BAB 11

SEGMENTASI MENGGUNAKAN WATERSHED DAN GRABCUT

Bab ini membahas dua cara yang digunakan untuk melakukan segmentasi citra, yaitu:

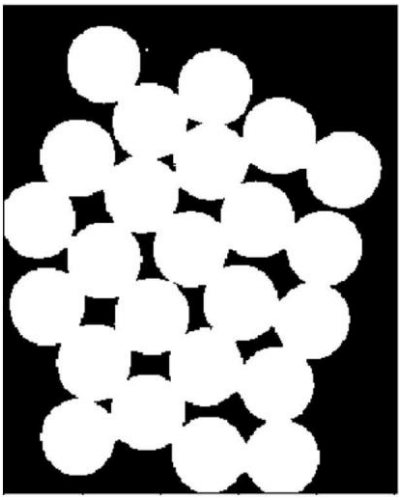
- Watershed dan
- GrabCut.

11.1 Pengantar Segmentasi Objek

Segmentasi objek adalah proses untuk memisahkan satu atau lebih dari satu objek terhadap latar belakang. Hal ini sebenarnya telah dibahas pada Bab 4.

Pada keadaan tertentu, segmentasi dengan cara yang telah dibahas tidaklah cukup. Hal ini terutama untuk objek-objek yang berhimpitan atau saling bertumpukan. Sebagai contoh, Gambar 11.1 menunjukkan sejumlah koin. Jika segmentasi dilakukan menggunakan metode Otsu, hasil yang didapat hanya satu objek seperti yang diperlihatkan pada Gambar 11.2.

Gambar 11.1 Contoh citra koinjpg dengan uang yang berhimpitan dan bertumpukan

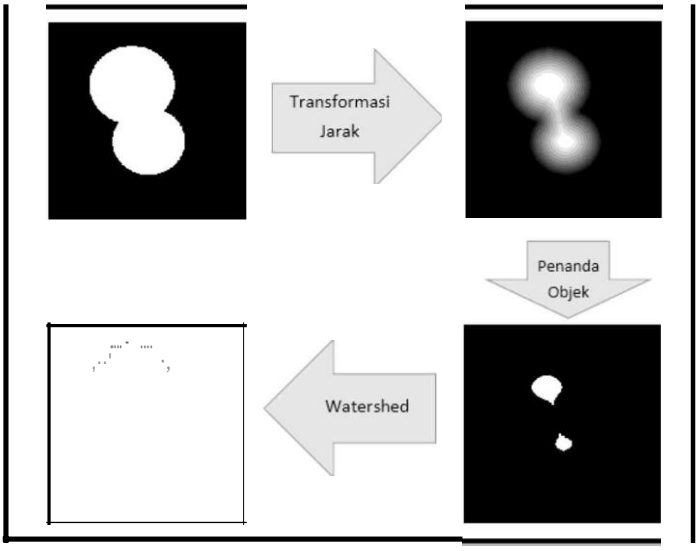


Gambar 11. Contoh hasil segmentasi koin dengan metode Otsu

Pada Subbab **8.4,** dijelaskan pula cara memisahkan dedaunan yang bersinggungan menggunakan transformasi morfologis. Namun, cara ini membuat dedaunan mengecil.

11.2 Segmentasi dengan Watershed

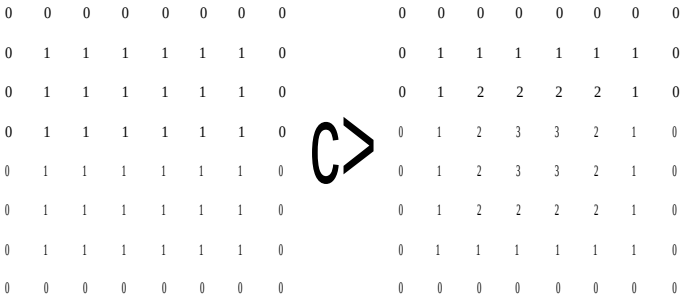
Watershed (batas air) adalah salah satu algoritma yang digunakan untuk memisahkan sejumlah objek yang berada dalam citra. Algoritma ini bekerja atas dasar penanda pada objek, yang memungkinkan dua objek yang bertumpukan dapat dipisahkan. Pertama-tama, penanda untuk setiap objek harus diberikan. Hal ini bisa dilakukan secara manual ataupun melalui suatu komputasi. Kemudian, berdasarkan penanda masing-masing, yang dapat dianggap sebagai suatu cekungan, dilakukan pembanjiran (terhadap cekungan) sehingga kedua batas dua objek yang bertumpukan atau bersinggungan bertemu.



Gambar 11.8 Proses pemisahan objek menggunakan watershed

Gambar 11.3 pada bagian kiri-atas memperlihatkan citra biner dua lingkaran yang bertumpukan. Dengan menggunakan transformasi jarak, diperoleh hasil seperti terlihat pada bagian kanan-atas. Untuk menandai kedua objek, suatu penanda perlu dibentuk. Hal ini dapat dilakukan dengan menerapkan nilai ambang tertentu sehingga diperoleh hasil seperti terlihat pada gambar bagian kanan-bawah. Masing-masing penanda itulah yang digunakan sebagai titik awal untuk penerapan watershed. Hasil akhirnya divisualisasikan pada gambar bagian kiri-bawah. Tampak bahwa kedua objek dapat dipisahkan.

Transformasi jarak adalah proses yang dikenakan pada citra biner untuk menghitung jarak setiap piksel pada area objek dihitung berdasarkan batas terdekat pada objek. Gambar 11.4 menunjukkan contoh transformasi jarak yang dikenakan pada objek kotak. Hasilnya berupa citra berkala keabu-abuan.



Gambar 11.4 Transformasijarak terhadap citra biner

Di OpenCV, transformasi jarak dapat diperoleh dengan menggunakan cv2. distanceTransform (). Bentuk pemanggilannya seperti berikut:

distanceTransform(citra, *tipelarak, ukuranCadar)*

Argumen pertama berupa sumber citra yang diproses. Argumen kedua berupa jenis operator yang digunakan untuk melakukan transformasi jarak. Konstanta yang menentukan cara menghitung jarak, antara lain:

- cv2.DIST_Ll Jarak= lxl-x21 + lyl-y21
- cv2.DIST_L2 Jarak Euclidean
- cv2.DIST_C Jarak = max(lxl-x2l, lyl-y2l)

Argumen ketiga berupa ukuran cadar yang digunakan untuk melakukan transformasi jarak. Nilai n menyatakan ukuran cadar sebesar n x n. konstanta cv2. DIST MASK 3 dan cv2. DIST MASK 5 disediakan di OpenCV. Nilai balik berupa larik berdimensi dua.

Skrip berikut memperlihatkan cara untuk melakukan segmentasi terhadap koin-koin yang terdapat pada Gambar 11.1:

Berkas : koin.py

```
# Pemrosesan koin
#
#    Adaptasi contoh di OpenCV
#    dengan menampilkan secara visual
#    hasil setiap langkah

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

citra = cv2.imread("coins.jpg")
if citra is None:
    print("Berkas citra tidak ditemukan")
    exit()

plt.subplot(241)
plt.imshow(citra[...,::   -1])
plt.xticks([]), plt.yticks([])
plt.title('Citra Asal')

abuAbu = cv2.cvtColor(citra, cv2.COLOR_BGR2GRAY)
ambang, citraBiner = cv2.thresholcl(abuAbu    , 0, 255,
         cv2.THRESH BINARY INV+ cv2.THRESH OTSU)

plt.subplot(242)
plt.imshow(citraBiner, cmap = "gray", vmin    0, vmax
255)
plt.xticks([]), plt.yticks([])
plt.title('Citra Siner')

#Hapus derau
kernel= np.ones((3, 3), np.uint8)
pembukaan = cv2.morphologyEx(citraBiner,
    cv2.MORPH_OPEN, kernel, iterations= 2)

plt.subplot(243)
plt.imshow(pembukaan, cmap = "gray",
           vmin = 0, vmax = 255)
plt.xticks([]), plt.yticks([])
plt.title('Pembukaan')

#Untuk memastikan daerah latar belakang
latarBelakang = cv2.dilate(pembukaan, kernel,
                           iterations= 2)

plt.subplot(244)
```

```
plt.imshow{latarBelakang, cmap = "gray",
           vmin = O, vmax = 255)
plt.xticks{[]), plt.yticks{[])
plt.title{'Latar Belakang')

# Untuk mendapatkan daerah latar depan
transformJarak = cv2.distanceTransform{pembukaan,
                     cv2.DIST_L2, cv2.DIST_MASK_5)
ambang, latarDepan = cv2.threshold{transformJarak,
   0.7 * transformJarak.max{), 255, cv2.THRESH_BINARY)

plt.subplot(245)
plt.imshow{latarDepan, cmap = "gray",
           vmin = O, vmax = 255)
plt.xticks{[]), plt.yticks{[])
plt.title{'Latar Depan')

# Untuk mendapatkan area yang tidak dikenal
latarDepan = np.uint8{latarDepan)
daerahTakBertuan = cv2.subtract(latarBelaka   ng,
                                  latarDepan)

plt.subplot(246)
plt.imshow {daerahTakBertuan, cmap   "jet")
plt.xticks{[]), plt.yticks{[])
plt.title{'Tak Bertuan')

# ----- Pelabelan pada penanda
jumObjek, penanda = cv2.connectedComponents(latarDepan)
print("Jumlah koin:", jumObjek - 1)

# Tambahkan 1 untuk semua penanda
#    supaya label untuk latar belakang berupa 1
penanda = penanda + 1

# Ubah penanda pada daerah tak bertuan dengan 0
penanda[daerahTakBertuan == 255] = 0

# ----- Penerapan watershed
penanda = cv2.watershed{citra, penanda)

plt.subplot(247)
plt.imshow{penanda, cmap = "jet")
plt.xticks{[]), plt.yticks([])
plt.title('Penanda')

# Beri tanda batas pada citra asal
citra[penanda == -1] = [255, 0, OJ
```

```
plt.subplot(248)
plt.imshow(citra[..., : : -1])
plt.xticks([]), plt.yticks([])
plt.title('Hasil Akhir')
```

Akhir berkas `tampilkan`

Untuk mendapatkan berkas coins. jpg, Anda bisa menyalinnya di folder:

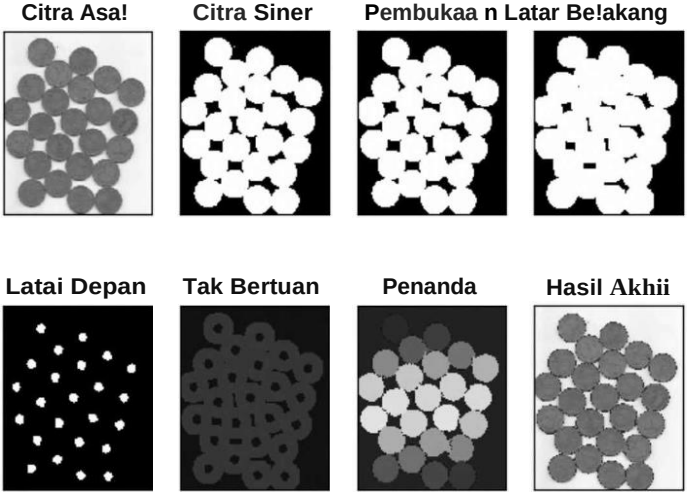
Downloads\opencv\sources\doc\js_tutorials\js_asset

Berikut adalah hasil eksekusi skrip koin.py:

```
C:\LatOpenCV>python koin.py
Jumlah koin: 24
```

- C: \LatOpenCV>

Gambar 11.5 menunjukkan gambar-gambar yang terbentuk. Gambar yang berjudul "Penanda" menyatakan penanda objek. Setiap objek diberi warna yang berbeda.



Gambar 11.5 Proses segmentasi koin

Pada skrip koin. py, **modul** pyplot **digunakan untuk menyajikan sejumlah gambar. Oleh karena itu, terdapat kode:**

```
from matplotlib import pyplot as plt
```

Pernyataan

```
abuAbu = cv2.cvtColor(citra, cv2.COLOR_BGR2GRAY)
```

digunakan untuk mengonversi citra yang berupa citra berwarna ke citra berskala keabu-abuan. Hasil yang didapat diubah ke citra biner melalui:

```
ambang, citraBiner = cv2.threshold(abuAbu, 0, 255,
         cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

Metode Otsu digunakan untuk menangani konversi ke citra biner. Konstanta cv2.THRESH_BINARY_INV **digunakan dengan maksud agar objek dinyatakan dengan warna putih dan latar belakang berwarna hitam.**

Pada hasil citra biner, kalau diperhatikan dengan saksama, akan terlihat keberadaan noktah-noktah kecil yang bertindak sebagai derau. Derau-derau yang muncul bisa dihilangkan melalui transformasi morfologis yang dinamakan pembukaan. Untuk keperluan inilah, pada skrip terdapat dua pernyataan berikut:

```
kernel= np.ones((3, 3), np.uint8)
pembukaan = cv2.morphologyEx(citraBiner,
    cv2.MORPH_OPEN, kernel, iterations= 2)
```

Pertama-tama, kernel berukuran 3 x 3 yang semua elemennya bernilai 1 dibentuk. Kemudian, transformasi pembukaan dengan menggunakan kernel tersebut dilakukan dua kali.

Selanjutnya, terdapat operasi dilasi untuk memperbesar hasil transformasi pembukaan, yaitu:

```
latarBelakang = cv2.dilate(pembukaan, kernel,
                           iterations= 2)
```

Hal ini untuk memastikan bahwa hasil yang didapat benar-benar berfungsi sebagai latar belakang.

Untuk mendapatkan objek yang digunakan sebagai penanda pada *watershed,* transformasi jarak dihitung melalui:

```
transformJarak = cv2.distanceTransform(pembukaan,
                    cv2.DIST_L2, cv2.DIST_MASK_5)
```

Dengan demikian, transformJarak **berupa larik yang berisi jarak setiap piksel dalam objek terhadap batas objek. Agar objek-objek yang bersinggungan terpisah, segmentasi diterapkan melalui:**

```
ambang, latarDepan = cv2.threshold(transformJarak,
   0.7 * transformJarak.max(), 255, cv2.THRESH_BINARY)
```

Hasil pada latarDepan **ini akan membuat objek-objek berukuran kecil muncul. Objek-objek inilah yang akan dijadikan sebagai penanda awal pada *watershed.* Perlu diketahui,** `0.7 * transformJarak.max()` **berarti "0,7 x nilai tertinggi pada transformJarak".**

Selisih antara nilai latarBelakang **dan** latarDepan **merupakan area tak bertuan untuk sementara waktu. Daerah ini dihitung melalui:**

```
daerahTakBertuan = cv2.subtract(
                       latarBelakang, latarDepan)
```

Sebenarnya, daerah tak bertuan ini adalah bagian dari objek-objek yang akan terbentuk setelah *watershed* dikenakan.

Objek-objek pada latarDepan **perlu diberi label (berupa angka urut). Pelabelan dilaksanakan dengan menggunakan perintah:**

```
jumObjek, penanda = cv2.connectedComponents(latarDepan)
```

Dengan cara seperti ini, penanda berisi penanda setiap objek yang digunakan untuk melakukan operasi *watershed.* Hasilnya, bagian latar belakang dilabeli dengan 0.

Supaya latar belakang diberi label 1, maka semua label perlu dinaikkan sebesar 1. Hal ini dilakukan melalui:

```
penanda = penanda + 1
```

Selanjutnya, label 0 digunakan untuk area yang tidak bertuan. Hal ini dilakukan melalui:

```
penanda[daerahTakBertuan == 255] = 0
```

Nah, penanda inilah yang dikirim ke watershed () bersama citra. Lalu, hasilnya disimpan ke penanda kembali. Perhatikan hasilnya pada Gambar 11.5 yang berjudul "Penanda".

Terakhir, tanda batas setiap objek ditambahkan ke citra asal melalui:

```
citra[penanda == -1] = [255, 0, OJ
```

Dalam hal ini, warna biru ([255, 0, 01) digunakan.

Bagaimana halnya, kalau dikehendaki untuk memroses setiap objek yang terdapat pada citra? Sekali lagi, pada contoh di atas, penanda dan j umObj ek bisa digunakan. Contoh dapat dilihat pada skrip ikut:

Berkas : koin2.py

```
# Pemrosesan koin
#
#    Adaptasi contoh di OpenCV
#    dengan menampilkan secara visual
#    basil setiap langkah

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

citra = cv2.imread("coins.jpg")
if citra is None:
    print("Berkas citra tidak ditemukan")
    exit()
```

```
abuAbu = cv2.cvtColor(citra, cv2.COLOR_BGR2GRAY)
ambang, citraBiner = cv2.threshold(abuAbu, 0, 255,
         cv2.THRESH_BINARY_INV+ cv2.THRESH_OTSU)

# Hapus derau
kernel= np.ones((3, 3), np.uint8)
pembukaan = cv2.morphologyEx(citraBiner,
    cv2.MORPH_OPEN, kernel, iterations= 2)

# Untuk memastikan daerah latar belakang
latarBelakang = cv2.dilate(pembukaan, kernel, iterations
  2)

# Untuk mendapatkan daerah latar depan
transformJarak = cv2.distanceTransform(pembukaan,
                                    cv2.DIST_L2,
cv2.DIST_MASK_5)
ambang, latarDepan    cv2.threshold(transformJarak,
   0.7 * transformJarak.max(), 255, cv2.THRESH_BINARY)

# Untuk mendapatkan area yang tidak dikenal
latarDepan = np.uintS(latarDepan)
daerahTakBertuan    cv2.subtract(latarBe lakang,
latarDepan)

plt.subplot(246)
plt.imshow(daer ahTakBertuan, cmap    "jet")
plt.xticks([]), plt.yticks([])
plt.title('Tak Bertuan')

# ----- Pelabelan pada penanda
jumObjek, penanda = cv2.connectedComponents(latarDepan)
print("Jumlah koin:", jumObjek - 1)

# Tambahkan 1 untuk semua penanda
#    supaya label untuk latar belakang berupa 1
penanda = penanda + 1

# Ubah penanda pada daerah tak bertuan dengan 0
penanda[daerahTakBertuan == 255] = 0

# ----- Penerapan watershed
penanda = cv2.watershed(citra, penanda)

# Beri tanda batas pada citra asal
citra[penanda == -1] = [255, 0, OJ

# ----------------------------------------
tinggi, lebar = citra.shape[:2]
```

```
for indeks in range(1, jumObjek):
    cadar = np.zeros((tinggi, lebar), np.uint8)
    cadar[penanda == indeks + 1] = 1
    objek    citra * cadar[:, :, np.newaxis]
    objek = objek[..., : : -1]

    plt.subplot(3, 10, indeks)
    plt.imshow(objek)
    plt.xticks([]), plt.yticks([])
    plt.title(indeks)

show()

Akhir berkas
```

Bagian awal skrip ini sama seperti pada koin. py,dengan bagian untuk menampilkan citra dibuang. Tambahan pertama berupa:

```
tinggi, lebar = citra.shape[:2]
```

Pernyataan ini digunakan untuk mendapatkan tinggi dan lebar citra, Selanjutnya, kedua informasi ini akan digunakan untuk membentuk citra cadar yang seluruh elemennya bernilai 0.

Pernyataan berikut digunakan untuk memroses setiap koin:

```
for indeks in range(1, jumObjek):
```

Pada setia iterasi di for, perintah berikut digunakan untuk membentuk larik cadar dengan seluruh elemen bernilai nol:

```
cadar = np.zeros((tinggi, lebar), np.uint8)
```

Kemudian, pernyataan

```
cadar[penanda == indeks + 1] = 1
```

dimaksudkan untuk membuat elemen-elemen di cadar **yang posisinya sama dengan elemen-elemen di penanda yang bernilai** indeks + **1 (label untuk objek) diisi dengan 1. Dengan demikian, hanya satu koin yang akan muncul pada** cadar.

Selanjutnya, pada cadar digunakan untuk membuat obj ek berisi satu koin sesuai dengan koin yang tercatat di cadar. Hal ini dilaksanakan melalui:

```
objek =  citra * cadar[:, :, np.newaxis]
```

Perintah ini identik dengan:

```
objek = cv2.bitwise_and(citra, citra, mask= cadar)
```

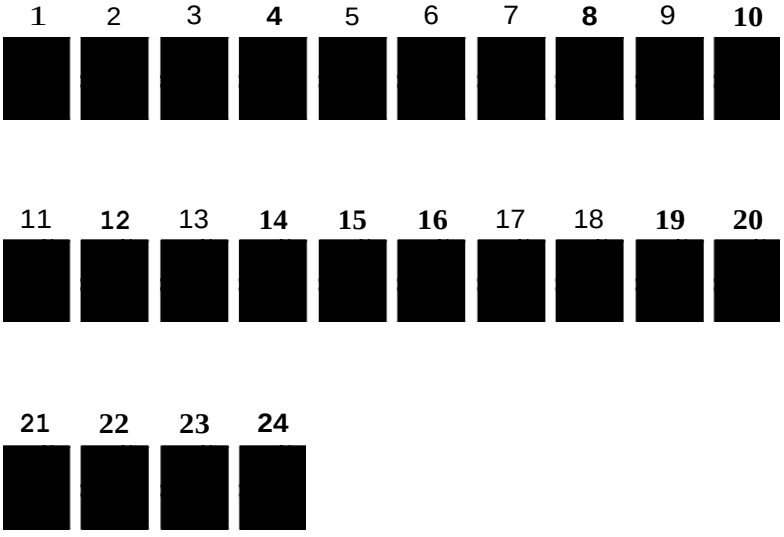
Lalu, perintah berikut digunakan untuk menjadikan obj ek berformat **RGB:**

```
objek = objek[..., : : -1]
```

Citra pada objek ini dtampilkan melalui:

```
plt.subplot(3, 10, indeks)
plt.imshow(objek)
```

Gambar 11.6 menunjukkan hasil skrip koin2 .py.

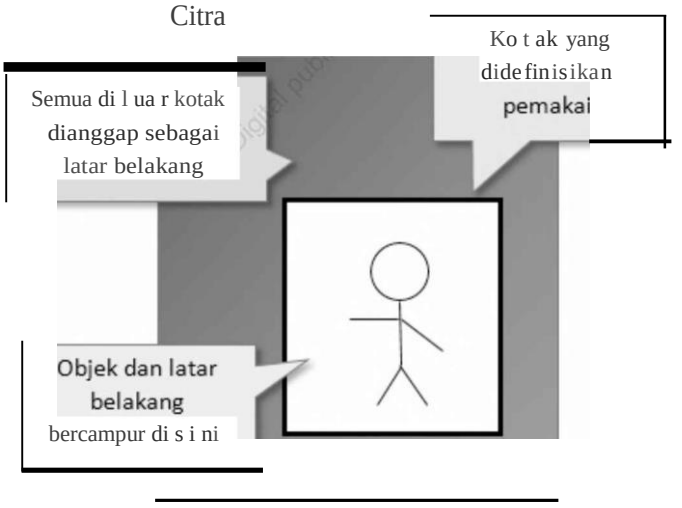


Gambar 11.6 Deretan objek koin yang didapat

11.3 Segmentasi dengan GrabCut

Grabcut didesain oleh Carsten Rother, Vladimir Kolmogorov & Andrew Blake, yang bekerja pada Microsoft Research Cambridge, Inggris. Implementasinya memerlukan waktu yang cukup lama untuk melakukan segmentasi; lebih lama daripada *watershed.* Kelebihannya, hasilnya lebih akurat. Karena memakan waktu lama, metode ini hanya cocok untuk gambar diam (bukan video).

Metode ini bertumpu pada kotak yang didefinisikan pemakai, yang mengandung objek. Semua yang berada di kotak dianggap sebagai latar belakang, sedangkan yang berada di dalam kotak belum diketahui. Kemudian, informasi di luar kotak akan digunakan sebagai acuan untuk menghapus apapun di dalam kotak yang menyerupai bagian di luar kotak.



Gambar 11. 7 Kotak yang didelinisikan pemakai mengandung objek

GrabCut bekerja atas dasar algoritma "Graph Cut" dan menggunakan Gaussian Mixture Model (GMM) untuk memperkirakan objek target

melalui distribusi warna. Dalam hal ini, setiap piksel akan dikelompokkan berdasarkan distribusi warnanya. Distribusi warna setiap piksel disusun dalam satu graf. Berdasarkan graf inilah, penentuan latar belakang dan latar depan dilaksanakan. lnformasi lebih lanjut mengenai metode ini dapat diakses melalui:

https://www.microsoft.com/en-us/research/wp-content/uploads/2004/08/siggraph04-grabcut.pdf

OpenCV menyediakan `cv2.grabCut()` untuk melaksanakan operasi GrabCut. Bentuknya seperti berikut:

cv2.grabCut(citra, *cadar, kotak, mode/LB, mode/LO, jumlterasi, mode)*

Argumen pertama berupa citra dalam format RGB yang hendak diproses. Argumen kedua berupa larik yang kelak akan diisi dengan cadar untuk segmentasi objek. Argumen ketiga menentukan area kotak untuk pemrosesan segmentasi objek. Argumen keempat berupa larik yang digunakan untuk pemrosesan latar belakang. Argumen kelima berupa larik yang digunakan untuk pemrosesan latar depan. Argumen keenam menentukan jumlah iterasi untuk mendapatkan objek. Argumen ketujuh diisi dengan mode pemrosesan. Dalam hal ini, berupa `cv2. GC_ INIT_WITH_RECT` untuk pemrosesan berbentuk kotak.

Sebagai contoh, dikehendaki untuk mendapatkan satu mobil yang berada pada Gambar 11.8. Maka, pertama perlu didefinisikan kotak yang mengandung mobil tersebut seperti terlihat pada Gambar 11.9.

Gambar 11.8 Citra mobil.png

Gambar 11.9 Kotak menentukan objek mobil yang hendak didapatkan

mewujudkannya ditunjukkan pada skrip berikut:

Berkas : ke:r:2.py

```
# Segmentasi dengan Grabcut

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

```
citra = cv2.imread("mobil.png")
if citra is None:
    print("Berkas citra tidak ditemukan")
    exit()

plt.subplot(231)
plt.imshow(citra[..., : : -1])
plt.xticks([]), plt.yticks([])
plt.title('Citra Asal')

# Koversi ke RGB
rgb = cv2.cvtColor(citra, cv2.COLOR_BGR2RGB)

# Kotak untuk objek yang akan diambil
kotak = (6, 170, 190, 85)

# citra dengan kotak
citraKotak  = citra.copy()
cv2.rectangle(citraKotak, (kotak[0], kotak[1]),
              (kotak[0] + kotak[2] , kotak[1] + kotak[3]),
              (255, 255, 255), 2)

plt.subplot(232)
plt.imshow(citraKotak[...,      -1])
plt.xticks([]), plt.yticks([])
plt.title('Citra berkotak')

# Buat cadar awal
cadar = np.zeros(rg b.shape[:2], np.uint8)

# Buat larik untuk grabCut
modelLB   np.zeros((1, 65), np.float64)
modelLD = np.zeros((1, 65), np.float64)

# Eksekusi grabCut
cv2.grabCut(rgb, cadar, kotak,
            modelLB, modelLD, 5,
            cv2.GC INIT_WITH RECT)

plt.subplot(233)
plt.imshow(cadar, cmap = "gray")
plt.xticks([]), plt.yticks([])
plt.title('Cadar hasil')

# Buat cadar dengan latar belakang dibuat menjadi 0
#    dan yang lain bernilai 1
cadarAkhir = np.where((cadar ==   0)
             (cadar == 2), 0, 1).astype('uint8')
```

```
plt.subplot(234)
plt.imshow(cadarAkhir, cmap = "gray")
plt.xticks([]), plt.yticks([])
plt.title('Cadar akhir')

# Proses untuk mendapatkan citra tanpa latar belakang
citraAkhir = rgb * cadarAkhir[:, :, np.newaxis]

plt.subplot(235)
plt.imshow(citraAkhir)
plt.xticks([]), plt.yticks([])
plt.title('Citra basil')

show()
```

Akhir berkas

Pernyataan

```
rgb = cv2.cvtColor(citra, cv2.COLOR_BGR2RGB)
```

digunakan untuk mengonversi citra BGR ke GBR. Perintah ini merupakan alternatif dari:

```
rgb = citra[...,       -1])
```

Pernyataan

```
kotak = (6, 170, 190, 85)
```

digunakan untuk membentuk tupel yang menyatakan kotak. Nilai pertama dan kedua menyatakan koordinat (X, Y) yang merupakan pojok kiri-atas kotak. Elemen ketiga menyatakan lebar kotak dan elemen keempat menyatakan tinggi kotak.

Pernyataan berikut digunakan untuk menciptakan larik berdimensi dua yang ukurannya sama dengan ukuran jumlah baris dan kolom pada larik rgb **dan semua elemennya bernilai O dengan tipe bilangan bulat 8-bit tak bertanda:**

```
cadar = np.zeros(rgb.shape[:2], np.uint8)
```

Pernyataan

```
citraKotak = citra.copy()
```

digunakan untuk menyalin citra ke ci traKotak. **Citra ini dimaksudkan untuk dilengkapi dengan gambar kotak untuk keperluan visualisasi saja. Perintah pembuatan kotak berupa:**

```
cv2.rectangle(citraKotak, (kotak[0],  kotak[l]),
              (kotak[O] + kotak[2], kotak[l] + kotak[3]),
              (255, 255, 255), 2)
```

Cadar yang digunakan oleh grabCut () **disediakan dengan semua elemen bernilai nol. Pernyataannya berupa:**

```
cadar = np.zeros(rgb.shape[:2], np.uint8)
```

Setelah grabCut () **dipanggil,** cadar **akan berubah menjadi cadar yang dapat digunakan untuk memperoleh objek.**

Larik yang digunakan untuk pemrosesan latar belakang dan latar depan oleh grabCut () **disediakan dengan menggunakan dua pernyataan berikut:**

```
modelLB = np.zeros((l, 65), np.float64)
modelLD = np.zeros((l, 65), np.float64)
```

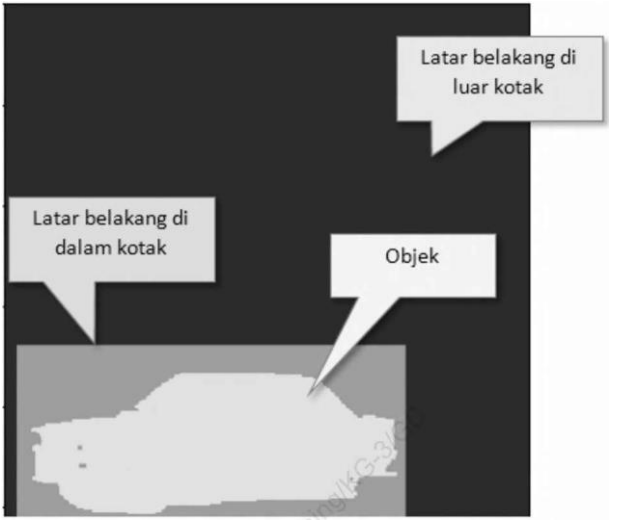
Masing-masing berupa larik berukuran 1 x 65 dengan tipe elemen berupa bilangan real 64-bit. Dalam hal ini, modelLB **untuk menangani latar belakang dan** modelLD **untuk menangani latar depan. Semua elemen diberi nilai nol.**

Pemanggilan grabCut () **dilaksanakan melalui:**

```
cv2.grabCut(rgb, cadar, kotak,
            modelLB, modelLD, 5,
            cv2.GC_INIT_WITH_RECT)
```

Hasil setelah pemanggilan grabCut (), cadar **mengandung tiga bagian, yaitu latar belakang di luar kotak, latar belakang di dalam kotak,**

dan objek di dalam kotak. Gambar 11.10 menunjukkan contoh mengenai hal ini. Nilai untuk masing-masing secara berturutan berupa 0, 2, dan 1.



Gambar 11.10 Cadar hasil pemrosesan dengan grabCutO

Oleh karena itu, agar hanya bagian objek bernilai 1 dan latar belakang bernilai O perlu diberikan perintah seperti berikut:

```
cadarAkhir =np.where((cadar ==0) |
              (cadar ==2), 0, 1).astype('uint8')
```

Perintah ini sekaligus membuat cadarAkhir bertipe bilangan bulat 8-bit tak bertanda sehingga sama dengan tipe elemen citra.

Cadar cadarAkhir inilah yang digunakan untuk menghilangkan latar belakang pada citra asal. Proses penghilangan latar belakang dilakukan dengan menggunakan:

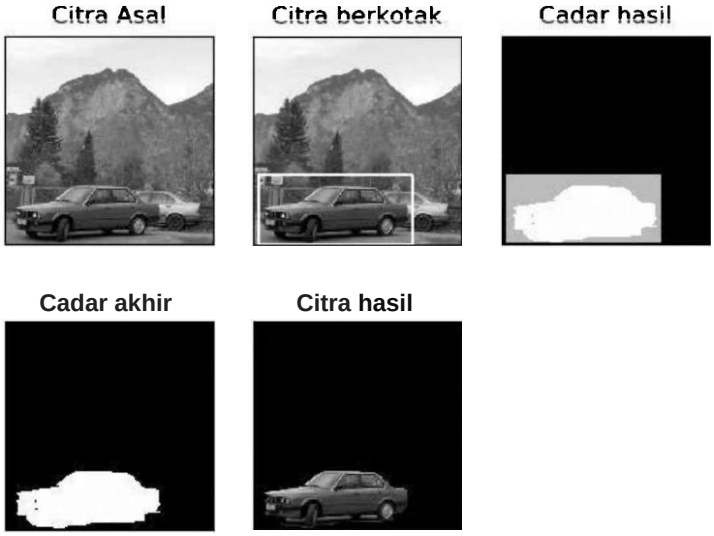
```
citraAkhir =rgb *cadarAkhir[:, ., np.newaxis]
```

Perintah ini identik dengan:

```
citraAkhir =cv2.bitwise and(rgb, rgb,
```

```
        mask= cadarAkhir)
```

Gambar 11.11 menunjukkan berbagai keadaan sebelum dan sesudah segmentasi terhadap objek dilaksanakan.



Gambar 11.11 Proses untuk melakukan segmentasi dengan grabCutO