LAPORAN UJIAN AKHIR SEMESTER
MATA KULIAH
PRAKTIKUM PENGOLAHAN GAMBAR DAN FOTOGRAFI

Topik
Deteksi Tepi dan Segmentasi Citra

Dosen Pengampu :
Sulistiyanto, S.Kom., M.Ti

PENYUSUN LAPORAN

Nama Mahasiswa	NIM	Kelas
Rio Adrian	062340833237	1MIO

PROGRAM STUDI MANAJEMEN INFORMATIKA
JURUSAN MANAJEMEN INFORMATIKA
POLITEKNIK NEGERI SRIWIJAYA

2023

KATA PENGANTAR

Puji dan Syukur penulis haturkan atas kehadirat Allah SWT, atas segala taufiq dan hidayah – Nya, sehingga penulis dapat menyelesaikan penyusunan laporan ini dalam rangka memenuhi tugas ujian akhir semester (UAS) untuk mata kuliah pengolaha gambar dan fotografi. Tak lupa sholawat serta salam mudah – mudahan selalu tercurah kepada baginda Nabi Muhammad SAW beserta keluarga dan para sahabatnya.

Laporan ini menggambarkan hasil implementasi deteksi tepi dan segmentasi warna pada gambar, suatu proyek yang menawarkan wawasan mendalam tentang pemrosesan gambar dan teknik-teknik yang digunakan. Melalui laporan ini, diharapkan pembaca dapat memahami pencapaian signifikan dalam deteksi tepi yang berhasil menyoroti kontur objek secara efektif. Selain itu, evaluasi segmentasi warna memberikan gambaran tentang keberhasilan dalam menghapus latar belakang dengan fokus pada warna yang diinginkan, meskipun tantangan muncul ketika fokus segmentasi diperluas untuk mencakup semua warna. Kesimpulan dan rekomendasi yang disajikan diharapkan dapat menjadi dasar untuk pengembangan lebih lanjut dan peningkatan kualitas implementasi deteksi tepi dan segmentasi warna pada proyek-proyek mendatang.

Demikian laporan ini penulis susun, semoga laporan ini memberikan manfaat bagi semua pihak dan penulis sendiri secara optimal.

Banyuasin, 10 januari 2024

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Pemrosesan citra telah menjadi aspek penting dalam dunia komputasi modern dengan berbagai aplikasi yang melibatkan analisis visual. Tugas akhir ini bertujuan untuk mengimplementasikan teknik dasar pemrosesan citra dengan fokus pada pengenalan warna, deteksi tepi, dan penghapusan latar belakang menggunakan library OpenCV.

1.2 PERMASALAHAN

Dalam pemrosesan citra, tahapan seperti ekstraksi warna, deteksi tepi, dan penghapusan latar belakang memiliki peran krusial dalam mendapatkan informasi yang akurat. Penelitian ini mencoba mendalami implementasi teknik-teknik tersebut untuk menganalisis objek dalam citra digital.

1.3 TUJUAN

1. Mengimplementasikan konsep dasar pemrosesan citra.
2. Menerapkan teknik ekstraksi warna, deteksi tepi, dan penghapusan latar belakang dengan library OpenCV.
3. Menganalisis setiap langkah implementasi dalam pemrosesan citra.
4. Memahami perbedaan pendekatan analisis objek berdasarkan karakteristik dan warna yang diinginkan.

1.4 MANFAAT

Hasil praktikum diharapkan dapat memberikan pemahaman yang lebih mendalam tentang konsep pemrosesan citra dan penerapannya dalam mengenali objek. Manfaat praktikum melibatkan kontribusi pada pengembangan sistem otomatisasi, pengolahan gambar medis, dan aplikasi cerdas lainnya.

BAB II

ALAT DAN BAHAN

2.1 Alat

1. Komputer atau Laptop
   Digunakan untuk menjalankan script Python dan program pemrosesan gambar.
2. Peragkat Lunak Python
   Sebagai platform pengembangan utama menggunakan bahasa pemrograman Python.
3. Library OpenCV, Matplotlib dan Numpy
   Digunakan untuk implementasi fungsi deteksi tepi dan segmentasi warna kuning pada gambar.

2.2 Bahan

1. Gambar uji berupa empat gambar buah – buahan dan dua gambar objek lain yang relevan.

BAB III

LANDASAN TEORI

3.1 Pemrosesan Citra

Pemrosesan citra merupakan suatu teknik yang digunakan untuk memanipulasi gambar atau citra guna mendapatkan informasi yang berguna. Proses ini melibatkan berbagai tahapan, termasuk segmentasi, ekstraksi fitur, dan analisis citra. Dalam konteks pemrosesan citra, terdapat berbagai metode yang dapat diterapkan, seperti pengenalan warna, deteksi tepi, dan penghapusan latar belakang.

3.2 Teknik Pemrosesan Citra

Teknik pemrosesan citra merujuk pada sekumpulan metode atau prosedur yang digunakan untuk memanipulasi atau mengubah citra digital dengan tujuan tertentu. Pemrosesan citra dapat melibatkan serangkaian langkah-langkah untuk meningkatkan kualitas citra, menggali informasi yang berguna, atau membuat citra tersebut lebih sesuai untuk analisis atau aplikasi tertentu.

3.2.1 Pengenalan Warna

Pengenalan warna menjadi aspek kritis dalam pemrosesan citra, khususnya dalam pengidentifikasian objek berdasarkan karakteristik warnanya. Pada praktikum ini, pengenalan warna digunakan untuk mengekstrak objek tertentu dari gambar dengan membatasi rentang warna yang dianggap relevan.

3.2.2 Deteksi Tepi Dalam Pemrosesan Citra

Deteksi tepi adalah teknik yang digunakan untuk meningkatkan ketajaman citra dengan menyorot batas antara objek. Pada praktikum ini, deteksi tepi dengan menggunakan metode Canny diterapkan pada hasil ekstraksi objek untuk memperoleh informasi batas yang lebih jelas.

3.2.3 Penghapusan Latar Belakang

Penghapusan latar belakang diperlukan untuk meningkatkan fokus pada objek utama dalam citra. Dalam praktikum, penghapusan latar belakang dilakukan dengan membuat masker yang menyoroti objek dan menghapus bagian latar belakang yang tidak diinginkan.

3.3 Library OpenCV

OpenCV (Open Source Computer Vision) adalah sebuah library atau pustaka perangkat lunak sumber terbuka yang menyediakan berbagai fungsi dan algoritma untuk pemrosesan citra dan penglihatan komputer. Library ini ditujukan untuk membantu pengembang perangkat lunak dalam membangun aplikasi yang melibatkan pengolahan citra, pengenalan pola, dan analisis video.

BAB IV

METODE PENGOLAHAN CITRA

4.1 Deteksi Tepi Gambar

Berikut adalah kode untuk melakukan deteksi tepi dan segmentasi secara menyeluruh :

```
Wajib > lemon.py > ...
1   import cv2
2   import numpy as np
3   import matplotlib.pyplot as plt
4
5   # Fungsi untuk melakukan deteksi tepi pada gambar
6   def edge_detection(image):
7       gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8       edges = cv2.Canny(gray, 50, 150)
9       return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)
10
11  # Fungsi untuk melakukan segmentasi pada gambar
12  def remove_background(image):
13      hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
14      lower_yellow = np.array([20, 100, 100])  # Nilai batas rendah untuk warna kuning dalam format HSV
15      upper_yellow = np.array([30, 255, 255])  # Nilai batas tinggi untuk warna kuning dalam format HSV
16      mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
17      result = cv2.bitwise_and(image, image, mask=mask)
18      return result
19
20  # Baca gambar
21  original = cv2.imread('Wajib/lemon.jpg')
22
23  # Deteksi tepi pada gambar
24  edges = edge_detection(original)
25
26  # Hapus background pada gambar dengan segmentasi warna kuning
27  segmented = remove_background(original)
28
29  # untuk menampilkan ketiga gambar dalam satu jendela
30  plt.figure(figsize=(10, 5))
31
```

```
Wajib > lemon.py > ...
28
29      # untuk menampilkan ketiga gambar dalam satu jendela
30      plt.figure(figsize=(10, 5))
31
32      # Gambar original
33      plt.subplot(1, 3, 1)
34      plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
35      plt.title('Gambar Original')
36      plt.axis('off')
37
38      # Gambar dengan deteksi tepi
39      plt.subplot(1, 3, 2)
40      plt.imshow(edges)
41      plt.title('Deteksi Tepi Gambar')
42      plt.axis('off')
43
44      # Gambar dengan segmentasi
45      plt.subplot(1, 3, 3)
46      plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
47      plt.title('Hasil Segmentasi')
48      plt.axis('off')
49
50      # Tampilkan jendela
51      plt.show()
```

Fungsi edge_detection pada implementasi ini bertujuan untuk menyoroti tepi objek dalam gambar. Berikut adalah penjelasan lebih rinci:

```
4
5   # Fungsi untuk melakukan deteksi tepi pada gambar
6   def edge_detection(image):
7       gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8       edges = cv2.Canny(gray, 50, 150)
9       return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)
10
```

- Konversi ke skala abu – abu (gray) : Proses dimulai dengan mengubah gambar warna ke skala abu-abu. Ini dilakukan untuk menyederhanakan informasi warna menjadi tingkat keabuan.
- Deteksi tepi dengan metode canny (cv2.canny) : Metode Canny digunakan untuk mendeteksi tepi dalam gambar. Fungsi ini memanfaatkan perubahan gradien intensitas piksel untuk menentukan lokasi tepi yang signifikan.
- Konversi hasil deteksi ke format RGB : Hasil deteksi tepi dalam format skala abu-abu dikonversi kembali ke format RGB agar dapat digabungkan dengan gambar asli dan hasil segmentasi.

4.2 Segmentasi Warna

Fungsi remove_background bertujuan untuk menghilangkan latar belakang gambar dan mempertahankan hanya objek dengan warna kuning. Berikut penjelasannya:

```
10
11  # Fungsi untuk melakukan segmentasi pada gambar
12  def remove_background(image):
13      hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
14      lower_yellow = np.array([20, 100, 100])  # Nilai batas rendah untuk warna kuning dalam format HSV
15      upper_yellow = np.array([30, 255, 255])  # Nilai batas tinggi untuk warna kuning dalam format HSV
16      mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
17      result = cv2.bitwise_and(image, image, mask=mask)
18      return result
19
```

- Konversi ke ruang warna HSV : Gambar awal dikonversi ke ruang warna HSV (Hue, Saturation, Value). HSV memisahkan informasi warna, kejenuhan, dan nilai kecerahan, memudahkan dalam segmentasi warna.
- Tentukan Range warna : Range warna ditentukan dalam format HSV. Fungsi ini memungkinkan kita untuk menentukan batas rendah dan tinggi untuk rentang warna yang diinginkan.
- Buat Mask untuk warna yang diinginkan : Mask dibuat dengan menggunakan nilai batas rendah dan tinggi. Mask ini menyoroti piksel yang memiliki nilai dengan warna yang diinginkan.
- Gunakan Bitwise AND untuk Menghapus latar belakang : Operasi bitwise AND digunakan untuk menggabungkan gambar asli dengan mask. Hasilnya adalah gambar yang hanya menampilkan objek dengan warna yang diinginkan dan latar belakangnya dihapus.

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Penggunaan Gambar Uji

Berikut adalah implementasi kode terhadap empat gambar buah – buahan dan dua gambar objek lain.

Gambar 1 : Implementasi kode untuk buah apel

Wajib > apel.py > remove_background

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

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi (hapus background) pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 0, 0])  # ambang batas bawah untuk warna merah
    upper_red = np.array([10, 400, 255])  # ambang batas atas untuk warna merah
    mask = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar original
original = cv2.imread('Wajib/apel.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna merah
segmented = remove_background(original)

# Tampilkan ketiga gambar
plt.figure(figsize=(10, 5))
```

```
# Tampilkan ketiga gambar
plt.figure(figsize=(10, 5))

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

# Gambar dengan deteksi tepi
plt.subplot(1, 3, 2)
plt.imshow(edges)
plt.title('Deteksi Tepi Gambar')
plt.axis('off')

# Gambar dengan segmentasi
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
plt.title('Segmentasi')
plt.axis('off')

# Tampilkan jendela
plt.show()
```

Gambar 2 : implementasi kode untuk gambar buah apel ke dua

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

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi (hapus background) pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 0, 0])  # ambang batas bawah untuk warna merah
    upper_red = np.array([10, 400, 255])  # ambang batas atas untuk warna merah
    mask = cv2.inRange(hsv, lower_red, upper_red)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar original
original = cv2.imread('Wajib/apel2.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna merah
segmented = remove_background(original)

# Tampilkan ketiga gambar
plt.figure(figsize=(10, 5))

# Gambar original
```

Gambar 3 : Implementasi kode untuk gambar buah pisang

Wajib > pisang.py > remove_background

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

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = np.array([20, 100, 100])  # Nilai batas rendah untuk warna kuning dalam format HSV
    upper_yellow = np.array([30, 255, 255])  # Nilai batas tinggi untuk warna kuning dalam format HSV
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar
original = cv2.imread('Wajib/pisang.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna kuning
segmented = remove_background(original)

# untuk menampilkan ketiga gambar dalam satu jendela
plt.figure(figsize=(10, 5))
```

apel.py | baskom.py | botol.py | apel2.py | pisang.py

Wajib > pisang.py > remove_background

```python
    # untuk menampilkan ketiga gambar dalam satu jendela
    plt.figure(figsize=(10, 5))

    # Gambar original
    plt.subplot(1, 3, 1)
    plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    plt.title('Gambar Original')
    plt.axis('off')

    # Gambar dengan deteksi tepi
    plt.subplot(1, 3, 2)
    plt.imshow(edges)
    plt.title('Deteksi Tepi Gambar')
    plt.axis('off')

    # Gambar dengan segmentasi
    plt.subplot(1, 3, 3)
    plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
    plt.title('Hasil Segmentasi')
    plt.axis('off')

    # Tampilkan jendela
    plt.show()
```

Gambar 4 : Implementasi untuk gambar buah lemon

Wajib > lemon.py > ...

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

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_yellow = np.array([20, 100, 100])  # Nilai batas rendah untuk warna kuning dalam format HSV
    upper_yellow = np.array([30, 255, 255])  # Nilai batas tinggi untuk warna kuning dalam format HSV
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar
original = cv2.imread('Wajib/lemon.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna kuning
segmented = remove_background(original)

# untuk menampilkan ketiga gambar dalam satu jendela
plt.figure(figsize=(10, 5))
```

Wajib > lemon.py > ...

```python
    # untuk menampilkan ketiga gambar dalam satu jendela
    plt.figure(figsize=(10, 5))

    # Gambar original
    plt.subplot(1, 3, 1)
    plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    plt.title('Gambar Original')
    plt.axis('off')

    # Gambar dengan deteksi tepi
    plt.subplot(1, 3, 2)
    plt.imshow(edges)
    plt.title('Deteksi Tepi Gambar')
    plt.axis('off')

    # Gambar dengan segmentasi
    plt.subplot(1, 3, 3)
    plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
    plt.title('Hasil Segmentasi')
    plt.axis('off')

    # Tampilkan jendela
    plt.show()
```

Gambar 5 : Implementasi kode untuk gambar botol minum berwarna hijau

Tambahan > botol.py > ...

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

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_green = np.array([40, 40, 40])  # Nilai batas rendah untuk warna hijau dalam format HSV
    upper_green = np.array([80, 255, 255])  # Nilai batas tinggi untuk warna hijau dalam format HSV
    mask = cv2.inRange(hsv, lower_green, upper_green)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar
original = cv2.imread('Tambahan/botol.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna kuning
segmented = remove_background(original)

# untuk menampilkan ketiga gambar dalam satu jendela
plt.figure(figsize=(10, 5))
```

Tambahan > botol.py > ...

```python
    # untuk menampilkan ketiga gambar dalam satu jendela
    plt.figure(figsize=(10, 5))

    # Gambar original
    plt.subplot(1, 3, 1)
    plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    plt.title('Gambar Original')
    plt.axis('off')

    # Gambar dengan deteksi tepi
    plt.subplot(1, 3, 2)
    plt.imshow(edges)
    plt.title('Deteksi Tepi Gambar')
    plt.axis('off')

    # Gambar dengan segmentasi
    plt.subplot(1, 3, 3)
    plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
    plt.title('Hasil Segmentasi')
    plt.axis('off')

    # Tampilkan jendela
    plt.show()
```

Gambar 6 : Implementasi kode untuk gambar baskom berwarna merah

Tambahan > baskom.py > ...

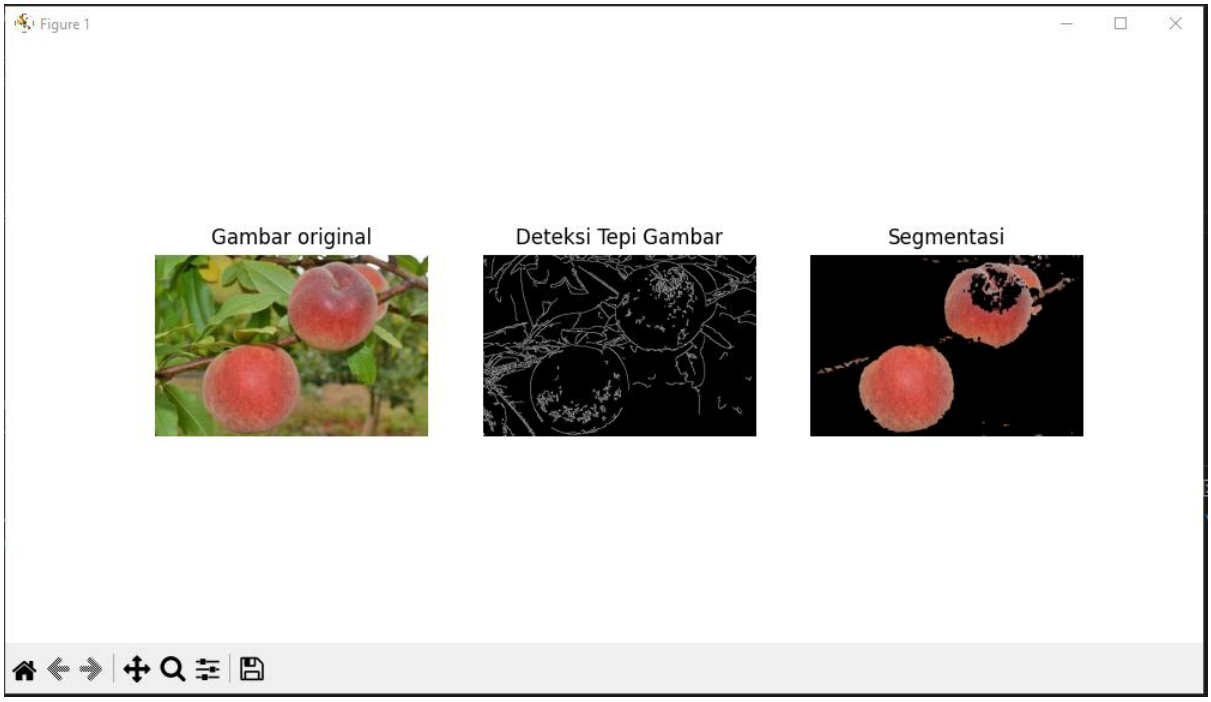
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk melakukan deteksi tepi pada gambar
def edge_detection(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)

# Fungsi untuk melakukan segmentasi pada gambar
def remove_background(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_red1 = np.array([0, 100, 100])  # Nilai batas rendah untuk warna merah dalam format HSV
    upper_red1 = np.array([10, 255, 255])  # Nilai batas tinggi untuk warna merah dalam format HSV
    lower_red2 = np.array([160, 100, 100])  # Nilai batas rendah alternatif untuk warna merah dalam format HSV
    upper_red2 = np.array([179, 255, 255])  # Nilai batas tinggi alternatif untuk warna merah dalam format HSV
    mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
    mask = cv2.bitwise_or(mask1, mask2)
    result = cv2.bitwise_and(image, image, mask=mask)
    return result

# Baca gambar
original = cv2.imread('Tambahan/baskom.jpg')

# Deteksi tepi pada gambar
edges = edge_detection(original)

# Hapus background pada gambar dengan segmentasi warna merah
segmented = remove_background(original)
```

```python

    # untuk menampilkan ketiga gambar dalam satu jendela
    plt.figure(figsize=(10, 5))

    # Gambar original
    plt.subplot(1, 3, 1)
    plt.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    plt.title('Gambar Original')
    plt.axis('off')

    # Gambar dengan deteksi tepi
    plt.subplot(1, 3, 2)
    plt.imshow(edges)
    plt.title('Deteksi Tepi Gambar')
    plt.axis('off')

    # Gambar dengan segmentasi
    plt.subplot(1, 3, 3)
    plt.imshow(cv2.cvtColor(segmented, cv2.COLOR_BGR2RGB))
    plt.title('Hasil Segmentasi')
    plt.axis('off')

    # Tampilkan jendela
    plt.show()

```

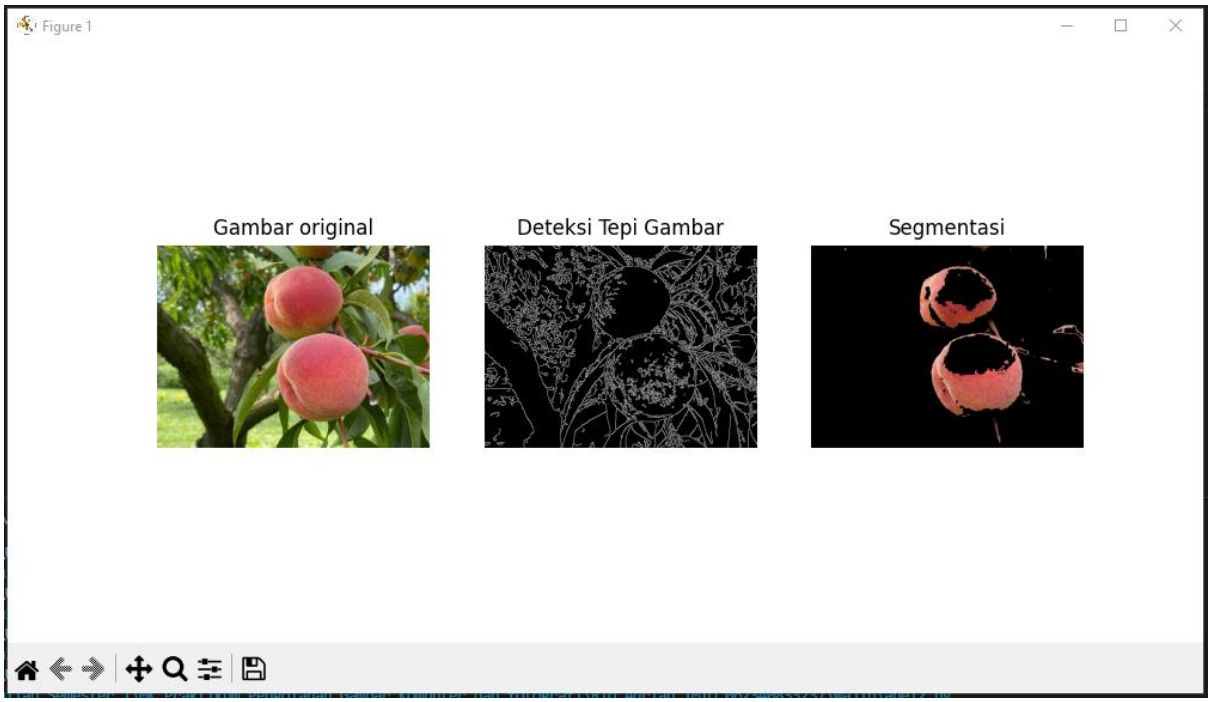
5.1.1 Hasil Implementasi

Berikut adalah hasil dari implementasi dari kode yang terlah diterapkan sebelumnya :

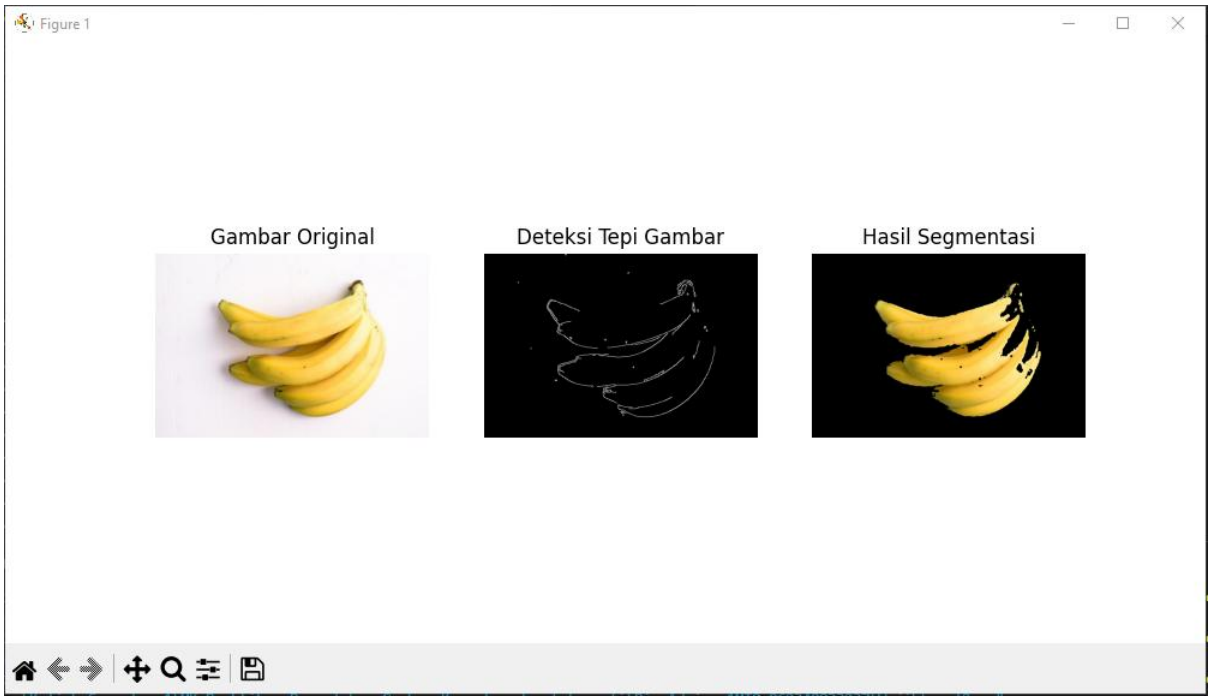
Gambar 1 : Output dari implementasi kode untuk buah apel pertama



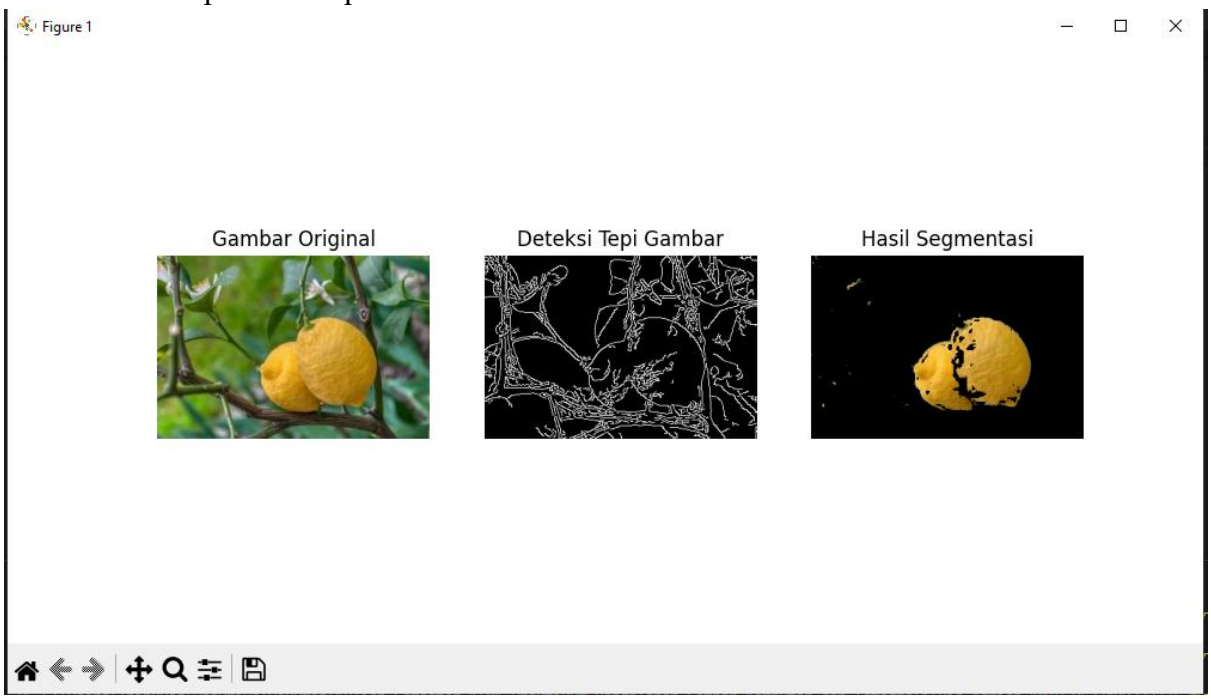
Gambar 2 : Output dari implementasi kode untuk buah apel kedua



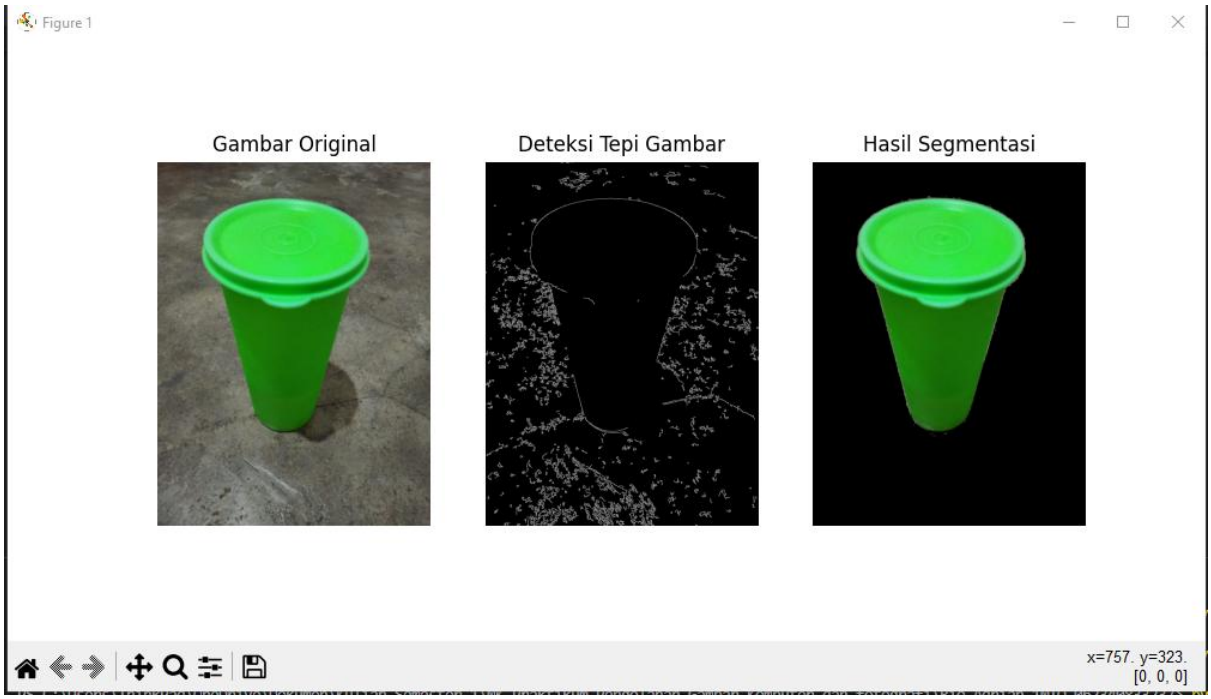
Gambar 3 : Output dari implementasi kode untuk buah pisang



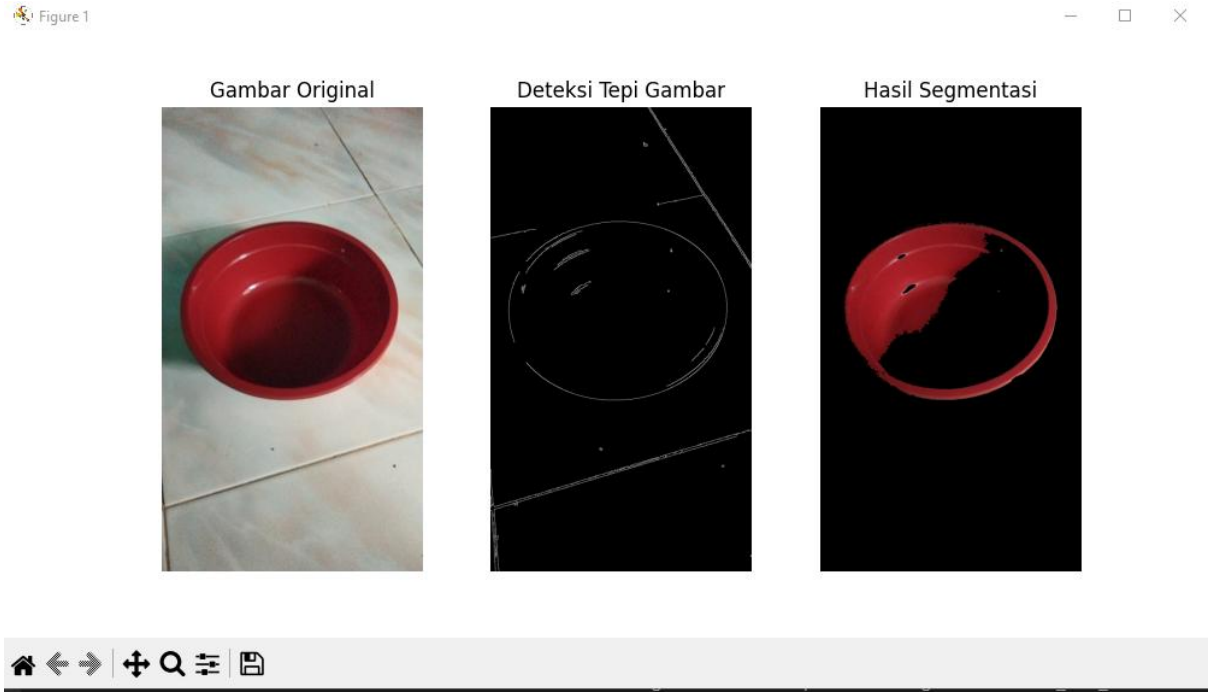
Gambar 4 : Output dari implementasi kode untuk buah lemon



Gambar 5 : Output dari implementasi kode untuk gambar botol minum



Gambar 6 : Output dari implementasi kode untuk gambar baskom merah



BAB VI

ANALISIS HASIL DAN FUNGSI PROGRAM

6.1 Analisis Hasil

Dari hasil implementasi deteksi tepi, dapat dilihat bahwa garis kontur pada gambar berhasil ditekankan dengan baik. Algoritma Canny mampu mengidentifikasi perubahan tajam dalam intensitas warna, sehingga garis-garis tepi pada objek di dalam gambar tampak jelas.

Segmentasi warna pada gambar berhasil menghapus latar belakang, menciptakan fokus pada objek dengan warna yang telah ditentukan. Namun, analisis lebih lanjut mengungkapkan bahwa terdapat beberapa bagian dari objek (buah) yang ikut terhapus selama proses segmentasi. Hal ini dapat disebabkan oleh variasi warna yang ada dalam objek itu sendiri, sehingga beberapa bagian dari buah yang memiliki warna serupa dengan latar belakang ikut terpengaruh.

Namun, perlu diingat bahwa tingkat kesuksesan deteksi tepi dan segmentasi juga tergantung pada kualitas dan karakteristik gambar uji. Ini dapat dibuktikan pada gambar botol dan juga baskom serta pisang yang tercetak hampir sempurna karena latar belakangnya yang tidak serumit buah apel dan lemon.

6.2 Analisis Fungsi Program

6.2.1 Deteksi Tepi

Fungsi deteksi tepi menggunakan metode Canny pada gambar. Berikut adalah analisis fungsi tersebut.

- Proses Konversi ke skala abu – abu (cv2.COLOR_BGR2GRAY)

  Langkah ini berguna untuk mengubah gambar berwarna menjadi skala abu-abu. Proses ini membantu mengurangi kompleksitas data, mempermudah analisis tingkat kecerahan piksel.
- Proses Deteksi Tepi (cv2.canny, 50,150)

  Fungsi Canny digunakan untuk mendeteksi tepi dalam gambar skala abu-abu. Parameter 50 dan 150 menentukan ambang batas untuk deteksi tepi. Nilai-nilai ini dapat disesuaikan untuk mengoptimalkan hasil deteksi.
- Konversi tepi ke warna RGB (cv2.COLOR_GRAY2RGB)

  Hasil deteksi tepi berupa gambar skala abu-abu. Proses ini mengonversi gambar tersebut kembali ke format warna RGB agar dapat diintegrasi dengan gambar asli untuk ditampilkan.

6.2.2 Segmentasi Warna

Fungsi segmentasi warna menghapus latar belakang gambar dan mempertahankan hanya bagian yang memiliki warna yang telah ditentukan. Berikut adalah analisis fungsi tersebut.

- Konversi ke model warna HSV (cv2.COLOR_BGR2HSV)

  Proses konversi gambar ke model warna HSV dilakukan untuk mempermudah segmentasi berbasis warna. Model warna HSV memisahkan informasi warna, kecerahan, dan kejenuhan.
- Penentuan rentang warna (lower dan upper)

  Rentang warna dalam format HSV ditentukan dengan mengatur nilai batas rendah **lower_color** dan nilai batas tinggi **upper_color**. Penyesuaian parameter ini mungkin diperlukan berdasarkan variasi warna pada gambar uji.
- Penggunaan Masking (cv2.inRange dan cv2.bitwise_and)

  Masking digunakan untuk menentukan area yang akan dihapus (latar belakang) berdasarkan rentang warna yang telah ditentukan. Operasi **bitwise_and** digunakan untuk mempertahankan bagian gambar yang sesuai dengan mask.
- Hasil Segmentasi (‘cv2.bitwise_and (image,image, mask = mask)’)

  Hasil akhir segmentasi adalah gambar yang hanya mempertahankan bagian yang memiliki warna yang telah ditentukan, sedangkan latar belakang dihapus.

6.3 Evaluasi Keseluruhan

Secara keseluruhan, implementasi deteksi tepi menggunakan metode Canny cukup efektif dalam menyoroti kontur objek pada gambar, menciptakan hasil yang memuaskan dengan mengidentifikasi perubahan tajam dalam intensitas warna. Namun, evaluasi segmentasi warna menunjukkan keberhasilan dalam menghapus latar belakang, tetapi tantangan muncul pada beberapa bagian objek yang ikut terhapus, mengindikasikan kesulitan dalam menangani variasi warna yang kompleks. Oleh karena itu, diperlukan penyesuaian parameter dan eksplorasi teknik segmentasi lanjutan untuk meningkatkan akurasi dan ketelitian dalam mempertahankan objek dengan variasi warna yang lebih kompleks.

BAB VII

PENUTUP

KESIMPULAN

Secara keseluruhan, proyek implementasi deteksi tepi dan segmentasi warna pada gambar telah mencapai hasil yang memuaskan dalam menyoroti kontur objek dan menghapus latar belakang dengan fokus pada warna yang telah ditentukan. Meskipun demikian, ketika fokus segmentasi diperluas untuk mencakup semua warna, terjadi tantangan dalam mempertahankan integralitas objek, menunjukkan kompleksitas dalam menangani variasi warna yang lebih luas. Pengembangan lebih lanjut diperlukan melalui penyesuaian parameter dan eksplorasi teknik segmentasi lanjutan guna meningkatkan ketelitian dan ketepatan segmentasi, serta mempertimbangkan opsi implementasi model pemelajaran mesin untuk meningkatkan adaptabilitas algoritma.