JURNAL TEKOMIN
Teknologi Komputer dan Informatika
E-ISSN :
P-ISSN :

DETEKSI WAJAH DENGAN METODE HAAR CASCADE MENGGUNAKAN OPENCV *(FACE DETECTION WITH HAAR CASCADE METHOD USING OPENCV)*

Dwi Robiul Rochmawati
Politeknik Pajajaran (Teknologi Komputer, Politeknik Pajajaran, Bandung, Indonesia)
dwi.robiul@poljan.ac.id

Abstract

Face detection is one of the most researched topics in the field of computer vision and digital image processing. The Haar Cascade method is one method that is often used to detect faces in an image or video. This method uses Haar features to identify faces by comparing them with existing faces in the database. This research aims to implement the Haar Cascade method to detect faces using the OpenCV library. The stages include image acquisition, image processing, face detection using Haar Cascade, and evaluation of detection results. The results show that the Haar Cascade method can detect faces quite well in various lighting conditions and viewing angles.

Keywords: *Face Detection; Haar Cascade; OpenCV*

Abstrak

Deteksi wajah merupakan salah satu topik yang banyak diteliti dalam bidang visi komputer dan pengolahan citra digital. Metode Haar Cascade adalah salah satu metode yang sering digunakan untuk mendeteksi wajah pada suatu citra atau video. Metode ini menggunakan fitur Haar untuk mengidentifikasi wajah dengan cara membandingkannya dengan wajah yang ada pada database. Penelitian ini bertujuan untuk mengimplementasikan metode Haar Cascade untuk mendeteksi wajah menggunakan library OpenCV. Tahapan yang dilakukan meliputi akuisisi citra, pemrosesan citra, deteksi wajah menggunakan Haar Cascade, dan evaluasi hasil deteksi. Hasil penelitian menunjukkan bahwa metode Haar Cascade dapat mendeteksi wajah dengan cukup baik dalam berbagai kondisi pencahayaan dan sudut pandang.

Kata kunci: Deteksi Wajah; Haar Cascade; OpenCV

PENDAHULUAN

Pendeteksian wajah merupakan salah satu topik yang telah lama menjadi perhatian dalam bidang pengolahan citra digital. Salah satu metode yang sering digunakan adalah Haar Cascade, yang memanfaatkan algoritma pembelajaran mesin untuk mendeteksi objek dalam suatu citra (Setiawan et al., 2021). Metode Haar Cascade diperkenalkan oleh Paul Viola dan Michael Jones pada tahun 2001, dan sejak saat itu telah banyak digunakan dalam berbagai aplikasi, seperti pengenalan wajah, deteksi mobil, dan sebagainya.

Dalam penelitian ini, kami akan membahas penerapan metode Haar Cascade untuk deteksi wajah menggunakan OpenCV, sebuah library open-source untuk pengolahan citra digital. OpenCV menyediakan implementasi siap pakai dari algoritma Haar Cascade, sehingga memudahkan penggunaannya dalam berbagai aplikasi. (Kosasih & Daomara, 2021) (Rifki et al., 2021) (Putra et al., 2013) (Setiawan et al., 2021) Proses penerapan metode Haar Cascade untuk deteksi wajah terdiri dari beberapa tahap, yaitu preprocessing citra, ekstraksi fitur, dan klasifikasi.

Pada tahap preprocessing, citra input akan dinormalisasi ukuran dan tingkat keabuannya untuk memudahkan proses ekstraksi fitur (Putra et al., 2013). Ekstraksi fitur dilakukan dengan mengidentifikasi ciri-ciri wajah yang khas, seperti mata, hidung, dan mulut, menggunakan filter Haar-like (Kosasih & Daomara, 2021). Selanjutnya, proses klasifikasi dilakukan dengan mencocokkan fitur-fitur yang diekstrak dengan model wajah yang telah dilatih sebelumnya.

Penerapan metode Haar Cascade untuk deteksi wajah telah banyak diteliti sebelumnya. Penggabungan metode GLCM dan PNN terbukti efektif untuk mengenali citra wajah. (Kosasih & Daomara, 2021) Selain itu, pemanfaatan metode LBPH juga mampu menghasilkan akurasi pengenalan yang baik. (Kosasih & Daomara, 2021) Di sisi lain, penggunaan Convolutional Neural Network juga telah diimplementasikan untuk pengenalan wajah, dengan hasil yang cukup memuaskan. (Rifki et al., 2021) (Setiawan et al., 2021)

Melalui penelitian ini, kami berharap dapat memberikan pemahaman yang lebih mendalam mengenai penerapan metode Haar Cascade untuk deteksi wajah menggunakan OpenCV, serta mengetahui berbagai macam pendekatan yang telah dilakukan dalam bidang pengenalan wajah..

KAJIAN PUSTAKA

Kajian pustaka dalam penelitian ini mencakup beberapa aspek penting yang mendukung penggunaan metode Haar Cascade untuk deteksi wajah dengan memanfaatkan OpenCV.

Deteksi Wajah

Deteksi wajah merupakan salah satu bidang dalam visi komputer yang memiliki tujuan untuk mengidentifikasi dan menemukan lokasi wajah manusia dalam suatu citra digital atau video. (Rifki et al., 2021) Proses deteksi wajah ini dapat dilakukan dengan menggunakan berbagai macam metode, salah satunya adalah metode Haar Cascade.

Metode Haar Cascade

Metode Haar Cascade merupakan sebuah algoritma pendeteksi objek berbasis machine learning yang awalnya dikembangkan oleh Paul Viola dan Michael Jones (Putra et al., 2013) (Kosasih & Daomara, 2021). Algoritma ini bekerja dengan menggunakan fitur-fitur Haar sebagai masukan untuk

mengklasifikasikan region pada citra yang mengandung objek tertentu, dalam hal ini adalah wajah. Proses deteksi wajah dengan Haar Cascade terdiri dari beberapa tahapan, yaitu ekstraksi fitur, pembentukan classifier, dan pendeteksian wajah. Fitur Haar adalah karakteristik lokal pada citra yang dapat direpresentasikan sebagai perbedaan intensitas piksel antara region yang berdekatan. Classifier yang dihasilkan dari proses pelatihan digunakan untuk mendeteksi wajah pada citra masukan.

Penerapan OpenCV

OpenCV merupakan library open source yang berfokus pada visi komputer dan machine learning.
Dikembangkan oleh Intel pada tahun 2000, OpenCV bertujuan untuk menyediakan infrastruktur yang kuat untuk berbagai aplikasi seperti pengolahan citra, pelacakan objek, pengenalan wajah, dan analisis video. Pustaka ini ditulis dalam bahasa C++ dengan antarmuka yang tersedia untuk berbagai bahasa pemrograman seperti Python, Java, dan MATLAB.
Fungsi Utama OpenCV

- Pengolahan Citra
  OpenCV memiliki kemampuan untuk melakukan berbagai operasi pengolahan citra seperti konversi warna, deteksi tepi, segmentasi, dan transformasi geometris. Contohnya adalah menggunakan metode seperti Canny Edge Detection untuk mendeteksi tepi atau Hough Transform untuk deteksi garis.
- Analisis Video
  Dalam pengolahan video, OpenCV mendukung pembacaan, penulisan, dan pemrosesan aliran video secara real-time.
- Pengenalan Wajah dan Objek
  Dengan algoritma seperti Haar Cascade dan HOG (Histogram of Oriented Gradients), OpenCV memungkinkan deteksi dan pengenalan pola seperti wajah, mata, dan objek lainnya.
- Pembelajaran Mesin
  OpenCV mencakup modul pembelajaran mesin dengan berbagai algoritma seperti K-Means Clustering, Support Vector Machines (SVM), dan Neural Networks. Modul ini mempermudah pengembangan aplikasi berbasis kecerdasan buatan.
- Kompatibilitas Multiplatform
  OpenCV mendukung berbagai sistem operasi seperti Windows, Linux, macOS, Android, dan iOS, menjadikannya fleksibel untuk digunakan pada berbagai perangkat.

Dalam implementasi deteksi wajah menggunakan Haar Cascade, OpenCV menyediakan fungsi-fungsi yang dapat digunakan untuk melakukan proses deteksi wajah, seperti `cv2.CascadeClassifier()` dan `cv2.detectMultiScale()`.

METODE PENELITIAN

Metode penelitian ini tergolong ke dalam penelitian terapan dengan pendekatan kuantitatif.

Penelitian terapan digunakan karena bertujuan untuk menghasilkan suatu prototipe atau model yang dapat diimplementasikan dalam dunia nyata (Rifki et al., 2021). Sedangkan pendekatan kuantitatif dipilih karena data yang digunakan berupa angka-angka hasil pengukuran dan pengujian.

Penelitian ini akan dilakukan melalui beberapa tahap, yaitu:

Studi Literatur

Tahap ini dilakukan untuk mempelajari dan memahami konsep-konsep dasar yang terkait

dengan deteksi wajah menggunakan Haar Cascade dan OpenCV.

Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah citra wajah yang diperoleh dari dataset publik atau hasil akuisisi menggunakan kamera.

Prapemrosesan Data

Pada tahap ini, citra wajah yang telah dikumpulkan akan dilakukan beberapa proses, seperti grayscale, normalisasi ukuran, dan perbaikan kualitas citra.

Implementasi Haar Cascade

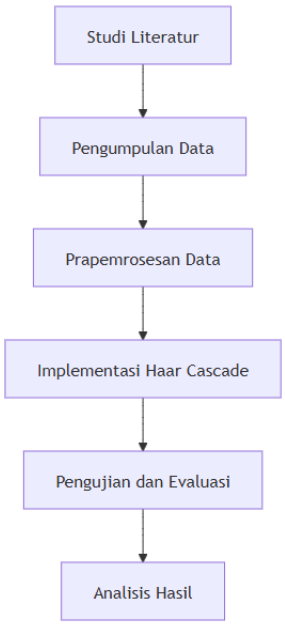
Tahap ini meliputi pelatihan classifier Haar Cascade menggunakan dataset citra wajah dan non-wajah, serta proses deteksi wajah pada citra uji.

Pengujian dan Evaluasi

Tahap ini bertujuan untuk mengevaluasi kinerja sistem deteksi wajah yang telah dibangun.

Analisis Hasil

Pada tahap akhir, akan dilakukan analisis terhadap hasil pengujian yang diperoleh untuk mengetahui keberhasilan dan keterbatasan dari sistem deteksi wajah menggunakan Haar Cascade.



Gambar 1. Tahapan Penelitian

HASIL DAN PEMBAHASAN

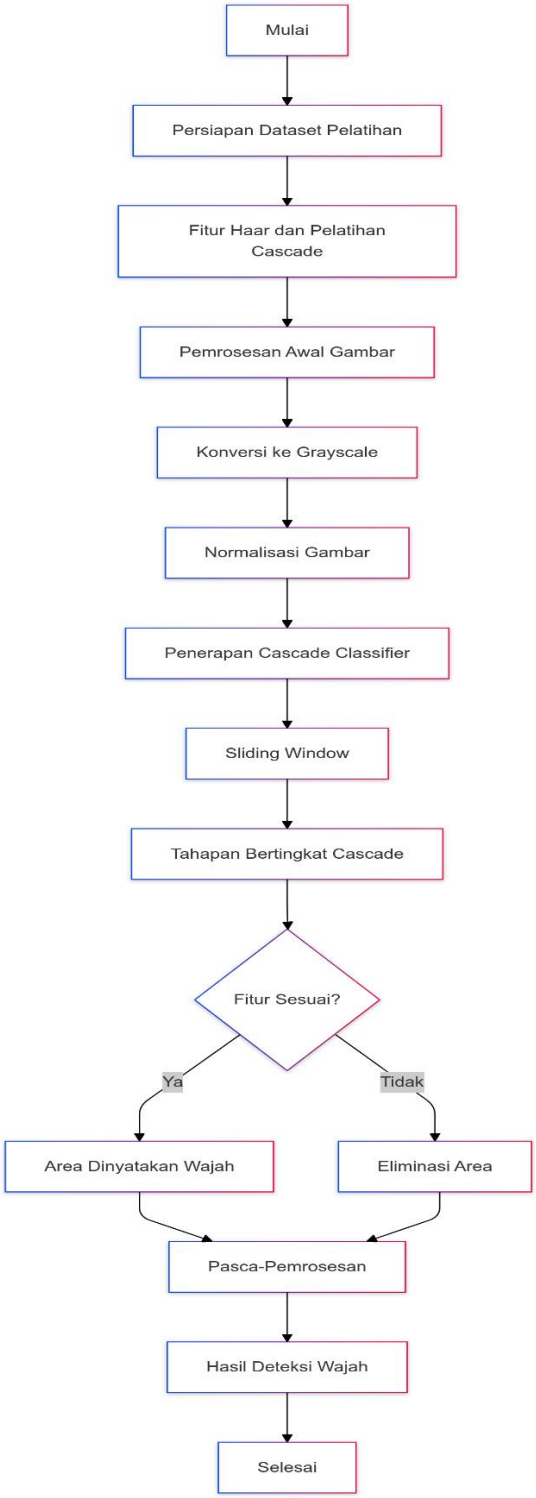
Uraian Proses

1. Persiapan Dataset Pelatihan
   - **Fitur Haar**: Haar Cascade menggunakan fitur Haar, yaitu pola intensitas piksel sederhana seperti area terang dan gelap dalam gambar (misalnya perbedaan warna di sekitar mata, hidung, dan mulut).
   - **Pelatihan dengan Algoritma Cascade**: Untuk mendeteksi wajah, classifier dilatih dengan dataset besar gambar positif (mengandung wajah) dan negatif (tanpa wajah). Algoritma Adaboost digunakan untuk memilih fitur terbaik dari kumpulan besar fitur Haar.
2. Pemrosesan Awal Gambar
   - **Konversi ke Grayscale**: Deteksi wajah lebih efisien dalam gambar

grayscale karena mengurangi kompleksitas data.
- **Normalisasi**: Gambar dapat dinormalisasi untuk memastikan deteksi tidak terganggu oleh pencahayaan.

3. Penerapan Cascade Classifier
   - **Sliding Window**: Algoritma menjalankan window persegi panjang kecil di seluruh area gambar, memeriksa fitur Haar pada setiap posisi dan ukuran window.
   - **Tahapan Bertingkat**: Cascade classifier bekerja secara hierarkis melalui tahapan bertingkat. Tahapan awal digunakan untuk mengeliminasi sebagian besar area non-wajah, sementara tahapan berikutnya memproses area yang kemungkinan besar berisi wajah.
4. Pendeteksian dan Identifikasi Wajah
   - Jika semua tahapan berhasil mendeteksi pola wajah, area tersebut diklasifikasikan sebagai wajah.
   - Algoritma dapat mendeteksi banyak wajah dalam satu gambar dengan memindai area yang berbeda pada skala berbeda.
5. Pasca-Pemrosesan
   - **Menggambar Kotak**: Setelah deteksi berhasil, wajah biasanya ditandai dengan kotak persegi panjang.
   - **Penyaringan Ganda**: Untuk mengurangi kesalahan deteksi, deteksi tambahan seperti pemfilteran ukuran atau rasio wajah dapat dilakukan.

Flowchart Diagram Proses Deteksi Wajah



Gambar 2 Flowchart Diagram Proses Deteksi Wajah

JURNAL TEKOMIN
Teknologi Komputer dan Informatika
E-ISSN :
P-ISSN :

Implementasi dengan Python dan OpenCV

Face detection with Haar cascade (using OpenCV)

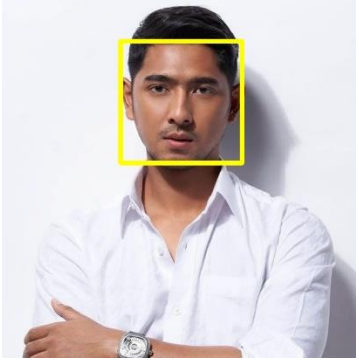
```
[24] image = cv2.imread('/content/person.jpg')

[25] image.shape
     (675, 667, 3)

[26] 675*667*3
     1350675
```

```
cv2_imshow(image)
```

```
[28] image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

[29] image_gray.shape
     (675, 667)

[30] 675*667
     450225

[31] 1350675 - 450225
     900450

     cv2_imshow(image_gray)
```

```
cv2_imshow(image_gray)
```

```
[33] face_detector = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')

[34] detections = face_detector.detectMultiScale(image_gray)

[35] detections
     array([[223,  79, 229, 229]], dtype=int32)

[36] len(detections)
     1

[37] for (x, y, w, h) in detections:
       #print(x, y, w, h)
       cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 7)
     cv2_imshow(image)
```



Mengubah ukuran gambar

Resizing the image

```
image = cv2.imread('/content/people1.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print(image_gray.shape)
#cv2_imshow(image_gray)
detections = face_detector.detectMultiScale(image_gray)
#print(len(detections))
for (x, y, w, h) in detections:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 3)
cv2_imshow(image)
```

JURNAL TEKOMIN
Teknologi Komputer dan Informatika
E-ISSN :
P-ISSN :

Manually

```
image = cv2.imread('/content/people1.jpg')
image_resized = cv2.resize(image, (600, 480))
print(image_resized.shape)
cv2_imshow(image_resized)
```

```
[77] new_width = 600
     proportion = 1680 / 1120
     print(proportion)
     new_height = int(new_width / proportion)
     print(new_height)
```

```
1.5
400
```

```
image = cv2.imread('/content/people1.jpg')
image_resized = cv2.resize(image, (new_width, new_height))
print(image_resized.shape)
cv2_imshow(image_resized)
```

```
(400, 600, 3)
```

Scale

```
[84] image = cv2.imread('/content/people1.jpg')
     image = cv2.resize(image, (0,0), fx = 0.5, fy = 0.5)
     image.shape
```

```
(512, 384, 3)
```

```
[85] cv2_imshow(image)
```

```
[86] image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     detections = face_detector.detectMultiScale(image_gray)
     print(len(detections))
     for (x, y, w, h) in detections:
       cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 3)
     cv2_imshow(image)
```

```
3
```

Haarscade Parameter

Haarscade parameter adalah serangkaian pengaturan yang digunakan untuk menyesuaikan perilaku deteksi objek menggunakan metode Haar Cascade di OpenCV. Haar Cascade adalah algoritma deteksi objek berbasis pembelajaran mesin yang sering digunakan untuk mendeteksi wajah, mata, dan objek lain di gambar atau video.

Berikut adalah penjelasan beberapa parameter penting dalam Haar Cascade:

1. `scaleFactor`

ScaleFactor adalah parameter yang digunakan untuk menentukan seberapa banyak ukuran gambar dikurangi pada setiap skala saat memproses gambar. Parameter ini berkaitan dengan image pyramid, yaitu teknik yang digunakan untuk memperbesar atau memperkecil gambar pada berbagai tingkatan agar deteksi objek dapat dilakukan pada ukuran yang berbeda.

Fungsinya:

- Memungkinkan algoritma Haar Cascade mendeteksi objek pada berbagai skala dalam gambar (misalnya, objek yang dekat akan tampak lebih besar, sedangkan objek yang jauh akan tampak lebih kecil).
- scaleFactor mengontrol rasio pengurangan ukuran gambar pada setiap iterasi.

Contoh:

- Jika scaleFactor diatur ke 1.1, maka pada setiap iterasi, ukuran gambar akan dikurangi sebesar 10%.
- Jika diatur ke 1.05, maka ukuran gambar akan dikurangi sebesar 5%, sehingga deteksi akan lebih akurat tetapi lebih lambat karena memerlukan lebih banyak iterasi.

Penyesuaian:

- Nilai lebih besar (misalnya 1.5): Proses lebih cepat, tetapi mungkin melewatkan beberapa objek.
- Nilai lebih kecil (misalnya 1.05): Proses lebih lambat, tetapi deteksi lebih akurat.

2. `minNeighbors`

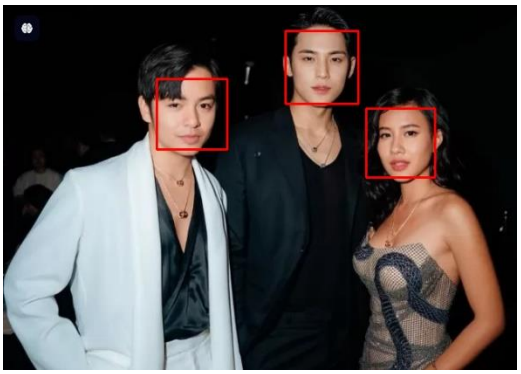
Deskripsi: Menentukan jumlah minimum tetangga persegi panjang yang diperlukan untuk mempertimbangkan objek sebagai deteksi yang valid.

Algoritma ini mengontrol berapa banyak tetangga yang harus dimiliki oleh setiap jendela agar area di jendela tersebut dianggap sebagai wajah. Ketika algoritme menganggap sebuah wajah berada di suatu wilayah, skor kepercayaan yang lebih tinggi akan dikembalikan. Jika ada cukup banyak nilai kepercayaan yang tinggi di area tertentu, maka kaskade akan melaporkan deteksi positif. Ini berarti pengklasifikasi kaskade akan mendeteksi beberapa jendela di sekitar wajah. Parameter ini memberi tahu algoritme berapa banyak tetangga (persegi panjang) yang perlu dideteksi agar jendela diberi label wajah.

Pengaruh:

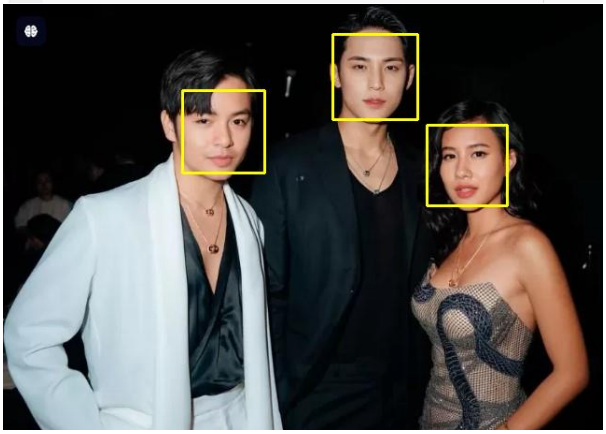
- Nilai rendah (misalnya 3) dapat mendeteksi lebih banyak objek, termasuk false positives.
- Nilai tinggi (misalnya 6) mengurangi false positives tetapi mungkin melewatkan beberapa deteksi valid.

Contoh: Jika minNeighbors = 5, maka algoritma memerlukan setidaknya lima deteksi tetangga untuk menganggap suatu area sebagai objek.



minNeighbors

```
image = cv2.imread('people2.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.2, minNeighbors=4)
for (x, y, w, h) in detections:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 2)
cv2_imshow(image)
```



3. `minSize` dan `maxSize`

Deskripsi: Menentukan ukuran minimum dan maksimum dari objek yang akan dideteksi dalam piksel.

Pengaruh:

- Parameter ini membantu menyaring deteksi objek yang ukurannya terlalu kecil atau terlalu besar.

Contoh:

- minSize = (30, 30) memastikan hanya mendeteksi objek yang ukurannya minimal 30x30 piksel.
- maxSize mencegah pendeteksian objek besar yang mungkin tidak relevan.

minSize and maxSize

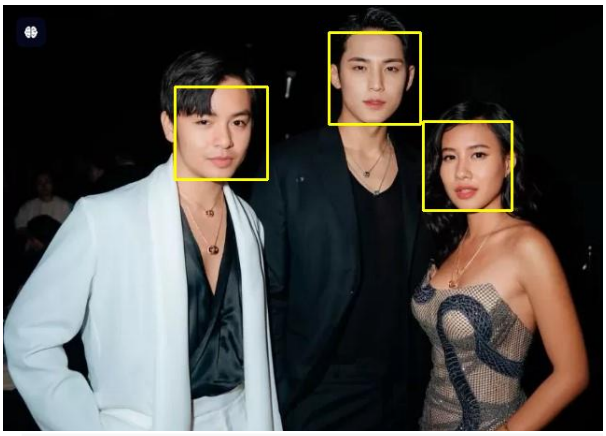
```
[92] image = cv2.imread("people2.jpg")
     image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     detections = face_detector.detectMultiScale(image_gray, minSize = (76,76))

     for (x, y, w, h) in detections:
       print(w, h)
       cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 2)
     cv2_imshow(image)

     98 98
     99 99
     95 95
```

Haarcascade parameters

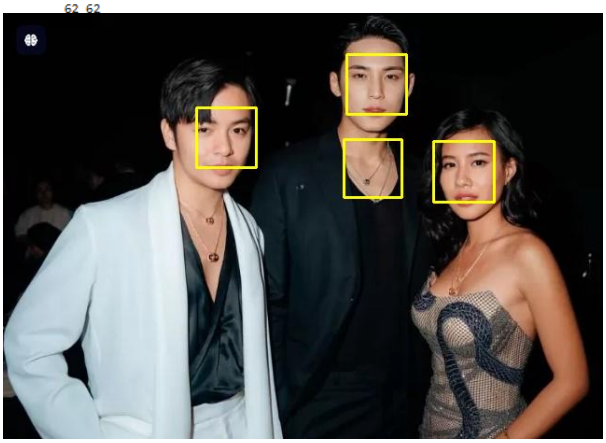
scaleFactor

```
image = cv2.imread('people2.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print(image_gray.shape)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.2)
for (x, y, w, h) in detections:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,0,255), 2)
cv2_imshow(image)
```

```
(456, 643)
```



```
[93] image = cv2.imread("people2.jpg")
     image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     detections = face_detector.detectMultiScale(image_gray, maxSize = (70,70))

     for (x, y, w, h) in detections:
       print(w, h)
       cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 2)
     cv2_imshow(image)

64 64
64 64
65 65
62 62
```

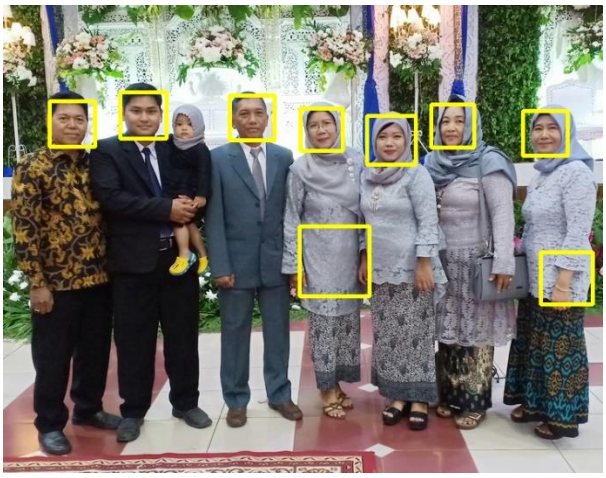


Pada gambar ini, terdapat 4 kotak kuning yang terdeteksi meskipun seharusnya hanya ada 3 wajah manusia nyata. Hal ini terjadi karena algoritma deteksi wajah mungkin mendeteksi area lain yang memiliki karakteristik menyerupai wajah.

Penyebabnya adalah algoritma deteksi menggunakan maxSize=(70,70).

ParametermaxSize=(70,70) membatasi ukuran maksimum wajah yang akan dideteksi. Wajah yang terdeteksi dalam gambar semuanya memiliki dimensi kecil (sekitar 64x64 hingga 66x66 piksel), sehingga memenuhi kriteria ukuran maksimum.

```
[100] image = cv2.imread("people145.jpg")
      image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
      detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.1, minSize=(50,50))

      for (x, y, w, h) in detections:
        print(w, h)
        cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 3)
      cv2_imshow(image)

57 57
55 55
56 56
53 53
52 52
55 55
60 60
64 64
86 86
```



KESIMPULAN DAN SARAN

Kesimpulan

Penelitian ini menunjukkan bahwa metode Haar Cascade adalah salah satu algoritma yang efektif untuk mendeteksi wajah secara real-time. Dengan menggunakan pustaka OpenCV, Haar Cascade dapat mengenali fitur wajah berdasarkan pola tertentu seperti mata, hidung, dan bibir. Penelitian ini berhasil membuktikan bahwa metode ini memiliki kecepatan dan akurasi yang memadai untuk aplikasi deteksi wajah, terutama dalam kondisi pencahayaan yang baik dan sudut pandang yang jelas. Namun, performa algoritma dapat menurun jika terdapat kondisi pencahayaan

yang buruk, sudut pandang yang kompleks, atau adanya objek yang mengganggu di sekitar wajah.

Saran

1. Peningkatan Dataset Pelatihan
   Untuk meningkatkan akurasi deteksi, perlu dilakukan pelatihan ulang model Haar Cascade menggunakan dataset yang lebih beragam, mencakup berbagai kondisi pencahayaan, sudut pandang, dan ekspresi wajah.
2. Penggunaan Metode Tambahan
   Kombinasi Haar Cascade dengan metode lain, seperti algoritma berbasis deep learning (contohnya CNN atau YOLO), dapat meningkatkan kemampuan deteksi wajah, terutama pada kondisi yang kompleks.
3. Optimasi lingkungan kerja
   Disarankan untuk memastikan kondisi pencahayaan yang memadai saat menggunakan Haar Cascade, agar performa deteksi lebih optimal.
4. Pengembangan aplikasi lanjutan
   Metode ini dapat diintegrasikan dengan teknologi lain untuk pengembangan aplikasi berbasis pengenalan wajah, seperti sistem keamanan, pengawasan, atau interaksi manusia-mesin.
5. Evaluasi lebih lanjut
   Perlu dilakukan evaluasi lebih lanjut terkait waktu pemrosesan dan konsumsi sumber daya pada perangkat dengan spesifikasi rendah, untuk memastikan metode ini tetap efisien dan dapat digunakan secara luas.

DAFTAR PUSTAKA

Kosasih, R., & Daomara, C. (2021). Pengenalan Wajah dengan Menggunakan Metode Local Binary Patterns Histograms (LBPH). In JURNAL MEDIA INFORMATIKA BUDIDARMA (Vol. 5, Issue 4, p. 1258). https://doi.org/10.30865/mib.v5i4.3171

Putra, T. W. A., Adi, K., & Isnanto, R. R. (2013). Pengenalan Wajah dengan Matriks Kookurensi Aras Keabuan dan Jaringan Syaraf Tiruan Probabilistik. In JURNAL SISTEM INFORMASI BISNIS (Vol. 3, Issue 2). Diponegoro University. https://doi.org/10.21456/vol3iss2pp82-94

Rifki, K., Priambodho, J., & Musthofa, A. (2021). Pengenalan Plat Nomor dan Wajah Pengendara Menggunakan Convolutional Neural Network dan Metode Absolute Difference pada Sistem Gerbang Otomatis. In Jurnal Teknik ITS (Vol. 10, Issue 2). Lembaga Penelitian dan Pengabdian kepada Masyarakat (LP2M). https://doi.org/10.12962/j23373539.v10i2.72508

Setiawan, D., Putra, A. D., Stefani, K., & Felisa, J. (2021). Implementasi Convolutional Neural Network untuk Facial Recognition. In Media Informatika (Vol. 20, Issue 2, p. 66). Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK). https://doi.org/10.37595/mediainfo.v20i2.68

Jiang, N., Yu, W., Tang, S., & Goto, S. (2011). A cascade detector for rapid face detection (p. 155). https://doi.org/10.1109/cspa.2011.5759863

Malhotra, S., Aggarwal, V., Mangal, H., Nagrath, P., & Jain, R. (2021). Comparison between attendance system implemented through haar cascade classifier and face recognition library. In IOP Conference Series Materials Science and Engineering (Vol. 1022, Issue 1, p. 12045). IOP Publishing. https://doi.org/10.1088/1757-899x/1022/1/012045