bawah adalah 50 dan ambang batas atas adalah 150. Hasilnya adalah gambar hitam putih yang tepinya disorot dengan warna putih.
6. Dalam fungsi ini, gambar dikonversi terlebih dahulu ke ruang warna HSV menggunakan `cv2.cvtColor()`. Kemudian, masker warna dibuat menggunakan fungsi `cv2.inRange()` untuk menentukan rentang warna kuning. Nilai kuning bawah adalah [20.100.100] dan nilai kuning atas adalah [30.255.255]. Terakhir, fungsi `cv2.bitwise_and()` digunakan untuk menerapkan mask ke gambar asli, sehingga secara efektif menghilangkan latar belakang.
7. Cuplikan kode terakhir menampilkan gambar asli, gambar dengan tepi terdeteksi, dan gambar tersegmentasi dengan latar belakang dihapus secara berdampingan menggunakan `plt.subplot()`.
8. Kode ini memberikan contoh bagus tentang cara menggunakan OpenCV dengan Python untuk melakukan tugas pemrosesan gambar seperti deteksi tepi dan penghapusan latar belakang.

KESIMPULAN

Secara kesimpulan, kode ini melakukan deteksi garis miring dan penghapusan latar belakang pada citra menggunakan OpenCV. Hasilnya adalah citra tersegmentasi dengan latar belakang dihapus dan garis miring dari citra asli yang ditampilkan.