

**Menampilkan koordinat titik yang diklik pada gambar menggunakan Python-OpenCV**

OpenCV membantu kita mengontrol dan mengelola berbagai jenis peristiwa mouse dan memberikan fleksibilitas untuk mengoperasikannya. Ada banyak jenis peristiwa mouse. Peristiwa ini dapat ditampilkan dengan menjalankan segmen kode berikut:

```
import cv2
[print(i) for i in dir(cv2) if 'EVENT' in i]
```

**Output:**

EVENT\_FLAG\_ALTKEY  
EVENT\_FLAG\_CTRLKEY  
EVENT\_FLAG\_LBUTTON  
EVENT\_FLAG\_MBUTTON  
EVENT\_FLAG\_RBUTTON  
EVENT\_FLAG\_SHIFTKEY  
EVENT\_LBUTTONDBLCLK  
EVENT\_LBUTTONDOWN  
EVENT\_LBUTTONUP  
EVENT\_MBUTTONDBLCLK  
EVENT\_MBUTTONDOWN  
EVENT\_MBUTTONUP  
EVENT\_MOUSEHWHEEL  
EVENT\_MOUSEMOVE  
EVENT\_MOUSEWHEEL  
EVENT\_RBUTTONDBLCLK  
EVENT\_RBUTTONDOWN  
EVENT\_RBUTTONUP

Sekarang mari kita lihat bagaimana cara menampilkan koordinat titik yang diklik pada gambar. Kami akan menampilkan kedua titik yang diklik dengan mengklik kanan dan mengklik kiri.

**Algoritma:**

1. Impor modul cv2.
2. Impor gambar menggunakan fungsi cv2.imread().
3. Tampilkan gambar menggunakan fungsi cv2.imshow().
4. Panggil fungsi cv2.setMouseCallback() dan berikan jendela gambar serta fungsi

yang ditentukan pengguna sebagai parameter.

5. Dalam fungsi yang ditentukan pengguna, periksa klik mouse kiri menggunakan atribut `cv2.EVENT_LBUTTONDOWN`.
6. Tampilkan koordinat di Shell.
7. Tampilkan koordinat di jendela yang dibuat.
8. Lakukan hal yang sama untuk klik mouse kanan menggunakan atribut `cv2.EVENT_RBUTTONDOWN`. Ubah warna saat menampilkan koordinat pada gambar untuk membedakan dari klik kiri.
9. Di luar fungsi yang ditentukan pengguna, gunakan fungsi `cv2.waitKey(0)` dan `cv2.destroyAllWindows()` untuk menutup jendela dan mengakhiri program.

Kami akan menggunakan versi berwarna dari gambar Lena.



```
# mengimpor modul
import cv2

# fungsi untuk menampilkan koordinat
# dari titik yang diklik pada gambar
def click_event(event, x, y, flags, params):

    # memeriksa klik mouse kiri
    if event == cv2.EVENT_LBUTTONDOWN:

        # menampilkan koordinat
        # di Shell
        print(x, ' ', y)

        # menampilkan koordinat
        # pada jendela gambar
        font = cv2.FONT_HERSHEY_SIMPLEX
```

```
cv2.putText(img, str(x) + ',' +
            str(y), (x, y), font,
            1, (255, 0, 0), 2)
cv2.imshow('image', img)

# memeriksa klik mouse kanan
if event == cv2.EVENT_RBUTTONDOWN:

    # menampilkan koordinat
    # di Shell
    print(x, ' ', y)

    # menampilkan koordinat
    # pada jendela gambar
    font = cv2.FONT_HERSHEY_SIMPLEX
    b = img[y, x, 0]
    g = img[y, x, 1]
    r = img[y, x, 2]
    cv2.putText(img, str(b) + ',' +
                str(g) + ',' + str(r),
                (x, y), font, 1,
                (255, 255, 0), 2)
    cv2.imshow('image', img)

# fungsi utama
if __name__ == "__main__":

    # membaca gambar
    img = cv2.imread('lena.jpg', 1)

    # menampilkan gambar
    cv2.imshow('image', img)

    # mengatur pengelola mouse untuk gambar
    # dan memanggil fungsi click_event()
    cv2.setMouseCallback('image', click_event)

    # tunggu hingga tombol ditekan untuk keluar
    cv2.waitKey(0)

    # tutup jendela
    cv2.destroyAllWindows()
```

Output:

[Klik link untuk melihat output](#)

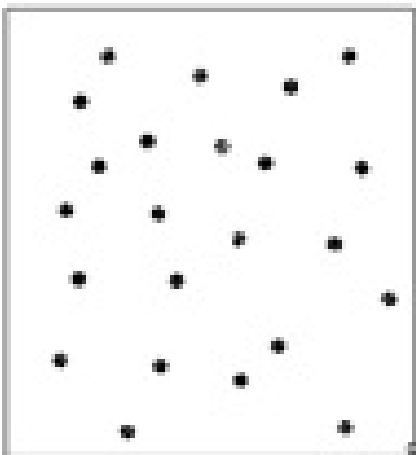
### Deteksi Titik Putih dan Hitam menggunakan OpenCV | Python

Pemrosesan gambar menggunakan Python adalah salah satu topik paling populer dalam dunia saat ini. Namun, pemrosesan gambar sedikit kompleks dan pemula bisa merasa bosan dalam pendekatan pertama mereka. Jadi dalam artikel ini, kami memiliki program pemrosesan gambar Python yang sangat dasar untuk menghitung titik-titik hitam di permukaan putih dan titik-titik putih di permukaan hitam menggunakan fungsi-fungsi OpenCV (cv2.imread, cv2.threshold, cv2.findContours, cv2.contourArea).

#### Menghitung titik-titik hitam pada permukaan putih –

Pertama-tama, kita perlu mengimpor perpustakaan OpenCV. Semua fungsi terkait pemrosesan gambar ada dalam perpustakaan ini. Untuk menyimpan path gambar yang akan kita proses dalam variabel, kita akan menggunakan kode berikut:

```
import cv2
# path = "C:/Users/Personal/Downloads/black dot.jpg"
path = "black dot.jpg"
```



Memuat gambar dalam mode grayscale. Dalam mode grayscale, gambar dikonversi menjadi gambar hitam putih yang terdiri dari bayangan abu-abu.

```
gray = cv2.imread(path, 0)
```

Fungsi cv2.threshold bekerja sebagai berikut: jika nilai piksel lebih besar dari nilai ambang, maka diberikan satu nilai (mungkin putih), jika tidak, maka diberikan nilai lain (mungkin hitam). Argumen pertama adalah gambar sumber, yang seharusnya merupakan gambar grayscale (telah dilakukan sebelumnya). Argumen kedua adalah nilai ambang yang digunakan untuk mengklasifikasikan nilai piksel. Untuk nilai ambang, cukup lewatkan nol. Kemudian algoritma akan menemukan nilai ambang yang optimal dan mengembalikannya sebagai output kedua, yaitu th. Jika pengambangan Otsu tidak digunakan, th sama dengan

nilai ambang yang Anda gunakan.

```
# ambang batas
th, threshed = cv2.threshold(gray, 100, 255,
                             cv2.THRESH_BINARY_INV|cv2.THRESH_OTSU)
```

Kontur dapat dijelaskan secara sederhana sebagai kurva yang menghubungkan semua titik-titik berkelanjutan (sepanjang batas), memiliki warna atau intensitas yang sama. Kontur adalah alat yang berguna untuk analisis bentuk dan deteksi serta pengenalan objek. Kontur memberikan akurasi yang lebih baik untuk menggunakan gambar biner. Ada tiga argumen dalam fungsi `cv2.findContours()`, yang pertama adalah gambar sumber, yang kedua adalah mode pengambilan kontur, dan yang ketiga adalah metode pendekatan kontur. Ini menghasilkan kontur dan hierarki. Kontur adalah daftar Python dari semua kontur dalam gambar. Setiap kontur individu adalah larik Numpy dari koordinat (x, y) dari titik-titik batas objek.

Ini terutama menghubungkan titik-titik hitam pada gambar untuk menghitung –

```
# temukan kontur
cnts = cv2.findContours(threshed, cv2.RETR_LIST,
                       cv2.CHAIN_APPROX_SIMPLE)[-2]
```

`cv2.contourArea()` dapat menghitung luas kontur objek. Di sini objeknya adalah titik-titik hitam. Ketika menemukan titik hitam, itu akan menghitung luasnya, dan jika memenuhi syarat area minimum untuk dihitung sebagai titik, maka akan memasukkan nilai luasnya ke dalam daftar `xcnts`.

```
# saring berdasarkan area
s1 = 3
s2 = 20
xcnts = []
for cnt in cnts:
    if s1<cv2.contourArea(cnt) <s2:
        xcnts.append(cnt)
```

Akhirnya, kita tidak memerlukan luasan. Jika dianggap sebagai titik, maka luasnya akan dimasukkan ke dalam daftar `xcnts`. Jadi kita akan mendapatkan jumlah titik jika kita menghitung panjang daftar tersebut.

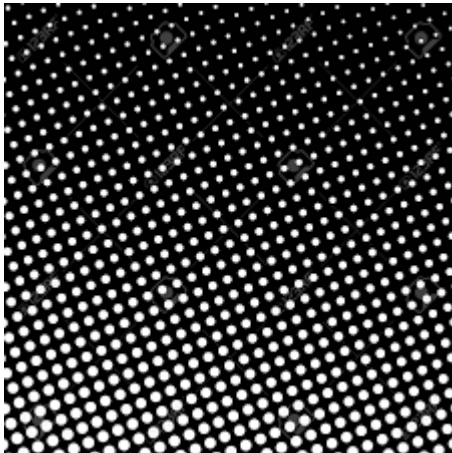
```
print("\nJumlah Titik: {}".format(len(xcnts)))
```

Output:

23

**Menghitung titik putih pada latar belakang hitam –**

Sekarang untuk menghitung titik putih, kita perlu mengubah ambang sedikit. kita harus menggunakan `cv2.THRESH_BINARY` alih-alih `cv2.THRESH_BINARY_INV` karena kita menghitung nilai putih pada permukaan hitam. Proses lainnya sama. Kita dapat mengubah nilai `s1` dan `s2` untuk memeriksa hasil terbaik.



```
import cv2
path = "white dot.png"

# membaca gambar dalam mode grayscale
gray = cv2.imread(path, 0)

# ambang batas
th, threshed = cv2.threshold(gray, 100, 255,
                             cv2.THRESH_BINARY|cv2.THRESH_OTSU)

# temukan kontur
cnts = cv2.findContours(threshed, cv2.RETR_LIST,
                       cv2.CHAIN_APPROX_SIMPLE)[-2]

# saring berdasarkan area
s1 = 3
s2 = 20
xcnts = []

for cnt in cnts:
    if s1 < cv2.contourArea(cnt) < s2:
        xcnts.append(cnt)

# mencetak output
print("\nJumlah Titik: {}".format(len(xcnts)))
```

Output:  
583



SISTEM INFORMASI

## Deteksi dan Ekstraksi Teks menggunakan OpenCV dan OCR

OpenCV (Open source computer vision) adalah perpustakaan dari fungsi-fungsi pemrograman yang ditujukan terutama untuk visi komputer waktu nyata. OpenCV dalam Python membantu untuk memproses gambar dan menerapkan berbagai fungsi seperti mengubah ukuran gambar, manipulasi piksel, deteksi objek, dan lainnya. Dalam artikel ini, kita akan belajar cara menggunakan kontur untuk mendeteksi teks dalam sebuah gambar dan menyimpannya ke dalam file teks.

### Instalasi yang Diperlukan:

```
pip install opencv-python  
pip install pytesseract
```

Paket OpenCV digunakan untuk membaca gambar dan menjalankan beberapa teknik pemrosesan gambar tertentu. Python-tesseract adalah wrapper untuk Mesin Tesseract-OCR milik Google yang digunakan untuk mengenali teks dari gambar.

Unduh file eksekusi Tesseract dari [tautan ini](#).

### Langkah-Langkah:

Setelah impor yang diperlukan, sebuah gambar sampel dibaca menggunakan fungsi ``imread`` dari OpenCV.

Penerapan pemrosesan gambar pada gambar:

Warna gambar pertama-tama diubah dan disimpan dalam sebuah variabel. Untuk konversi warna, kita menggunakan fungsi ``cv2.cvtColor(gambar_masukan, flag)``. Parameter kedua, ``flag``, menentukan jenis konversi. Kita dapat memilih antara ``cv2.COLOR_BGR2GRAY`` dan ``cv2.COLOR_BGR2HSV``. ``cv2.COLOR_BGR2GRAY`` membantu mengubah gambar RGB menjadi gambar skala abu-abu, dan ``cv2.COLOR_BGR2HSV`` digunakan untuk mengubah gambar RGB menjadi gambar ruang warna HSV (Hue, Saturation, Value). Di sini, kita menggunakan ``cv2.COLOR_BGR2GRAY``. Ambang diterapkan pada gambar yang telah dikonversi menggunakan fungsi ``cv2.threshold``.

Ada 3 jenis thresholding:

- Thresholding Sederhana
- Thresholding Adaptif
- Binerisasi Otsu

Untuk informasi lebih lanjut tentang thresholding, lihat Teknik Teknik Thresholding menggunakan OpenCV. ``cv2.threshold()`` memiliki 4 parameter, di mana parameter pertama adalah gambar yang telah mengalami perubahan warna, diikuti oleh nilai ambang minimum, nilai ambang maksimum, dan jenis thresholding yang perlu diterapkan.



**Untuk mendapatkan struktur berbentuk persegi:**

``cv2.getStructuringElement()`` digunakan untuk mendefinisikan elemen struktural seperti elips, lingkaran, persegi panjang, dll. Di sini, kita menggunakan elemen struktural berbentuk persegi panjang (``cv2.MORPH_RECT``). ``cv2.getStructuringElement`` mengambil parameter ukuran kernel tambahan. Kernel yang lebih besar akan mengelompokkan blok teks yang lebih besar bersama-sama. Setelah memilih kernel yang tepat, dilasi diterapkan pada gambar dengan menggunakan fungsi ``cv2.dilate``. Dilasi membuat kelompok teks yang akan dideteksi menjadi lebih akurat karena memperluas blok teks.

**Mencari Kontur:**

``cv2.findContours()`` digunakan untuk menemukan kontur dalam gambar yang telah mengalami dilasi. Ada tiga argumen dalam ``cv.findContours()``: gambar sumber, mode pengambilan kontur, dan metode pendekatan kontur.

Fungsi ini mengembalikan kontur dan hirarki. Kontur adalah daftar Python dari semua kontur dalam gambar. Setiap kontur individu adalah larik Numpy dari koordinat (x, y) dari titik-titik batas dalam objek. Kontur biasanya digunakan untuk menemukan objek berwarna putih dari latar belakang hitam. Semua teknik pemrosesan gambar di atas diterapkan sehingga Kontur dapat mendeteksi tepi batas blok teks dalam gambar. Sebuah file teks dibuka dalam mode penulisan dan dikosongkan. File teks ini dibuka untuk menyimpan teks dari hasil OCR.

**Penerapan OCR:**

Mengulangi setiap kontur dan mengambil koordinat x dan y serta lebar dan tinggi menggunakan fungsi ``cv2.boundingRect()``. Kemudian menggambar persegi panjang pada gambar menggunakan fungsi ``cv2.rectangle()`` dengan bantuan koordinat x dan y yang diperoleh dan lebar dan tinggi. Ada 5 parameter dalam ``cv2.rectangle()``, di mana parameter pertama menentukan gambar masukan, diikuti oleh koordinat x dan y (koordinat awal dari persegi panjang), koordinat akhir dari persegi panjang yang adalah (x+w, y+h), warna batas untuk persegi panjang dalam nilai RGB, dan ukuran batas. Sekarang, potong wilayah berbentuk persegi panjang dan kemudian lewatkan ke tesseract untuk mengekstrak teks dari gambar. Kemudian kita membuka file teks yang telah dibuat dalam mode penambahan untuk menambahkan teks yang diperoleh dan menutup file.

Gambar sampel yang digunakan untuk kode:

[insert\_sample\_image]

This is SAMPLE TEXT

Text is at different regions

```
# Impor paket yang diperlukan
import cv2
import pytesseract

# Sebutkan lokasi yang diinstal Tesseract-OCR di sistem Anda
pytesseract.pytesseract.tesseract_cmd = 'System_path_to_tesseract.exe'

# Baca gambar dari mana teks perlu diekstrak
img = cv2.imread("sample.jpg")

# Pemrosesan awal gambar dimulai

# Ubah gambar menjadi skala abu-abu
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Melakukan ambang OTSU
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU |
cv2.THRESH_BINARY_INV)

# Tentukan bentuk struktur dan ukuran kernel.
# Ukuran kernel meningkatkan atau mengurangi area
# persegi panjang yang akan terdeteksi.
# Nilai kernel yang lebih kecil seperti (10, 10) akan mendeteksi
# setiap kata daripada kalimat.
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (18, 18))

# Menerapkan dilasi pada gambar ambang
dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)

# Menemukan kontur
```

```
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL,
                                       cv2.CHAIN_APPROX_NONE)

# Membuat salinan gambar
im2 = img.copy()

# Sebuah file teks dibuat dan dikosongkan
file = open("recognized.txt", "w+")
file.write("")
file.close()

# Melalui kontur yang teridentifikasi
# Kemudian bagian berbentuk persegi panjang dipotong dan diteruskan
# ke pytesseract untuk mengekstrak teks darinya
# Teks yang diekstrak kemudian ditulis ke dalam file teks
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)

    # Menggambar persegi panjang pada gambar yang disalin
    rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Memotong blok teks untuk memberikan masukan ke OCR
    cropped = im2[y:y + h, x:x + w]

    # Buka file dalam mode penambahan
    file = open("recognized.txt", "a")

    # Terapkan OCR pada gambar yang dipotong
    text = pytesseract.image_to_string(cropped)

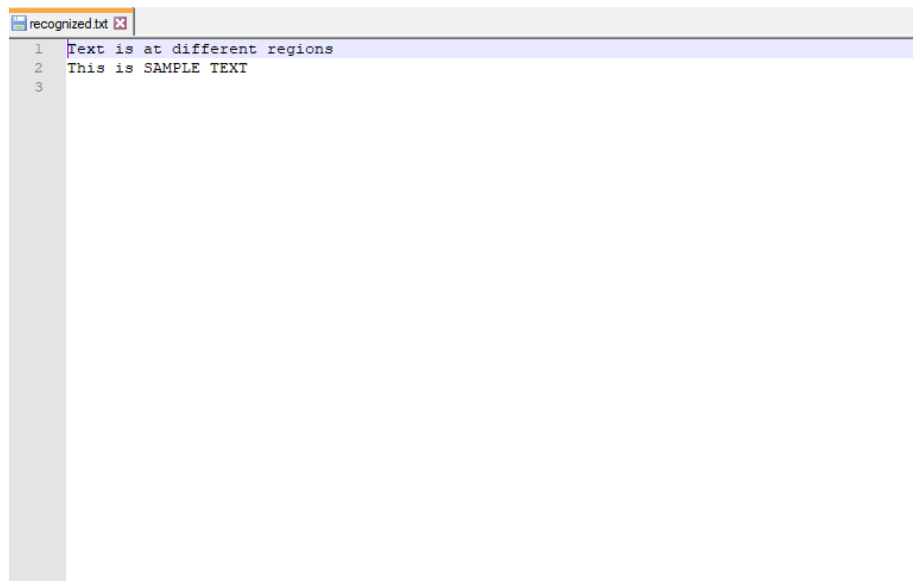
    # Menambahkan teks ke dalam file
    file.write(text)
    file.write("\n")

# Tutup file
file.close
```

Keluaran:

File teks akhir:

[insert\_final\_text\_file]



```
recognized.txt
1 Text is at different regions
2 This is SAMPLE TEXT
3
```

Blok teks yang terdeteksi:

[insert\_detected\_text\_blocks]

This is SAMPLE TEXT

Text is at different regions