

# **BAB 09**

## **PEMROSESAN CITRA BINER**

Pokok bahasan pada bab ini mencakup hal-hal berikut:

- pengantar operasi biner;
- ekstraksi kontur;
- penghampiran kurva poligon;
- cara menghitung perimeter;
- cara menghitung luas;
- kotak pembatas;
- elips untuk objek;
- *convex hull*;
- *fitur suatu objek*;
- *momen*;
- *momenHu*;
- *pembandingan bentuk objek*;
- *perhitungan jarak kesamaan dua objek*.

## 9.1 Pengantar Operasi Biner

Beberapa pemrosesan citra mengacu pada citra biner. Sebagai contoh, dengan menggunakan citra biner, perbandingan panjang dan lebar objek dapat diperoleh. Di depan juga telah dibahas aplikasi citra biner pada morfologi. Namun, tentu saja masih banyak operasi lain yang memanfaatkan citra biner. Beberapa contoh diulas dalam bab ini.

## 9.2 Ekstraksi Kontur

Kontur menyatakan kurva yang menghubungkan piksel-piksel yang mempunyai intensitas atau warna sama. Pada OpenCV, detail setiap piksel yang berada pada kontur objek dapat diperoleh dengan menggunakan `cv2.findContours()`. Pemanggilannya seperti berikut:

```
citra, daftarKontur, hierarki = cv2.findContours(citra8iner,  
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

Dalam hal ini, *citraBiner* berupa citra biner dengan objek harus berwarna putih. Argumen kedua menyatakan bahwa kontur yang diperoleh adalah di bagian luar objek. Kontur di dalam objek tidak disertakan. Argumen ketiga menyatakan bahwa semua piksel pada kontur disertakan. Nilai baliknya sebanyak tiga. Nilai balik pertama berupa citra yang telah dimodifikasi. Nilai balik kedua berupa senarai kontur dengan setiap kontur berupa vektor titik. Nilai balik ketiga menyatakan hierarki kontur.

Skip berikut menunjukkan contoh untuk memperoleh kontur pada objek huruf yang terdapat pada berkas `huruf_bin.png`:



Berkas : `caikunci.11y`

```
# Contoh untuk mendapatkan kontur eksternal objek
```

```
# menggunakan findContours()

import cv2
import numpy as np

# Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

# Bentuk citra berwarna dengan latar belakang hitam
jumBaris = citra.shape[0]
jumKolom = citra.shape[1]
citraKontur = np.zeros((jumBaris, jumKolom, 3),
                       np.uint8)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
                                             cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Tambahkan kontur
cv2.drawContours(citraKontur, kontur, -1,
                 (230, 216, 173), 2)

# Tampilkan hasilnya
citraRGB = cv2.merge((citra, citra, citra))
hasil = np.hstack((citraRGB, citraKontur))
cv2.imshow("Hasil", hasil)
cv2.waitKey()
```

**Akhir berkas**

**Pada skrip ini, dua pernyataan berikut digunakan untuk memperoleh jumlah baris dan jumlah kolom pada citra yang berisi objek-objek huruf:**

```
jumBaris = citra.shape[0]
jumKolom = citra.shape[1]
```

**Adapun pernyataan berikut digunakan untuk membentuk citra berwarna yang mengandung tiga kanal dengan semuanya bernilai 0 (warna hitam):**

```
citraKontur = np.zeros((jumBaris, jumKolom, 3),
                       np.uint8)
```

**Argumen 3 pada np. zeros () inilah yang menyatakan terdapat tiga kanal. Argumen pertama menentukan jumlah baris dan argumen**

menentukan jumlah kolom. Pada citra inilah kontur hendak digambar dan memungkinkan pewarnaan pada kontur.

Perintah untuk mendapatkan kontur ditangani oleh:

```
citra2, kontur, hierarki = cv2.findContours(citra,  
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

Melalui perintah inilah senarai kontur akan diletakkan pada kontur.

Lalu, data pada senarai ini digambar pada citraKontur melalui:

```
cv2.drawContours(citraKontur, kontur, -1,  
(230, 216, 173), 2)
```

Dalam hal ini, argumen ketiga yang bernilai -1 menyatakan bahwa semua kontur diberi warna sama. Warna yang digunakan ditentukan pada argumen keempat. Tupel (230, 216, 173) menyatakan warna biru muda. Adapun argumen kelima yang berupa 2 menyatakan bahwa ketebalan kontur sebesar dua piksel.

Pernyataan berikut digunakan untuk membentuk citra berwarna dari citra berskala keabu-abuan:

```
citraRGB = cv2.merge((citra, citra, citra))
```

Hal ini diperlukan citra yang berskala keabu-abuan dapat digabungkan dengan citraKontur yang merupakan citra berwarna, melalui:

```
hasil = np.hstack((citraRGB, citraKontur))
```

Hasil pemanggilan skrip diperlihatkan pada Gambar 9.1. Tampak bahwa kontur internal pada huruf B dan D tidak digambar mengingat konstanta yang dilibatkan untuk memperoleh kontur berupa `cv2.RETR_EXTERNAL`.



*Gambar 9.1 Kontur internal pada objek-objek huruf"*

Apabila dikehendaki untuk menggunakan warna berbeda pada setiap kontur, argumen ketiga pada `drawContours()` perlu diganti dengan indeks kontur bersangkutan. Sebagai contoh,

```
cv2.drawContours(citraKontur, kontur, 0,
                 (230, 216, 173), 2)
```

membuat kontur objek pertama saja. Angka 0 berarti indeks untuk objek pertama.

Skrrip berikut memberikan gambaran mengenai cara untuk menampilkan setiap kontur dengan warna berbeda:



Berkas : Ci:rikunci.11y

```
# Pengaturan warna kontur

import cv2
import numpy as np

# Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

# Bentuk citra berwarna dengan latar belakang hitam
jumBaris = citra.shape[0]
jumKolom = citra.shape[1]
citraKontur = np.zeros((jumBaris, jumKolom, 3),
                       np.uint8)
```

```
# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Tambahkan kontur dengan warna berlainan
warna = [(230, 216, 173), (255, 255, 255),
      (255, 255, 0), (255, 0, 255),
      (0, 255, 255)]

for indeks in range(len(kontur)):
    cv2.drawContours(citraKontur, kontur, indeks,
        warna[indeks % 5], 2)

# Tampilkan hasilnya
citraRGB = cv2.merge((citra, citra, citra))
hasil = np.hstack((citraRGB, citraKontur))
cv2.imshow("Hasil", hasil)
cv2.waitKey()
```

**Akhir berkas**

**Pada skrip ini, perintah berikut digunakan untuk membentuk senarai yang mengandung lima tupel warna:**

```
warna = [(230, 216, 173), (255, 255, 255),
      (255, 255, 0), (255, 0, 255),
      (0, 255, 255)]
```

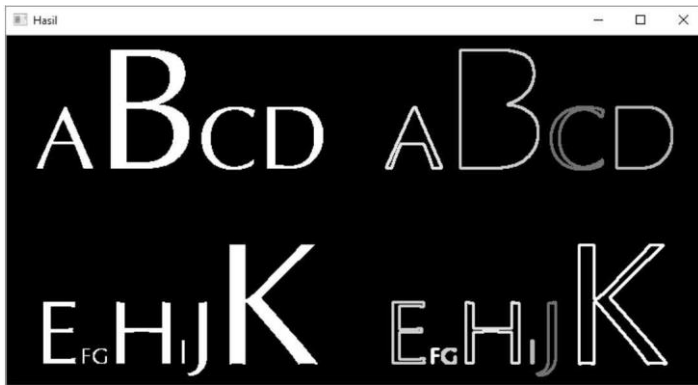
**Pengaturan warna untuk setiap kontur dilakukan melalui:**

```
for indeks in range(len(kontur)):
    cv2.drawContours(citraKontur, kontur, indeks,
        warna[indeks % 5], 2)
```

**Dalam hal ini, indeks akan bernilai dari 0 hingga jumlah kontur dikurangi**

**1. Karena pengaturan dilakukan secara individual untuk kontur masing-masing, argumen ketiga pada drawContours () diisi dengan nilai indeks. Adapun argumen keempat diisi dengan warna [ indeks % 5] . Penggunaan % 5 digunakan untuk memastikan indeks pada warna selalu dalam jangkauan 0 hingga 4.**

**Hasil skrip ini diperlihatkan pada Gambar 9.2.**



*Gambar 9\_g Setiap kontur digambar dengan warna berbeda*

Kontur internal yang terdapat pada suatu objek juga bisa diperoleh. Hal ini dilakukan dengan mengganti `cv2.RETR_EXTERNAL` dengan `cv2.RETR_LIST`. Jadi, perintah `findContours()` perlu diubah menjadi:

```
citra2, kontur, hierarki = cv2.findContours(citra,
cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
```

Hasilnya diperlihatkan pada Gambar 9.3.



*Gambar 9.3 Kontur internal dan eksternal digambar*

Aplikasi untuk memilih objek-objek huruf A hingga K yang memiliki panjang kontur tertentu diperlihatkan pada skrip berikut:

```
#Seleksi ukuran huruf

import cv2

#Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

#Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

#Tandai yang TIDAK terlalu besar atau terlalu kecil
minKontur    100
maxKontur    = 400

for indeks in range(len(kontur)):
    panjangKontur = len(kontur[indeks])
    if (panjangKontur >= minKontur) and\
        (panjangKontur <= maxKontur):
        cv2.drawContours(citraBerwarna, kontur, indeks,
            (255,255,0),3)

#Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)
cv2.waitKey()
```

### Akhir berkas

**Target skrip ini adalah menemukan lokasi huruf-huruf berukuran medium dan menandainya dengan gambar kontur. Hal ini dapat dilakukan dengan menggambar kontur yang memenuhi kriteria:**

```
(panjangKontur >= minKontur) and
(panjangKontur <= maxKontur)
```

**Dalam hal ini, nilai untuk minKontur dan maxKontur dapat diatur secara eksperimental seperti berikut:**

```
minKontur    100
maxKontur    400
```



Pada skrip ini, perintah berikut digunakan untuk memperoleh citra berwarna berdasarkan citra berskala keabu-abuan citra:

```
citraBerwarna = cv2.merge((citra, citra, citra))
```

Hal ini dimaksudkan agar warna kontur diberikan ke citraBerwarna. Penambahan kontur ke citra ini dilakukan melalui:

```
cv2.drawContours(citraBerwarna, kontur, indeks,  
                (255, 255, 0), 3)
```

Warna yang digunakan adalah cyan yang ditentukan melalui (255, 255, 0). Adapun ketebalan yang digunakan adalah tiga piksel. Hasilnya diperlihatkan pada Gambar 9.4. Dalam hal ini, huruf A, C, D, E, H, dan J saja yang memenuhi kriteria yang telah ditentukan.



*Gambar 9.4 Seleksi hanya untuk huruf J:-huruf[berukuran medium*

### 9.3 Penghampiran Kurva Poligon

Detail kontur dapat disederhanakan sehingga kurva yang dihasilkan mengandung titik-titik yang lebih sedikit. Sebagai contoh, objek berbentuk persegi panjang dapat diwakili oleh empat titik. Hal ini dapat

dilaksanakan dengan menggunakan `cv2.approxPolyDB()`. Bentuk pemakaiannya seperti berikut:

`cv2.approxPolyDB(kontur, epsilon, tertutup)`

Dalam hal ini, *kontur* menyatakan kurva hasil `findContours()`. Argumen kedua menentukan presisi penghampiran, yang menyatakan jarak maksimum antara kurva asal dan kurva hasil. Argumen ketiga berupa `True` atau `False`. Nilai `True` menyatakan kurva tertutup dan `False` menyatakan kurva terbuka.

Contoh penggunaan `approxPolyDB()` dapat dilihat pada skrip berikut:

**11!1** Berkas : penghampiran.py

```
# Contoh penggunaan approxPolyDB()

import cv2

# Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

print("Jumlah piksel semula:", len(kontur[0]))

epsilon= 0.01 * cv2.arcLength(kontur[0], True)
approx= cv2.approxPolyDP(kontur[0], epsilon, True)

print("Jumlah piksel sekarang:", len(approx))

cv2.drawContours(citraBerwarna, approx, -1,
      (255, 0, 255), 8)

cv2.imshow("Hasil", citraBerwarna)
waitKey()
```

**Akhir berkas**

Pada skrip ini, mula-mula jumlah piksel yang membentuk kontur ditampilkan melalui:

```
print("Jumlah piksel semula:", len(kontur[0]))
```

Kemudian, jarak maksimum antara kurva asal dan kurva hasil ditentukan melalui:

```
epsilon= 0.01 * cv2.arcLength(kontur[0], True)
```

Lalu, penghampiran kurva dilakukan melalui:

```
approx= cv2.approxPolyDP(kontur[0], epsilon, True)
```

Jumlah piksel yang membentuk kurva kontur ditampilkan melalui:

```
print("Jumlah piksel sekarang:", len(approx))
```

Selanjutnya, titik-titik penyusun kontur terbaru ini ditampilkan dengan menggunakan pernyataan:

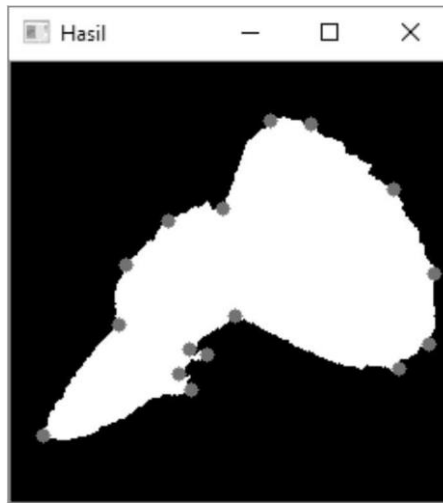
```
cv2.drawContours(citraBerwarna, approx, -1,  
                (255, 0, 255), 8)
```

Warna yang digunakan ditentukan oleh (255, 0, 255) dan ketebalan piksel yang digunakan sebesar 8.

Hasil pengujian skrip ditunjukkan berikut ini:

```
.....  
: C:\LatOpenCV>python penghampiran.py<P  
: Jumlah piksel semula: 675  
: Jumlah piksel sekarang: 16  
:.....!
```

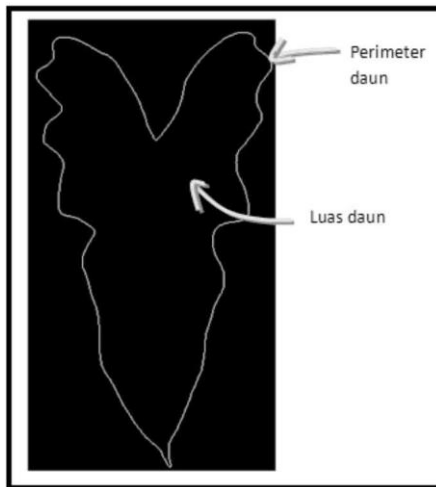
Gambar 9.5 menunjukkan titik-titik hasil penghampiran kontur.



*Gambar 9.5 Kontur hasil penghampiran*

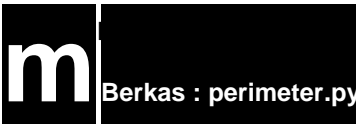
## 9.4 Cara Menghitung Perimeter

Perimeter sering digunakan sebagai bahan untuk mendapatkan fitur suatu objek. Perimeter atau keliling menyatakan panjang tepi suatu objek. Ilustrasinya ditunjukkan pada Gambar 9.6.



*Gambar 9.6 Perimeter dan luas objek*

Secara sederhana, perimeter dapat didekati dengan menerapkan fungsi `len ()` pada kontur untuk menghitung jumlah piksel. Hal inilah yang dilakukan pada skrip seleksi. `py`. Akan tetapi, OpenCV juga menyediakan `cv2. arcLength ()` yang menggunakan pendekatan berbeda dalam menghitung keliling kontur. Dalam hal ini, argumennya berupa kontur dan nilai logika yang menyatakan kontur dalam keadaan tertutup atau tidak. Nilai baliknya menyatakan keliling objek.



penggunaan `cv2. arcLength ()`:

```
# Perimeter

import cv2

# Baca citra biner
citra = cv2.imread("guppy-5.png", 0)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Peroleh perimeter
keliling = cv2.arcLength(kontur[0], True)

print("Keliling =", keliling)
```



**Penghitungan keliling citra guppy-5.png dilakukan oleh:**

```
keliling = cv2.arcLength(kontur[0], True)
```

**Hasil pengujian skrip ditunjukkan berikut ini:**

```
C:\LatOpenCV>python perimeter.py
Keliling = 811.2762540578842

C:\LatOpenCV>
```

Contoh berikut menunjukkan cara untuk memperoleh huruf yang mempunyai keliling terpanjang:



**Berkas : kelilingterpanjang.py**

```
# Huruf dengan keliling terpanjang

import cv2

# Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Cari yang kelilingnya terpanjang

for indeks in range(len(kontur)):
    keliling = cv2.arcLength(kontur[indeks], True)
    if indeks == 0:
        posisi = 1
        terpanjang = keliling
    else:
        if keliling > terpanjang:
            posisi = indeks
            terpanjang = keliling

# Proses untuk menghapus objek
for indeks in range(len(kontur)):
    if indeks != posisi:
        cv2.drawContours(citraBerwarna, kontur,
            indeks, (0, 0, 0), -1)

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)

cv2.waitKey()
```

**Akhir berkas**

**Secara prinsip semua keliling huruf diperoleh. Variabel terpanjang digunakan untuk mencatat indeks huruf yang memiliki keliling terpanjang dan variabel posisi menyatakan indeks huruf yang**

memiliki keliling terpanjang. Selanjutnya, dilakukan operasi penghapusan semua huruf pada `ci traBerwarna`, yang mewakili citra huruf `_bin. png` dalam bentuk citra berwarna, yang bukan merupakan huruf dengan keliling terpanjang. Proses penghapusan dilakukan melalui:

```
for indeks in range(len{kontur}):  
    if indeks != posisi:  
        cv2.drawContours{citraBerwarna, kontur,  
                           indeks, (0, 0, 0), -1)
```

Pada `drawContours ()`, `indeks` menyatakan indeks huruf yang hendak dihapus, `(0, 0, 0)` menyatakan warna hitam yang berefek pada penghapusan gambar, `-1` menyatakan bagian dalam kontur juga diwarnai dengan warna `(0, 0, 0)`.

Hasil skrip ditunjukkan pada Gambar 9.7.



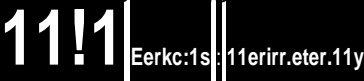
*Gambar 9.7 Huruf dengan perimeter terpanjang*

## 9.5 Cara Menghitung Luas

Luas menyatakan jumlah piksel yang terkandung dalam suatu objek.

Besaran ini dapat diperoleh dengan memanggil `cv2.contourArea()`. Argumen yang dilibatkan adalah kontur dan nilai logika yang menyatakan arah objek. Secara bawaan, argumen kedua bernilai `False`, yang memberikan nilai absolut luas tanpa menunjukkan arah (searah jarum jam atau berlawanan).

Contoh berikut menunjukkan penggunaan `cv2.contourArea()`:



```
# Perhitungan luas

import cv2

# Baca citra biner
citra = cv2.imread("guppy-5.png", 0)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
                                             cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Peroleh luas
luas = cv2.contourArea(kontur[0], False)

print("Luas =", luas)
```

Akhir berkas

Luas ikan pada citra `guppy-5.png` ditunjukkan berikut ini:

```
.....
: C:\LatOpenCV>python luas.py
Luas = 19135.5
```

```
C:\LatOpenCV>
```

Adapun skrip berikut menunjukkan cara untuk memperoleh huruf yang mempunyai luas terbesar:





```
# Huruf dengan luas terbesar

import cv2

# Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Cari yang luasnya terbesar

for indeks in range(len(kontur)):
    luas = cv2.contourArea(kontur[indeks])
    if indeks == 0:
        posisi = 1
        te_rbesar = luas
    else:
        if luas > terbesar:
            posisi = indeks
            terbesar = luas

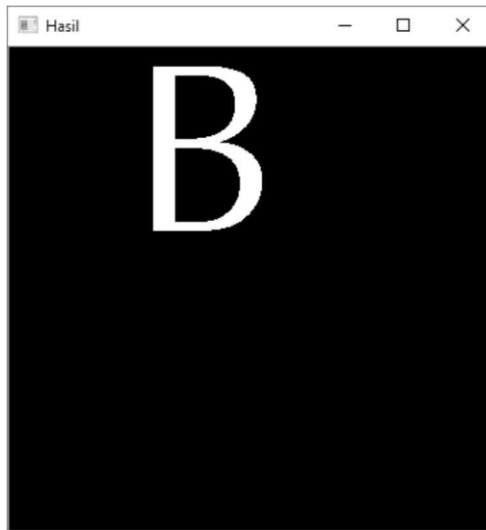
# Proses untuk menghapus objek
for indeks in range(len(kontur)):
    if indeks != posisi:
        cv2.drawContours(citraBerwarna, kontur,
            indeks, (0, 0, 0), -1)

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)

cv2.waitKey()
```

**Akhir berkas**

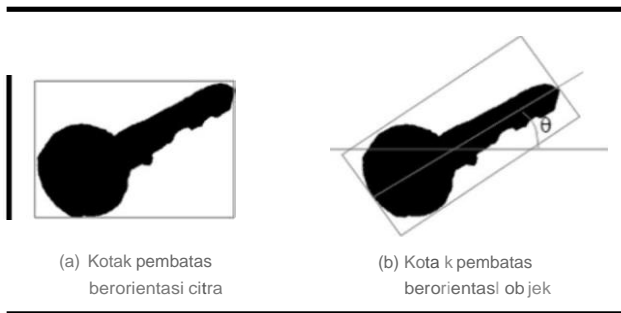
Secara prinsip, proses yang digunakan pada skrip ini serupa dengan pada keliling terpanjang. py. Hasilnya ditunjukkan pada Gambar 9.8.



*Gambar 9.8 Huruf dengan luas terbesar*

## 9.6 Kotak Pembatas

Kotak pembatas (*bounding box*) adalah kotak terkecil yang dapat melingkupi sebuah objek. Kotak pembatas dibedakan menjadi dua jenis: 1) kotak pembatas yang berorientasi citra atau kadang disebut kotak pembatas langsung dan 2) kotak pembatas yang berorientasi objek atau kadang dinamakan kotak pembatas terputar. Perbedaan kedua kotak pembatas ditunjukkan pada Gambar 9.9.



*Gambar 9.9 Perbedaan dua jenis kotak pembatas*

Kotak pembatas berorientasi citra milik suatu objek dapat diperoleh dengan menggunakan `cv2.boundingRect()`. Argumennya berupa kontur dan nilai baliknya berupa  $x$ ,  $y$ ,  $w$ , dan  $h$  dengan  $(x, y)$  menyatakan koordinat pojok kiri-atas kotak pembatas,  $w$  menyatakan lebar kotak dan  $h$  menyatakan tinggi kotak.

Kotak pembatas berorientasi objek diperoleh dengan menggunakan `cv2.minAreaRect()`. Argumennya berupa kontur dan nilai baliknya berupa struktur `Box2D` yang berisi data  $(x, y)$  menyatakan koordinat pusat kotak, *width* yang menyatakan lebar kotak, *height* menyatakan tinggi kotak, dan *angle* yang menyatakan sudut putaran kotak.

Contoh penggunaan baik `cv2.boundingRect()` maupun `cv2.minAreaRect()` dapat dilihat pada skrip berikut:

**I!JI** Berkas : kotakpembatas.py

```
# Kotak pembatas

import cv2
import numpy as np

# Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Peroleh kotak pembatas langsung
x, y, lebar, tinggi = cv2.boundingRect(kontur[0])

# Gambar kotak pembatas
cv2.rectangle(citraBerwarna, (x, y),
      (x + lebar, y + tinggi),
      (255, 255, 0), 2)

# Peroleh kotak pembatas terputar
persegiPanjang = cv2.minAreaRect(kontur[0])
```

```

kotak = cv2.boxPoints(persegiPanjang)
kotak = np.int32(kotak)
cv2.drawContours(citraBerwarna,[kotak], 0,
                 (128, 128, 128), 2)

```

```

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)

```

**Akhir berkas**

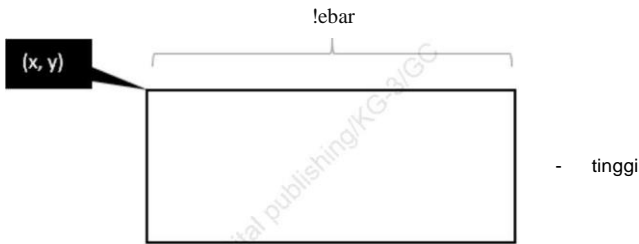
Pada skrip ini, kotak pembatas langsung diperoleh melalui:

```

x, y, lebar, tinggi = cv2.boundingRect(kontur[0])

```

**Hubungan kotak pembatas langsung terhadap keempat nilai balik yang diberikan diperlihatkan pada Gambar 9.10.**



**Gambar 9.10 Kotak pembatas langsung**

**Oleh karena itu, kotak pembatas ini digambar seperti berikut:**

```

cv2.rectangle(citraBerwarna, (x, y),
              (x + lebar, y + tinggi),
              (255, 255, 0), 2)

```

**Dalam hal ini, warna yang digunakan untuk menggambar kotak berupa (255, 255, 0) dan ketebalannya 2 piksel.**

**Adapun kotak pembatas terputar diperoleh melalui:**

```

persegiPanjang = cv2.minAreaRect(kontur[0])

```

**Untuk mendapatkan keempat koordinat persegipanjang dari persegipanj ang hasil minAreaRect (), supaya bisa digambar**

oleh `drawContours ()`, `cv2.boxPoints ()` diperlukan. Perintahnya berupa:

```
kotak = cv2.boxPoints{persegiPanjang}
```

Hasilnya perlu dikonversi ke bilangan bulat melalui:

```
kotak = np.int32{kotak}
```

Selanjutnya, data pada kotak digambar melalui:

```
cv2.drawContours{citraBerwarna,[kotak], 0,  
                (128, 128, 128), 2)
```

Dalam hal ini, warna yang digunakan untuk menggambar kotak berupa (128, 128, 128) atau warna abu-abu dan ketebalannya 2 piksel.

Hasilnya dapat dilihat pada Gambar 9.11.



*Gambar 9.11 Visualisasi kotak pembatas objek*

## 9.7 Lingkaran Pembatas

Lingkaran pembatas menyatakan lingkaran terkecil yang melingkupi suatu objek. Hal ini dapat diperoleh dengan menggunakan

```
cv2.minEnclosingCircle()).
```

Contoh

penggunaannya

**m**atkan pada skrip berikut:

**Berkas : kelilin:ter15anjan:11v**

```
#Lingkaran pembatas

import cv2
import numpy as np

#Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Peroleh lingkaran pembatas
(x, y), radius= cv2.minEnclosingCircle(kontur[0])

pusat = (int(x), int(y))
radius= int(radius)
cv2.circle(citraBerwarna, pusat,
      radius,
      (255, 255, 0), 2)

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)
waitKey()
```

**Akhir berkas**

**Pada skrip ini, pembatas lingkaran diperoleh:**

```
(x, y), radius= cv2.minEnclosingCircle(kontur[0])
```

**Berdasarkan data  $x$ ,  $y$ , dan  $radius$ , lingkaran dapat digambar dengan menyertakan pernyataan-pernyataan berikut:**

```
pusat = (int(x), int(y))
radius= int(radius)
cv2.circle(citraBerwarna, pusat,
      radius,
      (255, 255, 0), 2)
```

Dalam hal ini, warna yang digunakan untuk menggambar lingkaran berjari-jari radius dan pusat (x, y) berupa (255, 255, 0) atau warna abu-abu dan ketebalannya 2 piksel. Adapun `int()` digunakan untuk mengonversi data ke bilangan bulat.

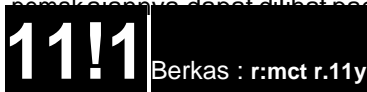
Hasil skrip dapat dilihat pada Gambar 9.12.



*Gambar 9.12 Lingkaran pembatas objek*

## 9.8 Elips untuk Objek

Elips untuk objek tidak benar-benar berupa elips yang membatasi objek, melainkan berupa elips terputar yang berada di atas objek. Hal ini bisa diperoleh dengan menggunakan `cv2.fitEllipse()`. Contoh pemakaiannya dapat dilihat pada skrip berikut:



```
#Elips untuk objek
```

```
import cv2
import numpy as np
```

```
#Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

#Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

#Peroleh elips
elips = cv2.fitEllipse(kontur[0])
cv2.ellipse(citraBerwarna, elips,
      (255, 255, 0), 2)

#Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)
waitKey()
```

**Akhir berkas**

**Pada skrip ini, pembatas lingkaran diperoleh:**

```
elips = cv2.fitEllipse(kontur[0])
```

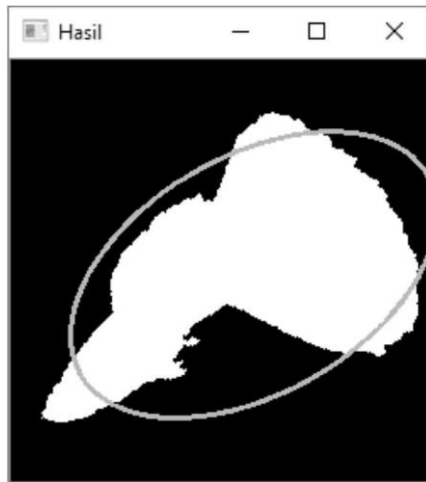
**Selanjutnya, hasil elips digambar melalui:**

```
cv2.ellipse(citraBerwarna, elips,
      (255, 255, 0), 2)
```

**Dalam hal ini, warna yang digunakan untuk menggambar elips berupa (255,255,0) atau warna abu-abu dan ketebalannya 2 piksel.**

**Hasil skrip dapat dilihat pada Gambar 9.13.**

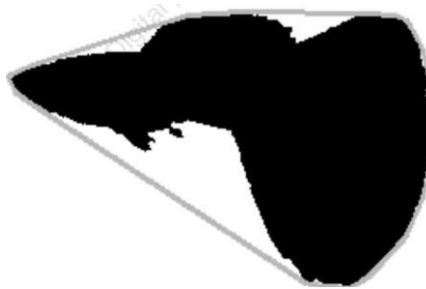




*Gambar 9.13 Elips untuk objek*

## 9.9 Convex Hull

*Convex hull* dapat dibayangkan seperti karet gelang yang melingkupi suatu benda. Contoh diperlihatkan pada Gambar 9.14.



*Gambar 9.14 Convex hull seperti karet gelang yang melingkupi objek*

*Convex hull* diperoleh dengan menggunakan `cv2.convexHull 1 ()`. Contoh pemakaiannya dapat dilihat pada skrip berikut:

```
# Convex hull

import cv2

# Baca citra biner huruf_bin.png
citra = cv2.imread("guppi-5.png", 0)

citraBerwarna = cv2.merge((citra, citra,citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Peroleh convex hull
convex= cv2.convexHull(kontur[0])

Jika: "convexhull.py" ke "convexhull.py"
cv2.drawContours(citraBerwarna, [convex], -1,
      (255, 255, 0), 2)

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)
cv2.waitKey()
```

**Akhir berkas**

Pada skrip ini, **convex hull** diperoleh:

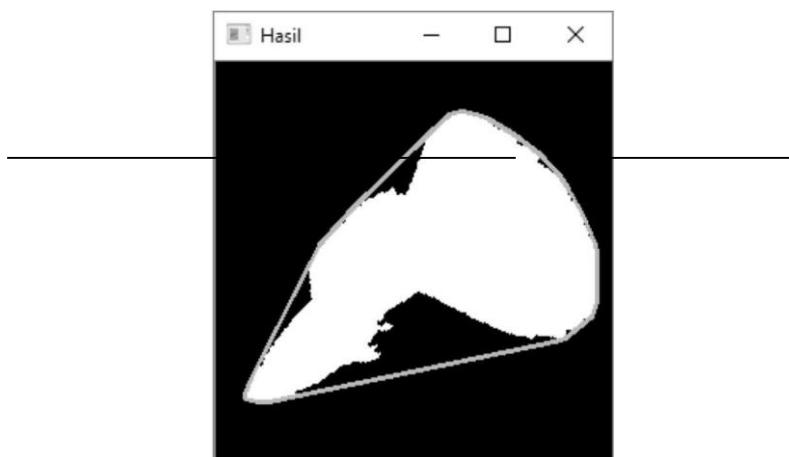
```
convex= cv2.convexHull(kontur[0])
```

Kemudian, hasilnya digambar melalui:

```
cv2.drawContours(citraBerwarna, [convex], -1,
      (255, 255, 0), 2)
```

Dalam hal ini, warna yang digunakan untuk menggambar kontur berupa (255, 255, 0) atau warna abu-abu dan ketebalannya 2 piksel.

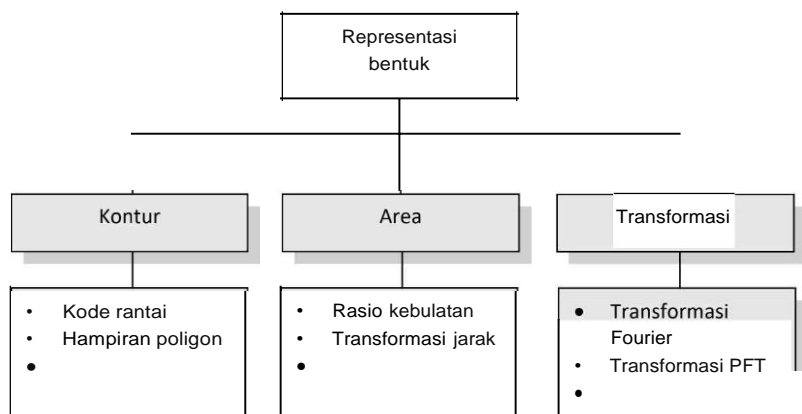
Hasil skrip diperlihatkan pada Gambar 9.15.



*Gambar9.15 Convex hull*

## 9.10 Fitur Suatu Objek

Fitur suatu objek merupakan karakteristik yang melekat pada objek. Fitur bentuk merupakan suatu fitur yang diperoleh melalui bentuk objek dan dapat dinyatakan melalui kontur, area, dan transformasi, sebagaimana ditunjukkan pada Gambar 9.16.



*Gambar 9.16 Representasi bentuk objek*

Sejumlah fitur yang diperoleh melalui sejumlah parameter yang telah dibahas di depan dikupas di sini dan ringkasannya dapat dilihat pada Tabel 9.1. Perlu diketahui, gabungan sejumlah fitur dapat digunakan untuk membedakan antara satu objek dengan objek lainnya.

**Tabel 9.1 Fitur-litur objek**

Fitur	Keterangan
Kebulatan bentuk	<p>Fitur ini merupakan perbandingan antara luas objek dan kuadrat perimeter objek:</p> $kebulatan = 4, \frac{luasObjek}{perimeterObjek^2}$ <p>Hasilnya berupa nilai yang lebih kecil daripada atau sama dengan 1. Nilai 1 menyatakan bahwa objek berbentuk lingkaran. Terkadang, fitur ini dinamakan kekompakan.</p>
Rasio aspek	<p>Fitur ini merupakan perbandingan antara lebar dengan panjang, yang dinyatakan dengan rumus seperti berikut :</p> $rasioAspek = \frac{lebar}{panjang}$ <p>Dalam hal ini, panjang adalah panjang objek dan lebar adalah lebar objek. Fitur ini terkadang disebut kerampingan bentuk.</p>
Konveksitas	<p>Fitur konveksitas didefinisikan seperti berikut :</p> $konveksitas = \frac{perimeterKonveks}{perimeterObjek}$
Soliditas	<p>Fitur ini dinyatakan seperti berikut :</p> $Soliditas = \frac{luasObjek}{luasKonveks}$
Rasio kotak pembatas berorientasi citra	<p>Fitur dinyatakan seperti berikut:</p> $rasio = \frac{luasObjek}{luasPersegipanjang}$

Fitur	Keterangan
Rasio kotak pembatas berorientasi objek	Fitur dinyatakan seperti berikut : $\text{rasio} = \frac{\text{luasObjek}}{\text{luasPersegiPanjang}}$

Contoh berikut menunjukkan cara menghitung keseluruhan fitur yang tercantum pada Tabel 0.1

**11!1** Berkas : cenv : dml.11y

```
# Contoh untuk mendapatkan
#   sejumlah fitur objek

import cv2
import math

# Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

kelilingObjek = cv2.arcLength(kontur[0], True)
luasObjek = cv2.contourArea(kontur[0], False)
x, y, lebar, tinggi = cv2.boundingRect(kontur[0])

# Peroleh convex hull
convex = cv2.convexHull(kontur[0])
kelilingKonveks = cv2.arcLength(convex, True)
luasKonveks = cv2.contourArea(convex, False)

# Hitung berbagai fitur
kebulatanBentuk = 4 * math.pi * \
      luasObjek / (kelilingObjek * kelilingObjek)

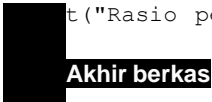
if tinggi > lebar:
    lebarObjek = lebar
    panjangObjek = tinggi
else:
    lebarObjek = tinggi
    panjangObjek = lebar

kerampinganBentuk = lebarObjek / panjangObjek
```

```
konveksitas = kelilingKonveks / kelilingObjek
soliditas = luasObjek / luasKonveks

rasioPembatasObjek = luasObjek / (lebar * tinggi)

print("Kebulatan bentuk      =", kebulatanBentuk)
print("Kerampingan bentuk    =", kerampinganBentuk)
print("Konveksitas bentuk     =", konveksitas)
print("Soliditas bentuk       =", soliditas)
print("Rasio pembatas objek   =", rasioPembatasObjek)
```





Pada skrip ini, *convex hull* diperoleh:




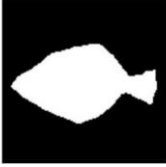
```
C:\LatOpenCV>python fiturobjek.py
Kebulatan bentuk      0.36535255578609793
Kerampingan bentuk    0.8251121076233184
Konveksitas bentuk     0.7775160173436593
Soliditas bentuk       0.7964662546044827
Rasio pembatas objek   0.46635552739325403

C:\LatOpenCV>
```

Tabel 9.2 memperlihatkan contoh sejumlah objek dan fitur-fiturnya.

*Tabel 9.12 Perbandingan fitur SEjumlah objek*

Citra	Fitur
	Kebulatan bentuk      = 0.37448231305916424 Kerampingan bentuk = 0.6473029045643154 Konveksitas bentuk  = 0.7870135317360417 Soliditas bentuk    = 0.7962488277586746 Rasio pembatas objek = 0.5081391637408235
	Kebulatan bentuk      = 0.37448231305916424 Kerampingan bentuk = 0.6473029045643154 Konveksitas bentuk  = 0.7870135317360417 Soliditas bentuk    = 0.7962488277586746 Rasio pembatas objek = 0.5081391637408235

Citra	Fitur
 guppy-3.png	Kebulatan bentuk = 0.37448231305916424 Kerampingan bentuk = 0.6473029045643154 Konveksitas bentuk = 0.7870135317360417 Soliditas bentuk = 0.7962488277586746 Rasio pembatas objek=0.5081391637408235
 guppy-4.png	Kebulatan bentuk = 0.37448231305916424 Kerampingan bentuk = 0.6473029045643154 Konveksitas bentuk = 0.7870135317360417 Soliditas bentuk = 0.7962488277586746 Rasio pembatas objek=0.5081391637408235
 guppy-5.png	Kebulatan bentuk = 0.36535255578609793 Kerampingan bentuk = 0.8251121076233184 Konveksitas bentuk = 0.7775160173436593 Soliditas bentuk = 0.7964662546044827 Rasio pembatas objek = 0.46635552739325403
 ikan-1.png	Kebulatan bentuk = 0.5462040255174448 Kerampingan bentuk = 0.5545454545454546 Konveksitas bentuk = 0.8932420585880085 Soliditas bentuk = 0.8571071696227829 Rasio pembatas objek = 0.5752421758569299

## 9.11 Momen

Momen adalah besaran kuantitatif yang digunakan untuk menjabarkan distribusi spasial sekumpulan titik yang membentuk objek. Karena menjabarkan distribusi spasial, momen ini biasa disebut pula momen spasial.

Momen spasial orde (m,n) didefinisikan sebagai berikut:

$$M_{mn} = \sum_{x=1}^M \sum_{y=1}^N x^m y^n I(x,y)$$

Dalam hal ini,

- $i, j = 0, 1, 2, \dots$ , dengan  $ij$  menyatakan orde momen;
- $M$  menyatakan jumlah kolom pada citra;
- $N$  menyatakan jumlah baris pada citra;
- $x$  adalah ordinat piksel;
- $y$  adalah absispiksel;
- $I(x,y)$  menyatakan intensitas piksel pada posisi  $(x, y)$ .

Berdasarkan momen, titik massa suatu objek atau dikenal dengan sebutan centroid dapat dihitung dengan menggunakan rumus berikut:

$$\bar{x} = \frac{M10}{M00}$$

$$\bar{y} = \frac{M01}{M00}$$

Terkait dengan hal ini, atribut-atribut berikut tersedia:

- **m00**
- **m10**
- **m01**
- m20
- **m11**
- m02
- m30
- m21



- m12
- m03

Contoh berikut menunjukkan cara menentukan centroid suatu objek:

11!1

Berkas :cenvexhull.11y

```
# Titik massa

import cv2

# Baca citra biner huruf_bin.png
citra = cv2.imread("huruf_bin.png", 0)

citraBerwarna = cv2.merge((citra, citra, citra))

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Visualisasi titik massa
for k in kontur:
    mom = cv2.moments(k)

    # Hitung titik massa
    x = int(mom["m10"] / mom["m00"])
    y = int(mom["m01"] / mom["m00"])

    cv2.circle(citraBerwarna, (x, y), 3,
      (255, 255, 0), -1)

# Tampilkan hasilnya
cv2.imshow("Hasil", citraBerwarna)
cv2.waitKey()
```

**Akhir berkas**

Perhitungan momen suatu objek k dilaksanakan melalui:

```
mom = cv2.moments(k)
```

Kemudian, centroid diperoleh dengan menggunakan:

```
x = int(mom["m10"] / mom["m00"])
y = int(mom["m01"] / mom["m00"])
```

Berdasarkan data (x, y), centroid digambar melalui:

```
cv2.circle(citraBerwarna, (x, y), 3,
           (255, 255, 0), -1)
```

Dalam hal ini, warna yang digunakan untuk menggambar lingkaran beradius 3 dengan pusat (x, y) berupa (255, 255, 0) atau warna abu-abu. Argumen -1 menyatakan lingkaran dipenuhi warna sesuai dengan (255, 255, 0).

Hasil skrip ini dapat dilihat pada Gambar 9.17. Titik-titik yang terlihat menyatakan titik massa setiap huruf.



*Gambar 9.17 Titik massa setiap huruf*

Adapun momen pusat adalah momen spasial yang dihitung relatif terhadap pusat massa. **Jika** pusat massa adalah (x, y), momen pusat ditulis seperti berikut:

$$\mu_{ij} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^i (y - \bar{y})^j I(x, y)$$

Momen ini bersifat invarian (tidak terpengaruh) terhadap translasi.  
 Terkait dengan hal ini, atribut-atribut berikut tersedia:

- mu20
- null
- mu02
- mu30
- mu21
- mu12
- mu03

Agar momen pusat bersifat bebas terhadap translasi, penyekalaan, dan rotasi, maka momen perlu dinormalisasi. Momen pusat ternormalisasi berupa:

$$nu_{ji} = \frac{c_{i+1}n_{i2}+1}{m00}$$

Terkait dengan momen ternormalisasi, atribut-atribut berikut tersedia:

- nu20
- null
- nu02
- nu30
- nu21
- nu12
- nu03

**Skrip berikut menunjukkan contoh untuk mendapatkan**

**momen**

**normalisasi pada suatu objek:**



**Berkas : momenternormalisasi.py**

```
#Momen ternormalisasi

import cv2

#Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

#Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

mom= cv2.moments(kontur[0])
nu20    mom["nu20"]
null1    mom["null1"]
nu02    mom["nu02"]
nu30    mom["nu30"]
nu21    mom["nu21"]
nul2    mom["nul2"]
nu03    mom["nu03"]

print("nu20  -"  nu20)
print("null1  -"  null1)
print("nu02  -"  nu02)
print("nu30  -"  nu30)
print("nu21  -"  nu21)
print("nul2  -"  nul2)
print("nu03  -"  nu03)
```

**Akhir berkas**

**Pada contoh ini, perintah seperti**

```
nu20 = mom["nu20"]
```

**digunakan untuk mendapatkan momen ternormalisasi nu20.**

**Hasil skrip ini seperti berikut:**





```
C:\LatOpenCV>python momenternormalisasi.py
nu20    0.16791576873842493
null    0.037461145052161825
nu02    0.0711770272642017
nu30    -0.04584406146063428
nu21    -0.011149271311942991
nu12    0.008025156404767463
nu03    0.010584425158314235
```

```
C:\LatOpenCV>
```

Tabel 9.3 memperlihatkan contoh sejumlah objek dan hasil momen ternormalisasi masing-masing.

*Tabel 9.5 Momen ternormalisasi untuk stifumlah objek*

Citra	Momen Ternormalisas
<p>guppy-1.png</p>	nu20 = 0.16791576873842493 null= 0.037461145052161825 nu02 = 0.0711770272642017 nu30 = -0.04584406146063428 nu21 = -0.011149271311942991 nu12 = 0.008025156404767463 nu03 = 0.010584425158314235
<p>guppy-2.png</p>	nu20 = 0.16791576873842493 null= -0.037461145052161665 nu02 = 0.0711770272642017 nu30 = 0.04584406146063458 nu21 = -0.01114927131194303 nu12 = -0.008025156404767565 nu03 = 0.010584425158314235
<p>guppy-3.png</p>	nu20 = 0.07117702726420187 null= 0.03746114505216191 nu02 = 0.16791576873842493 nu30 = -0.01058442515831469 nu21 = -0.008025156404767754 nu12 = 0.011149271311942821 nu03 = 0.04584406146063458

Citra	Momen Ternormalisas
 guppy-4.png	nu20 = 0.07117702726420187 null= -0.0374614505161665 nu02 = 0.16791576873842493 nu30 = -0.01058442515831469 nu21 = 0.008025156404767433 nu12 = 0.01149271311942745 nu03 = -0.04584406146063428
 guppy-5.png	nu20 = 0.15684528572562326 null= -0.04835308937200059 nu02 = 0.08197457626435728 nu30 = -0.015670662944296778 nu21 = 0.02659713431595698 nu12 = -0.011290796596980601 nu03 = -0.0004399823209866225
 ikan-1.png	nu20 = 0.1608549559560691 null= -0.0025307306426665677 nu02 = 0.045783871364706026 nu30 = 0.013331044664288937 nu21 = 0.0024233180084518463 nu12 = -0.0008432349504921509 nu03 = -0.0005614313859661277
 kunci.png	nu20 = 0.07294521019504348 null= -0.0016482291975787665 nu02 = 0.17792967174377836 nu30 = -0.0009802190662417515 nu21 = 0.01697653986632631 nu12 = 0.0010040134457949476 nu03 = -0.03404510656803336

## 9.12 Momen Hu

Momen Hu dikenal sebagai momen invarian karena sifatnya yang tidak terpengaruh oleh transformasi penyekalaan, translasi, rotasi, dan bahkan pencerminan dengan menghitung tujuh besaran terhadap suatu objek. Enam fitur pertama bersifat bebas dari penyekalaan, translasi,

dan rotasi, sedangkan fitur ketujuh mempunyai tanda berlawanan untuk objek yang dicerminkan.

Ketujuh fitur pada momen Hu dihitung dengan menggunakan rumus berikut:

$$\varnothing_1 = \eta_{20} + \eta_{02}$$

$$\varnothing_2 = (1J_{20} - 1J_{02})^2 + (21J_{02})^2$$

$$\varnothing_3 = (1J_{30} - 31J_{12})^2 + (1J_{03} - 31J_{21})^2$$

$$\varnothing_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2$$

$$\varnothing_5 = (1J_{30} - 31J_{12})(1J_{30} + 1J_{12})[(1J_{30} + 1J_{12})^2 - 3(1J_{21} + 1J_{03})^2] + \\ (1J_{03} - 31J_{12})(1J_{03} + 1J_{21})[(1J_{03} + 1J_{21})^2 - 3(1J_{12} + 1J_{30})^2]$$

$$\varnothing_6 = (1J_{20} - 1J_{02})[(1J_{30} + 1J_{12})^2 - (1J_{21} + 1J_{03})^2] + \\ 4(1J_{12} - 1J_{30})(1J_{03} + 1J_{21})$$

$$\varnothing_7 = (31J_{21} - 1J_{03})(1J_{30} + 1J_{12})[(1J_{30} + 1J_{12})^2 - 3(1J_{21} + 1J_{03})^2] + \\ (1J_{30} - 31J_{12})(1J_{21} + 1J_{03})[(1J_{03} + 1J_{21})^2 - 3(1J_{30} + 1J_{12})^2]$$

Dalam hal ini,  $\eta$  dwakili oleh nu. Rumus-rumus itulah yang digunakan pada `cv2.huMoments()`.

Skip berikut menunjukkan penggunaan `cv2.huMoments()`:



```
# Momen Hu

import cv2

# Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

mom= cv2.moments(kontur[0])
hu = cv2.HuMoments(mom)

print("hu[0]  -" , hu[0] [0])
print("hu[1]  -" , hu[1] [0])
print("hu[2]  -" , hu[2] [0])
print("hu[3]  -" , hu[3] [0])
print("hu[4]  -" , hu[4] [0])
print("hu[S]  -"   hu[SJ [0J)
t("hu[6]  -"   hu[6J [0J)
```

**Akhir berkas**

**Hasilnya seperti berikut:**

**C:\LatOpenCV>python momenhu.py**

```
hu[0J      0.23881986198998054
hu[1J      0.014957708142500008
hu[2J      0.006768378042359732
hu[3J      0.0014111169010839189
hu[4J      3.791292408143585e-06
hu[5J      0.00013959991674787148
hu[6J      2.155106688674846e-06
```

C:\LatOpenCV>

**Karena besaran ketujuh momen Hu mempunyai      jangkauan    yang  
terlampau luas, penggunaan logaritma sering dilakukan. Rumus    yang  
digunakan:**



$$H; = -\text{sign}(hu;) \log i hu; /$$



at dilihat pada skrip berikut:

```
# Momen Hu dengan penerapan logaritma

import cv2
import math

# Baca citra biner
citra = cv2.imread("guppi-5.png", 0)

# Peroleh kontur
citra2, kontur, hierarki = cv2.findContours(citra,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

mom= cv2.moments(kontur[0])
hu = cv2.HuMoments(mom)

# Terapkan logaritma
for i in range(0,7):
    hu[i] = -1 * math.copysign(1.0, hu[i]) * \
        math.log10(abs(hu[i]))

print("hu[0] -" hu[0] [0])
print("hu[1] -" hu[1] [0])
print("hu[2] -" hu[2] [0])
print("hu[3] -" hu[3] [0])
print("hu[4] -" hu[4] [0])
print("hu[5] -" , hu[5] [0])
print("hu[6] -" hu[6] [0])
```

**Akhir berkas**

Pada skrip ini, `import math` disertakan karena terdapat `copysign ()`, yang berada pada pustaka `math`. Secara umum, `math. copysign (x, y)` memberikan hasil berupa `-x` atau `+x` tergantung pada tanda bilangan `y`. Jika `y` negatif, hasilnya `-x`. Jika `y` positif, hasilnya `+x`.

Hasilnya seperti berikut:

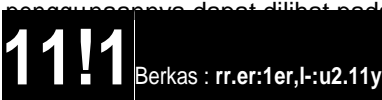
```
C:\LatOpenCV>python momenhu2.py<P
hu[0]    0.6219295569650309
hu[1]    1.8251349450621983
hu[2]    2.1695153921195125
hu[3]    2.8504370065201603
hu[4]    5.421212718779602
hu[5]    3.85511484070964
hu[6]    5.6665312251989075
```

```
C:\LatOpenCV>
```

Mengingat nilai yang dihasilkan melebihi 1, normalisasi perlu dilakukan terhadap momen Hu yang menggunakan pendekatan logaritma ini.

## 9.13 Pembandingan Bentuk Objek

Untuk kepentingan pembandingan dua objek, OpenCV menyediakan `cv2.matchShapes()`. Hasilnya berupa jarak dua objek didasarkan pada momen Hu. Dengan menggunakan fungsi ini, kesamaan bentuk dua gambar akan dihitung. Dua objek yang mempunyai bentuk yang tepat sama akan menghasilkan jarak sebesar 0. Semakin besar perbedaan bentuk kedua objek, nilai balik fungsi ini kian besar. Contoh pengaplikasiannya dapat dilihat pada skrip berikut:



```
# Pembandingan dua bentuk gambar
```

```
import cv2
```

```
# Baca citra biner
```

```
guppi1  cv2.imread("guppi-1.png", 0)
guppi2  cv2.imread("guppi-2.png", 0)
guppi3  cv2.imread("guppi-3.png", 0)
guppi4  cv2.imread("guppi-4.png", 0)
guppi5  cv2.imread("guppi-5.png", 0)
ikan1 = cv2.imread("ikan-1.png", 0)
```

```
#Peroleh kontur
jarakA = cv2.matchShapes(guppi2, guppi1,
                        cv2.CONTOURS_MATCH_1l, 0)
jarakB = cv2.matchShapes(guppi3, guppi1,
                        cv2.CONTOURS_MATCH_1l, 0)
jarakC = cv2.matchShapes(guppi4, guppi1,
                        cv2.CONTOURS_MATCH_1l, 0)
jarakD = cv2.matchShapes(guppi5, guppi1,
                        cv2.CONTOURS_MATCH_1l, 0)
jarakE = cv2.matchShapes(ikan1, guppi1,
                        cv2.CONTOURS_MATCH_1l, 0)

print("guppi-2 terhadap guppi-1 =", jarakA)
print("guppy-3 terhadap guppi-1 =", jarakB)
print("guppy-4 terhadap guppi-1 =", jarakC)
print("guppy-5 terhadap guppi-1 =", jarakD)
print("ikan-1 terhadap guppi-1 =", jarakE)
```

**Akhir berkas**

Pada `matchShapes()`, dua argumen pertama berupa citra yang hendak dibandingkan. Argumen ketiga menyatakan metode yang digunakan untuk melakukan perbandingan bentuk objek. Argumen keempat menyatakan suatu parameter. Namun, hingga kini belum didukung. Oleh karena itu, argumen keempat biasa diisi dengan 0.

Hasil pengujian skrip dapat dilihat berikut ini:

```
C:\LatOpenCV>python bandingbentuk.py<P
guppi-2 terhadap guppi-1      5.551115123125783e-17
guppy-3 terhadap guppi-1      5.551115123125783e-17
guppy-4 terhadap guppi-1      0.0
guppy-5 terhadap guppi-1      3.51810363747318e-05
ikan-1 terhadap guppi-1       0.006546341884136386
```

```
C:\LatOpenCV>
```

Hasil yang didapat menunjukkan bahwa jarak terbesar terletak pada perbandingan `ikan-1` dan `guppi-1`. Tentu saja, hasil ini masuk akal, mengingat kedua ikan pada kedua citra tersebut berbeda.

Metode yang digunakan untuk melakukan perbandingan bentuk objek dapat berupa `cv2.CONTOURS_MATCH_I1`, `cv2.CONTOURS_MATCH_I2`, atau `cv2.CONTOURS_MATCH_I3`. Perbedaan ketiga metode ini dapat dilihat pada Tabel 9.4.

**Tabel 9.4** Tiga metode pada `matchShapes()`

Metode	Keterangan
<code>cv2.CONTOURS_MATCH_I1</code>	Jarak dihitung menggunakan rumus : $I_1(A, B) = \sum_{i=0}^n  L_h u_i(A) - L_h u_i(B) $
<code>cv2.CONTOURS_MATCH_I2</code>	Jarak dihitung menggunakan rumus : $I_2(A, B) = \sum_{i=0}^n  L_h u_i(A) - L_h u_i(B) $
<code>cv2.CONTOURS_MATCH_I3</code>	Jarak dihitung menggunakan rumus : $I_3(A, B) = \max_{i=0 \dots n}  L_h u_i(A) - L_h u_i(B) $

## 9.14 Perhitungan Jarak Kesamaan Dua Objek

Kadangkala diperlukan untuk menghitung jarak kesamaan dua objek dengan melibatkan fitur-fitur lain dan bahkan menggunakan metode lain untuk pengukuran jarak kesamaan. Sebagai contoh, perhitungan jarak dapat dilakukan dengan menggunakan jarak Euclidean seperti berikut:

$$J(A,B)= \sqrt{\sum_{i=0}^{n-1} (Fitw_i(A)- Fitu_i(B))^2}$$

Skrip berikut menunjukkan penggunaan jarak Euclidean dengan melibatkan momen  $H_u$ , kerampingan bentuk, kebuatan bentuk, konveksitas, dan soliditas.



```
# Penggunaan jarak Euclidean
#      untuk menentukan kesamaan dua objek

import cv2
import math

def perolehFitur(berkasCitra):
    #Baca citra biner
    citra = cv2.imread(berkasCitra, 0)

    #Peroleh kontur
    citra2, kontur, hierarki = cv2.findContours(citra,
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    kelilingObjek = cv2.arcLength(kontur[0], True)
    luasObjek = cv2.contourArea(kontur[0], False)
    x,y, lebar, tinggi = cv2.boundingRect(kontur[0])

    #Peroleh convex hull
    convex= cv2.convexHull(kontur[0])
    kelilingKonveks = cv2.arcLength(convex, True)
    luasKonveks = cv2.contourArea(convex, False)

    #Hitung berbagai fitur
    kebulatanBentuk = 4 *math.pi*\
        luasObjek / {kelilingObjek * kelilingObjek)

    if tinggi > lebar:
        lebarObjek = lebar
        panjangObjek = tinggi
    else:
        lebarObjek = tinggi
        panjangObjek = lebar

    kerampinganBentuk = lebarObjek / panjangObjek
    konveksitas = kelilingKonveks / kelilingObjek
    soliditas = luasObjek / luasKonveks

    #Hitung Hu moment
    mom= cv2.moments(kontur[0])
    hu = cv2.HuMoments(mom)

    return kerampinganBentuk, kebulatanBentuk, \
        konveksitas, soliditas, hu

# -----
```

```

def jarakEuclidean(fX1, fX2, fX3, fX4, huX,
                   fY1, fY2, fY3, fY4, huY):
    jarak = (fX1 - fY1) ** 2 + (fX2 - fY2) ** 2 + \
            (fX3 - fY3) ** 2 + (fX4 - fY4) ** 2
    for indeks in range(0, 7):
        jarak = jarak + (huX[indeks] - huY[indeks]) ** 2

    jarak = math.sqrt(jarak)

    return jarak

# -----

# Program utama
ftA1, ftA2, fiA3, fiA4, huA    perolehFitur("guppi-1.png")
ftB1, ftB2, fiB3, fiB4, huB    perolehFitur("guppi-2.png")
fte1, fte2, fie3, fie4, hue    perolehFitur("guppi-3.png")
ftD1, ftD2, fiD3, fiD4, huD    perolehFitur("guppi-4.png")
ftE1, ftE2, fiE3, fiE4, huE    perolehFitur("guppi-5.png")
ftF1, ftF2, fiF3, fiF4, huF    perolehFitur("ikan-1.png")

jarakBA    jarakEuclidean(ftB1, ftB2, fiB3, fiB4, huB,
                           ftA1, ftA2, fiA3, fiA4, huA)
jarakeA    jarakEuclidean(fte1, fte2, fie3, fie4, hue,
                           ftA1, ftA2, fiA3, fiA4, huA)
jarakDA    jarakEuclidean(ftD1, ftD2, fiD3, fiD4, huD,
                           ftA1, ftA2, fiA3, fiA4, huA)
jarakeA    jarakEuclidean(ftE1, ftE2, fiE3, fiE4, huE,
                           ftA1, ftA2, fiA3, fiA4, huA)
jarakFA    jarakEuclidean(ftF1, ftF2, fiF3, fiF4, huF,
                           ftA1, ftA2, fiA3, fiA4, huA)

print("guppi-2 =", jarakBA)
print("guppi-3 =", jarakeA)
print("guppi-4 =", jarakDA)
print("guppi-5 =", jarakeA)
print("ikan-1 =", jarakFA)

```

### Akhir berkas

Pada skrip ini, perhitungan fitur ditangani oleh fungsi `perolehFitur()`. Fungsi memberikan nilai balik berupa fitur kerampingan objek, kebulatan objek, konveksitas, soliditas, dan tujuh momen Hu. Adapun perhitungan jarak Euclidean ditangani oleh fungsi `jarakEuclidean()`. Lima objek dibandingkan dengan objek yang berada pada `guppi-1.png`.

```
C:\LatOpenCV>python jarakeuclidean.py<P
guppi-2      4.421432798793072e-06
guppi-3      4.421432798793231e-06
guppi-4      1.8918060573493057e-16
guppi-5      0.17829692591778207
ikan-1 = 0.23276762555712144
```

```
C:\LatOpenCV>
```

Hasil yang didapat menunjukkan bahwa jarak terbesar terletak pada perbandingan ikan-1 dan guppi-1. Tentu saja, hasil ini masuk akal, mengingat kedua ikan pada kedua citra tersebut berbeda.