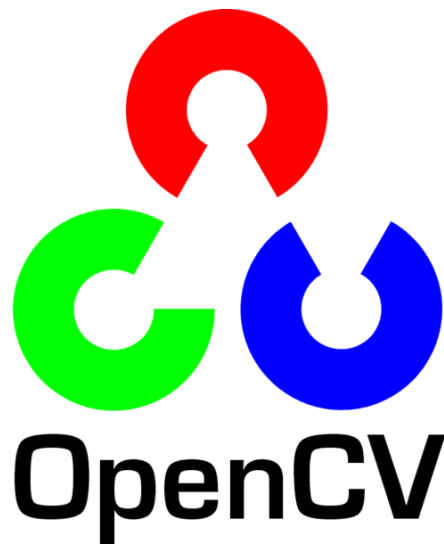


## Tutorial

Computer Vision atau yang lebih dikenal dan disingkat CV merupakan salah satu cara untuk mengolah video dan foto menggunakan barisan code. CV sendiri mendukung berbagai bahasa, contohnya Java, C++ dan pastinya Python yang juga akan kita pakai dalam tutorial Face dan Edge Detector kali ini menggunakan OpenCV.



Sebelum kita membuat program kita untuk Face dan Edge Detector kita memerlukan bantuan modul yaitu modul yang telah saya sebutkan diatas yaitu OpenCV. Untuk menginstall OpenCV kalian bisa menggunakan PIP. Caranya cukup ketikkan Pip Instal Opencv-python di cmd seperti gambar di bawah ini :

```
Command Prompt
Microsoft Windows [Version 10.0.17763.973]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dzihan>pip install opencv-python
Requirement already satisfied: opencv-python in c:\users\dzihan\anaconda3\lib\site-packages (4.1.1.26)
Requirement already satisfied: numpy>=1.14.5 in c:\users\dzihan\anaconda3\lib\site-packages (from opencv-python) (1.16.4)

C:\Users\dzihan>
```

Untuk mengecek apakah modul sudah terinstall kedalam Python kalian. Kalian bisa buka Python, kemudian ketikkan `import cv2`

Pertama-tama ikuti tutorial Program untuk membuat Edge Detector terlebih dahulu. Cara pembuatan Edge Detector ini begitu mudah. Fitur sudah disediakan yaitu fitur built-in dari dalam modul OpenCV itu sendiri.

Fitur ini dapat digunakan cukup dengan memanggil Canny method yang merupakan nama dari fitur built-in OpenCV Python untuk dengan mudah mendeteksi Edge. Tapi sebelum itu pastikan terdapat Camera atau web cam untuk mengikuti tutorial yang live menggunakan camera.

Tapi sebelum memulai Pemrograman yang menggunakan video live dari kamera, lebih baik belajar bagaimana untuk membuat edge detector menggunakan gambar yang sudah ada terlebih dahulu. Langsung saja mulai coding nya.

Hal pertama yang harus kita lakukan adalah import ! Yap, kita akan import modul OpenCV kita dan juga gambar yang akan kita deteksi edge nya. Langsung saja berikut codenya.

```
import cv2

img = cv2.imread('foto-kamu.jpg')
```

Nah silahkan kalian sesuaikan file foto agar dapat terimport dengan benar. Agar lebih mudah dalam hal importing resource seperti ini saya selalu anjurkan agar menaruh file yang akan dipanggil kedalam script python kedalam satu directory atau folder yang sama dengan script Python kita.

Jika quest pertama telah kalian lewati, maka selanjutnya kita akan menggunakan Canny method milik OpenCV untuk diterapkan kedalam gambar kita, langsung saja kalian buat baris code seperti ini `edge = cv2.Canny(img, 70, 70)`

Maksud dari baris code diatas adalah kita mendklarasikan Canny method milik OpenCV tersebut kedalam sebuah variabel. Disana tertera memiliki 3 parameter. Parameter pertama yaitu `img` adalah variabel untuk gambar kita, kemudian 2 parameter selanjutya adalah semacam tingkat kedetailan dari sudut atau edge yang akan di deteksi oleh OpenCV.

Kalian bebas gunakan angka berapa saja untuk 2 parameter terakhir ini, tetapi perlu diingat jika semakin kecil angkanya maka tingkat kedetailan dan kesensitifannya terhadap edge akan semakin besar, yang tentunya juga memberikan noise berlebih kedalam gambar kita. Langkah terakhir adalah memunculkan gambar kita yang telah diubah wujudnya menjadi gambar sudut - sudut, dan mendeskripsikan cara untuk exit dari program kita. Ini sangatlah mudah, silahkan kalian salin terlebih dahulu code ini.

```
cv2.imshow('Edge Detector', edge)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Penjelasan dari code diatas adalah kita menampilkan gambar kita dengan `imshow()` method. Didalamnya tertera 2 parameter, parameter pertama mendefinisikan nama window untuk menampilkan foto kita. Silahkan isi dengan string apa saja terserah kalian, tetapi saya disini menggunakan string 'Edge Detector'.

Kemudian isi parameter kedua adalah nama variabel kita yang digunakan untuk menerapkan Canny method kedalam gambar kita. Kemudian ada `waitKey()` method yang berfungsi untuk menunggu key atau tombol untuk ditekan. Disini kita isi menggunakan parameter 0. Parameter 0 (Nol) berarti kita membolehkan tombol apa saja untuk mengeksekusi kode dibawahnya.

Adapun kita dapat mengisinya dengan angka 1 untuk mendeskripsikan dengan lebih spesifik tombol apa yang akan mengeksekusi code dibawahnya. Dengan membuat method `waitKey()` maka kita akan membuat method untuk keluar dari program dibawahnya. Itu terselesaikan hanya dengan menggunakan `destroyAllWindows()` method, yang fungsinya akan menutup dan exit dari semua windows OpenCV yang terbuka.

Langsung saja kita akan run code kita, tapi sebelumnya pastikan code kalian sudah tampak kira - kira seperti ini

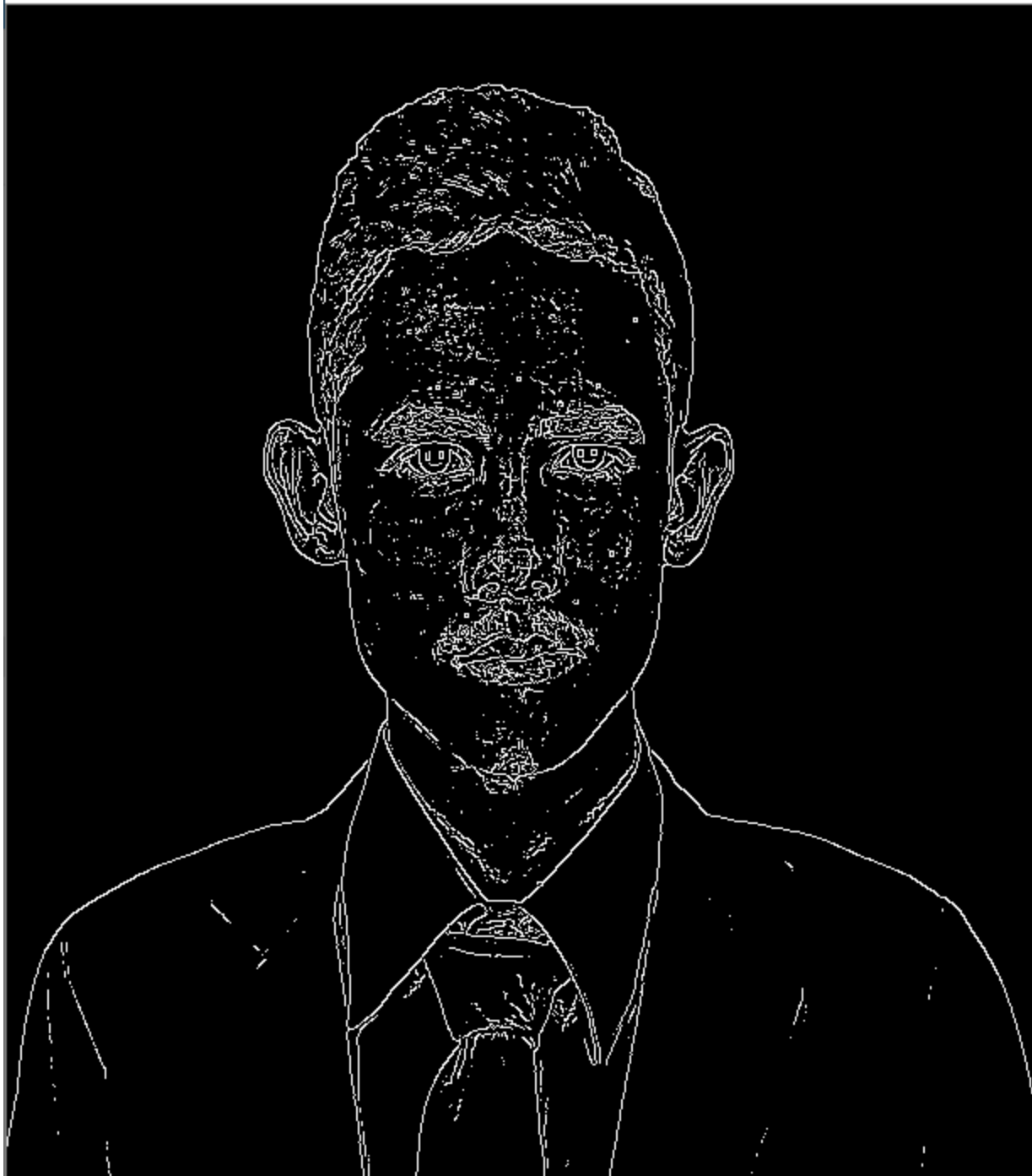
```
import cv2

img = cv2.imread('foto-kamu.jpg')

edge = cv2.Canny(img, 70, 70)

cv2.imshow('Edge Detector', edge)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dan jika code tersebut kalian run maka akan terlihat hasil kurang lebih seperti ini, karena sejatinya ini juga terpengaruh oleh seberapa rumit foto kalian.



Kalau kalian lihat Foto diatas memiliki code yang lebih complex di sisi kiri, itu telah ditambahkan code untuk Face Detection. tetapi saya close window nya untuk meng screenshot khusus edge detecion saja. Oleh karena itu ikuti terus Tutorial Membuat Program Face, Eye dan Edge Detector Menggunakan OpenCV Python ini sampai habis agar kalian ngerti code yang saya tuliskan tersebut.

## **Program Edge Detector OpenCV Python pada Live Video**

Kemudian jika kalian sudah mengerti basic edge detector yang diterapkan kedalam foto. Maka kita akan mencoba menerapkannya kembali kedalam sebuah Live Video dari kamera yang kalian miliki. Kamera ini tentunya harus terhubung dan terbaca oleh OS kita.

Kalian bisa gunakan HandPhone kalian dan hubungkan ke PC kalian, atau menggunakan WebCam baik internal ataupun external. Jika kalian sudah siap dari sisi kamera, maka siapkanlah diri kalian menerima barisan code berikut.

Sebenarnya hanya terdapat sedikit perbedaan dari menerapkan OpenCV kedalam foto dan kedalam video. Perbedaan pertama dari hal import. Yap karena ini merupakan Live Video maka tidak ada file yang harus diimport, tetapi di tangkap. Oleh karena itu kita menggunakan `videoCam = cv2.VideoCapture(0)`

Nah, code diatas berguna sebagai penangan video dari kamera kita. Pendefinisian kameranya terletak pada parameter didalamnya. Untuk menggunakan kamera pada PC kamu, maka gunakanlah 0 (Nol) sebagai parameternya. Tetapi jika kalian menggunakan kamera kedua, maka masukkan 1 sebagai parameternya. Begitu seterusnya.

Kemudian agar video kita dapat terus - menerus di detect edge nya, maka kita perlu menggunakan while loop selamanya. Berikut adalah codenya.

```
while True:
    cond, frame = videoCam.read()
    edge = cv2.Canny(frame, 70, 70)

    cv2.imshow('Edge Detect', edge)

    exit = cv2.waitKey(1) & 0xff
    if exit == ord('q'):
        break

videoCam.release()
cv2.destroyAllWindows()
```

Penjelasan code diatas adalah pertama adalah kita loop selamanya dengan condition True. kemudian kita mendeskripsikan 2 variabel kedalam `videoCam.read()` Maksud penggunaan 2 variabel adalah, variabel pertama yaitu `cond` berarti sebuah kondisi apakah True atau False, jika True maka akan menampilkan video yaitu yang ada pada variabel `frame` dan jika kondisinya nanti berubah False saat di `break` maka video pada `frame` akan selesai.

Kemudian di `show` sama seperti saat menggunakan gambar. Nah seperti yang saya bilang diawal untuk mendeskripsikan tombol apa untuk exit dari program kita gunakan parameter 1 pada `waitKey()` method kemudian diikuti `0xff` sebagai penjas kalau tombol ini khusus dan bukan tombol sembarangan yang akan membreak loop kita.

Selanjutnya kita gunakan conditional untuk menentukan jenis tombol yang akan kita tekan. Disana kita menggunakan `ord()` function untuk menempatkan jenis tombol kita didalamnya sebagai parameter. Di contoh diatas saya membuat tombol 'q' menjadi tombol exit program Edge Detector kita.

2 baris terakhir akan tereksekusi ketika loop berhenti, yaitu `release()` method untuk megatakan pada camera 'eh udah, kita udh exit berhenti ngerekam yah !' kemudian close semua windows dengan `destroyAllWindows()` method.

Jika sudah maka code kalian seharusnya akan terlihat seperti ini

```
import cv2

videoCam = cv2.VideoCapture(0)

while True:
    cond, frame = videoCam.read()
    edge = cv2.Canny(frame, 70, 70)

    cv2.imshow('Edge Detect', edge)

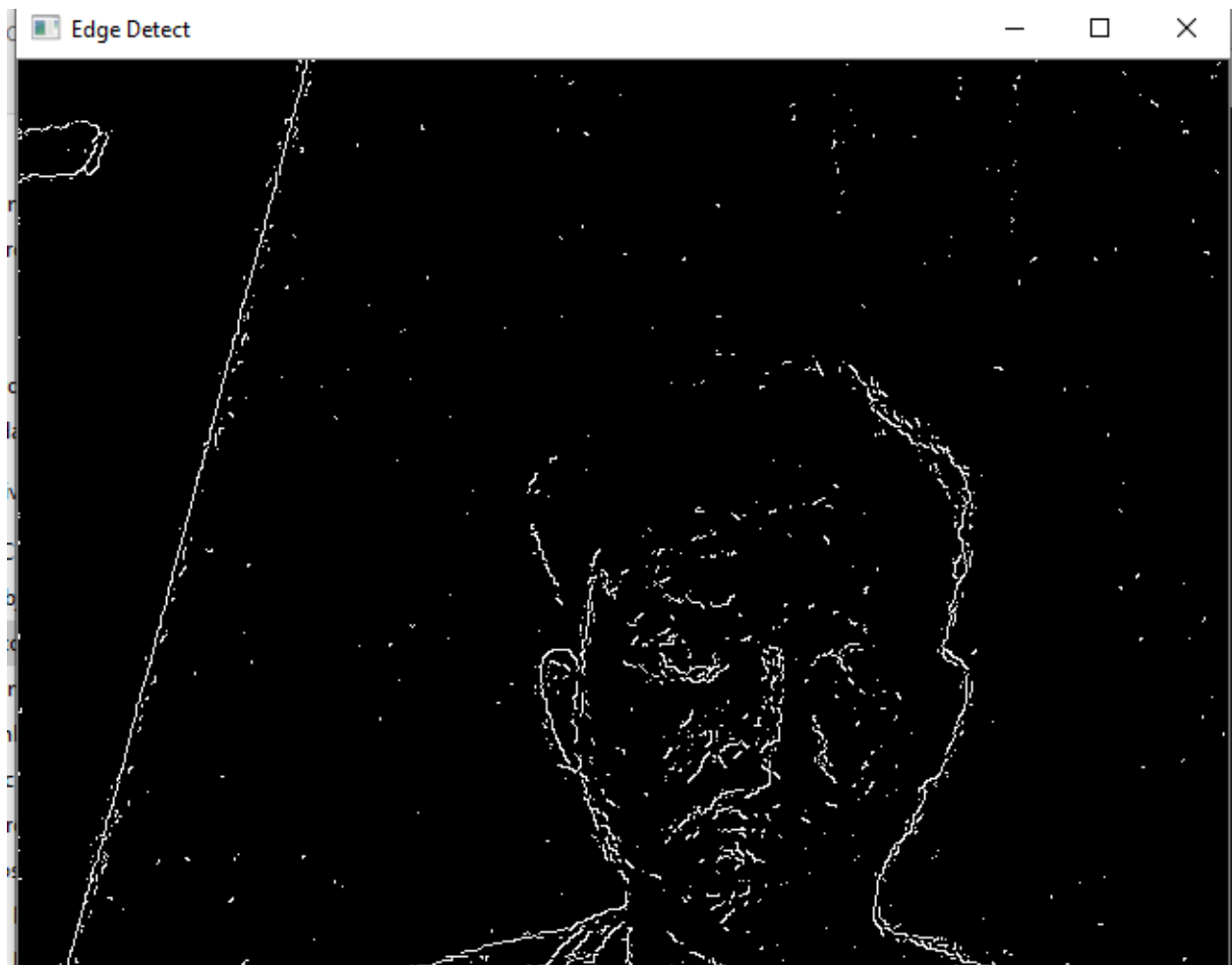
    exit = cv2.waitKey(1) & 0xff
    if exit == ord('q'):
        break

videoCam.release()
cv2.destroyAllWindows()
```

Dan jika kalian Run code kalian akan terlihat seperti ini







# **PROGRAM FACE DAN EYE DETECTOR DENGAN OPENCV PYTHON**

Nah, beberapa fungsi yang kita pakai dalam quest kedua ini tidak lah jauh berbeda dengan saat kita membuat Edge Detector. Hal ini tentunya dikarenakan kita menggunakan modul yang sama yaitu OpenCV Python. Seperti biasa kita akan mulai menerapkan Face dan Eye Detector kepada Gambar terlebih dahulu, baru kemudian kita terapkan kedalam Live Video.

Tapi sebelum kalian memulai untuk membuat Face dan Eye Detector menggunakan OpenCV Python ini. Perlu kalian ketahui kita perlu menggunakan file external yang berisi barisan algoritma untuk mencari wajah dan mata pada gambar dan juga video kita nantinya.

File external ini berbasis XML layaknya AIML tetapi memiliki pendefinisian fungsinya sendiri. File ini dinamakan HaarCascade. Membuat HaarCascade akan begitu lama untuk menemukan racikan yang pas. Hal ini dikarenakan kita perlu melakukan train kepada banyak image untuk mengasah keakuratan file tersebut.

Oleh karena itu kita gunakan saja HaarCascade yang sudah tersedia di pasaran Open Source. Kita bisa menemukannya di GitHub. Kita dalam Tutorial ini menggunakan File HaarCascade untuk Wajah, dan juga File HaarCascade untuk Mata.

Yang pertama namanya `haarcascade_frontalface_default`

Isinya seperti berikut :

```
<?xml version="1.0"?>
<!--
  Stump-based 24x24 discrete(?) adaboost frontal face detector.
  Created by Rainer Lienhart.

  //////////////////////////////////////

  IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.

  By downloading, copying, installing or using the software you agree to this license.
  If you do not agree to this license, do not download, install,
  copy or use the software.

  | | | | | | | | Intel License Agreement
  | | | | | | | | For Open Source Computer Vision Library


  Copyright (C) 2000, Intel Corporation, all rights reserved.
  Third party copyrights are property of their respective owners.

  Redistribution and use in source and binary forms, with or without modification,
  are permitted provided that the following conditions are met:

  * Redistribution's of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

  * Redistribution's in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

  * The name of Intel Corporation may not be used to endorse or promote products
  derived from this software without specific prior written permission.
```

 Do you mind ta

```
Selection View Go Debug Terminal Help haarcascade_frontalface_default.xml - Visual Studio Code
g.py face_datasets.py detectfoto.py coba.py coba2.py haarcascade_eye.xml haarcascade_frontalface_default.xml X
rs > dzihan > Desktop > fix2 > face > haarcascade_frontalface_default.xml
* the name or Intel Corporation may not be used to endorse or promote products
derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and
any express or implied warranties, including, but not limited to, the implied
warranties of merchantability and fitness for a particular purpose are disclaimed.
In no event shall the Intel Corporation or contributors be liable for any direct,
indirect, incidental, special, exemplary, or consequential damages
(including, but not limited to, procurement of substitute goods or services;
loss of use, data, or profits; or business interruption) however caused
and on any theory of liability, whether in contract, strict liability,
or tort (including negligence or otherwise) arising in any way out of
the use of this software, even if advised of the possibility of such damage.
-->
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          </internalNodes>
        </_>
      </weakClassifiers>
    </_>
  </stages>
</cascade>
</opencv_storage>
```

Do you mind taking a quick feedback survey?

Take Survey Remind Me later

```
rs > dzihan > Desktop > fix2 > face > haarcascade_frontalface_default.xml
1 <internalNodes>
2   0 -1 0 -3.1511999666690826e-02</internalNodes>
3 <leafValues>
4   2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
5 </_>
6 <internalNodes>
7   0 -1 1 1.2396000325679779e-02</internalNodes>
8 <leafValues>
9   -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
10 </_>
11 <internalNodes>
12   0 -1 2 2.1927999332547188e-02</internalNodes>
13 <leafValues>
14   -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
15 </_>
16 <internalNodes>
17   0 -1 3 5.7529998011887074e-03</internalNodes>
18 <leafValues>
19   -8.7463897466659546e-01 1.1760339736938477e+00</leafValues></_>
20 </_>
21 <internalNodes>
22   0 -1 4 1.5014000236988068e-02</internalNodes>
23 <leafValues>
24   -7.7945697307586670e-01 1.2608419656753540e+00</leafValues></_>
25 </_>
26 <internalNodes>
27   0 -1 5 9.9371001124382019e-02</internalNodes>
28 <leafValues>
29   5.5751299858093262e-01 -1.8743000030517578e+00</leafValues></_>
30 </_>
31 <internalNodes>
32   0 -1 6 2.7340000960975885e-03</internalNodes>
33 <leafValues>
```

Do you mind taking a quick feedback survey?

Take Survey Remind Me later

```
C:\Users\> dzihan > Desktop > fix2 > face > haarcascade_frontalface_default.xml
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
```

```
<_>
  <internalNodes>
    0 -1 6 2.7340000960975885e-03</internalNodes>
  <leafValues>
    -1.6911929845809937e+00 4.4009700417518616e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 7 -1.8859000876545906e-02</internalNodes>
    <leafValues>
      -1.4769539833068848e+00 4.4350099563598633e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 8 5.9739998541772366e-03</internalNodes>
      <leafValues>
        -8.5909199714660645e-01 8.525559021911621e-01</leafValues></_></weakClassifiers></_>
  <_>
    <maxWeakCount>16</maxWeakCount>
    <stageThreshold>-4.9842400550842285e+00</stageThreshold>
    <weakClassifiers>
      <_>
        <internalNodes>
          0 -1 9 -2.111000088810921e-02</internalNodes>
        <leafValues>
          1.2435649633407593e+00 -1.5713009834289551e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 10 2.0355999469757080e-02</internalNodes>
          <leafValues>
            -1.6204780340194702e+00 1.1817760467529297e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 11 2.1308999508619308e-02</internalNodes>
          <leafValues>
```

Do you mind taking a quick feedback survey?

Take Survey Remind Me later

```
0 -1 1129 -2.4527000263333321e-02</internalNodes>
<leafValues>
  -1.5872869491577148e+00 -2.1817000582814217e-02</leafValues></_>
<_>
  <internalNodes>
    0 -1 1130 2.3024000227451324e-02</internalNodes>
  <leafValues>
    1.6853399574756622e-01 -3.8106900453567505e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 1131 -2.4917000904679298e-02</internalNodes>
    <leafValues>
      5.0810897350311279e-01 -2.7279898524284363e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 1132 1.0130000300705433e-03</internalNodes>
    <leafValues>
      -4.3138799071311951e-01 2.6438099145889282e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 1133 1.5603000298142433e-02</internalNodes>
    <leafValues>
      -3.1624200940132141e-01 5.5715900659561157e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 1134 -2.6685999706387520e-02</internalNodes>
    <leafValues>
      1.0553920269012451e+00 2.9074000194668770e-02</leafValues></_>
  <_>
    <internalNodes>
      0 -1 1135 1.3940000208094716e-03</internalNodes>
    <leafValues>
```

```

    0 -1 238 1.274250007152557e-01</internalNodes>
  <leafValues>
    2.0165599882602692e-01 -1.5467929840087891e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 239 4.7703001648187637e-02</internalNodes>
  <leafValues>
    -3.7937799096107483e-01 3.7885999679565430e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 240 5.3608000278472900e-02</internalNodes>
  <leafValues>
    2.1220499277114868e-01 -1.2399710416793823e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 241 -3.9680998772382736e-02</internalNodes>
  <leafValues>
    -1.0257550477981567e+00 5.1282998174428940e-02</leafValues></_>
<_>
  <internalNodes>
    0 -1 242 -6.7327000200748444e-02</internalNodes>
  <leafValues>
    -1.0304750204086304e+00 2.3005299270153046e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 243 1.3337600231170654e-01</internalNodes>
  <leafValues>
    -2.0869000256061554e-01 1.2272510528564453e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 244 -2.0919300615787506e-01</internalNodes>
  <leafValues>
    8.7020808500442505e-01 -4.4254000607801437e-02</leafValues></_>

```

```

rs > dzhnan > Desktop > fix2 > face > HaarCascadeFrontalfaceDefault.xml
    0 -1 245 -6.55890003264904022e-02</internalNodes>
  <leafValues>
    1.0443429946899414e+00 -2.1682099997997284e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 246 6.1882998794317245e-02</internalNodes>
  <leafValues>
    1.3798199594020844e-01 -1.9009059667587280e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 247 -2.5578999891877174e-02</internalNodes>
  <leafValues>
    -1.6607600450515747e+00 5.8439997956156731e-03</leafValues></_>
<_>
  <internalNodes>
    0 -1 248 -3.4827001392841339e-02</internalNodes>
  <leafValues>
    7.9940402507781982e-01 -8.2406997680664062e-02</leafValues></_>
<_>
  <internalNodes>
    0 -1 249 -1.8209999427199364e-02</internalNodes>
  <leafValues>
    -9.6073997020721436e-01 6.6320002079010010e-02</leafValues></_>
<_>
  <internalNodes>
    0 -1 250 1.5070999972522259e-02</internalNodes>
  <leafValues>
    1.9899399578571320e-01 -7.6433002948760986e-01</leafValues></_></weakClassifiers></_>
<_>
  <maxWeakCount>72</maxWeakCount>
  <stageThreshold>-3.8832089900970459e+00</stageThreshold>
</weakClassifiers>

```

```

1.9899399578571320e-01 -7.6433002948760986e-01</leafValues></_></weakClassifiers></_>
<_>
<maxWeakCount>72</maxWeakCount>
<stageThreshold>-3.8832089900970459e+00</stageThreshold>
<weakClassifiers>
<_>
<internalNodes>
0 -1 251 4.6324998140335083e-02</internalNodes>
<leafValues>
-1.0362670421600342e+00 8.2201498746871948e-01</leafValues></_>
<_>
<internalNodes>
0 -1 252 1.5406999737024307e-02</internalNodes>
<leafValues>
-1.2327589988708496e+00 2.9647698998451233e-01</leafValues></_>
<_>
<internalNodes>
0 -1 253 1.2808999978005886e-02</internalNodes>
<leafValues>
-7.5852298736572266e-01 5.7985502481460571e-01</leafValues></_>
<_>
<internalNodes>
0 -1 254 4.9150999635457993e-02</internalNodes>
<leafValues>
-3.8983899354934692e-01 8.9680302143096924e-01</leafValues></_>
<_>
<internalNodes>
0 -1 255 1.2621000409126282e-02</internalNodes>
<leafValues>
-7.1799302101135254e-01 5.0440901517868042e-01</leafValues></_>
<_>
<internalNodes>

```

```

-7.1799302101135254e-01 5.0440901517868042e-01</leafValues></_>
<_>
<internalNodes>
0 -1 256 -1.8768999725580215e-02</internalNodes>
<leafValues>
5.5147600173950195e-01 -7.0555400848388672e-01</leafValues></_>
<_>
<internalNodes>
0 -1 257 4.1965000331401825e-02</internalNodes>
<leafValues>
-4.4782099127769470e-01 7.0985502004623413e-01</leafValues></_>
<_>
<internalNodes>
0 -1 258 -5.1401998847723007e-02</internalNodes>
<leafValues>
-1.0932120084762573e+00 2.6701900362968445e-01</leafValues></_>
<_>
<internalNodes>
0 -1 259 -7.0960998535156250e-02</internalNodes>
<leafValues>
8.3618402481079102e-01 -3.8318100571632385e-01</leafValues></_>
<_>
<internalNodes>
0 -1 260 1.6745999455451965e-02</internalNodes>
<leafValues>
-2.5733101367950439e-01 2.5966501235961914e-01</leafValues></_>
<_>
<internalNodes>
0 -1 261 -6.2400000169873238e-03</internalNodes>
<leafValues>
3.1631499528884888e-01 -5.8796900510787964e-01</leafValues></_>
<_>
<internalNodes>

```



```

    0 -1 262 -3.939/9996442/9480e-02</internalNodes>
  <leafValues>
    -1.049121022244263e+00 1.6822400689125061e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 263 0.</internalNodes>
  <leafValues>
    1.6144199669361115e-01 -8.7876898050308228e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 264 -2.2307999432086945e-02</internalNodes>
  <leafValues>
    -6.9053500890731812e-01 2.3607000708580017e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 265 1.8919999711215496e-03</internalNodes>
  <leafValues>
    2.4989199638366699e-01 -5.6583297252655029e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 266 1.0730000212788582e-03</internalNodes>
  <leafValues>
    -5.0415802001953125e-01 3.8374501466751099e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 267 3.9230998605489731e-02</internalNodes>
  <leafValues>
    4.2619001120328903e-02 -1.3875889778137207e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 268 6.2238000333309174e-02</internalNodes>
  <leafValues>
    1.4119400084018707e-01 -1.0688860416412354e+00</leafValues></_>

```

```

    1.4119400084018707e-01 -1.0688860416412354e+00</leafValues></_>
<_>
  <internalNodes>
    0 -1 269 2.1399999968707561e-03</internalNodes>
  <leafValues>
    -8.9622402191162109e-01 1.9796399772167206e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 270 9.1800000518560410e-04</internalNodes>
  <leafValues>
    -4.5337298512458801e-01 4.3532699346542358e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 271 -6.9169998168945312e-03</internalNodes>
  <leafValues>
    3.3822798728942871e-01 -4.4793000817298889e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 272 -2.3866999894380569e-02</internalNodes>
  <leafValues>
    -7.8908598423004150e-01 2.2511799633502960e-01</leafValues></_>
<_>
  <internalNodes>
    0 -1 273 -1.0262800008058548e-01</internalNodes>
  <leafValues>
    -2.2831439971923828e+00 -5.3960001096129417e-03</leafValues></_>
<_>
  <internalNodes>
    0 -1 274 -9.5239998772740364e-03</internalNodes>
  <leafValues>
    3.9346700906753540e-01 -5.2242201566696167e-01</leafValues></_>
  <internalNodes>

```

```
1.4119400084018/0/e-01 -1.0688860416412354e+00</leafValues></_>
<_>
<internalNodes>
  0 -1 269 2.139999996870561e-03</internalNodes>
<leafValues>
  -8.9622402191162109e-01 1.9796399772167206e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 270 9.1800000518560410e-04</internalNodes>
<leafValues>
  -4.5337298512458801e-01 4.3532699346542358e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 271 -6.9169998168945312e-03</internalNodes>
<leafValues>
  3.3822798728942871e-01 -4.4793000817298889e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 272 -2.3866999894380569e-02</internalNodes>
<leafValues>
  -7.8908598423004150e-01 2.2511799633502960e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 273 -1.0262800008058548e-01</internalNodes>
<leafValues>
  -2.2831439971923828e+00 -5.3960001096129417e-03</leafValues></_>
<_>
<internalNodes>
  0 -1 274 -9.5239998772740364e-03</internalNodes>
<leafValues>
  3.9346700906753540e-01 -5.2242201566696167e-01</leafValues></_>
<_>
<internalNodes>
```

```
<internalNodes>
  0 -1 274 -9.5239998772740364e-03</internalNodes>
<leafValues>
  3.9346700906753540e-01 -5.2242201566696167e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 275 3.9877001196146011e-02</internalNodes>
<leafValues>
  3.2799001783132553e-02 -1.5079489946365356e+00</leafValues></_>
<_>
<internalNodes>
  0 -1 276 -1.3144999742507935e-02</internalNodes>
<leafValues>
  -1.0839990377426147e+00 1.8482400476932526e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 277 -5.0590999424457550e-02</internalNodes>
<leafValues>
  -1.8822289705276489e+00 -2.2199999075382948e-03</leafValues></_>
<_>
<internalNodes>
  0 -1 278 2.4917000904679298e-02</internalNodes>
<leafValues>
  1.4593400061130524e-01 -2.2196519374847412e+00</leafValues></_>
<_>
<internalNodes>
  0 -1 279 -7.6370001770555973e-03</internalNodes>
<leafValues>
  -1.0164569616317749e+00 5.8797001838684082e-02</leafValues></_>
<_>
<internalNodes>
  0 -1 280 4.2911998927593231e-02</internalNodes>
<leafValues>
```

```

0 -1 282 9.692999022237014e-03</internalNodes>
<leafValues>
-1.145009945068359e-01 7.1091300249099731e-01</leafValues></_>
<_>
<internalNodes>
0 -1 283 1.1145000346004963e-02</internalNodes>
<leafValues>
7.000099867582321e-02 -1.0534820556640625e+00</leafValues></_>
<_>
<internalNodes>
0 -1 284 -5.2453000098466873e-02</internalNodes>
<leafValues>
-1.7594360113143921e+00 1.9523799419403076e-01</leafValues></_>
<_>
<internalNodes>
0 -1 285 -2.3020699620246887e-01</internalNodes>
<leafValues>
9.5840299129486084e-01 -2.5045698881149292e-01</leafValues></_>
<_>
<internalNodes>
0 -1 286 -1.6365999355912209e-02</internalNodes>
<leafValues>
4.6731901168823242e-01 -2.1108399331569672e-01</leafValues></_>
<_>
<internalNodes>
0 -1 287 -1.7208000645041466e-02</internalNodes>
<leafValues>
7.0835697650909424e-01 -2.8018298745155334e-01</leafValues></_>
<_>
<internalNodes>
0 -1 288 -3.6648001521825790e-02</internalNodes>
<leafValues>
-1.1013339757919312e+00 2.4341100454330444e-01</leafValues></_>

```

```

-1.1013339757919312e+00 2.4341100454330444e-01</leafValues></_>
<_>
<internalNodes>
0 -1 289 -1.0304999537765980e-02</internalNodes>
<leafValues>
-1.0933129787445068e+00 5.6258998811244965e-02</leafValues></_>
<_>
<internalNodes>
0 -1 290 -1.3713000342249870e-02</internalNodes>
<leafValues>
-2.6438099145889282e-01 1.9821000099182129e-01</leafValues></_>
<_>
<internalNodes>
0 -1 291 2.9308000579476357e-02</internalNodes>
<leafValues>
-2.2142399847507477e-01 1.0525950193405151e+00</leafValues></_>
<_>
<internalNodes>
0 -1 292 2.4077000096440315e-02</internalNodes>
<leafValues>
1.8485699594020844e-01 -1.7203969955444336e+00</leafValues></_>
<_>
<internalNodes>
0 -1 293 6.1280000954866409e-03</internalNodes>
<leafValues>
-9.2721498012542725e-01 5.8752998709678650e-02</leafValues></_>
<_>
<internalNodes>
0 -1 294 -2.2377999499440193e-02</internalNodes>
<leafValues>
1.9646559953689575e+00 2.7785999700427055e-02</leafValues></_>
<_>
<internalNodes>

```

```

1.96465599536895/5e+00 2.7785999/0042/055e-02</leafValues></_>
<_>
<internalNodes>
  0 -1 295 -7.044000854432583e-03</internalNodes>
<leafValues>
  2.1427600085735321e-01 -4.8407599329948425e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 296 -4.0603000670671463e-02</internalNodes>
<leafValues>
  -1.1754349470138550e+00 1.6061200201511383e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 297 -2.4466000497341156e-02</internalNodes>
<leafValues>
  -1.1239900588989258e+00 4.1110001504421234e-02</leafValues></_>
<_>
<internalNodes>
  0 -1 298 2.5309999473392963e-03</internalNodes>
<leafValues>
  -1.7169700562953949e-01 3.2178801298141479e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 299 -1.9588999450206757e-02</internalNodes>
<leafValues>
  8.2720202207565308e-01 -2.6376700401306152e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 300 -2.9635999351739883e-02</internalNodes>
<leafValues>
  -1.1524770259857178e+00 1.4999300241470337e-01</leafValues></_>
<_>
<internalNodes>

```

```

-1.1524770259857178e+00 1.4999300241470337e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 301 -1.5030000358819962e-02</internalNodes>
<leafValues>
  -1.0491830110549927e+00 4.0160998702049255e-02</leafValues></_>
<_>
<internalNodes>
  0 -1 302 -6.0715001076459885e-02</internalNodes>
<leafValues>
  -1.0903840065002441e+00 1.5330800414085388e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 303 -1.2790000066161156e-02</internalNodes>
<leafValues>
  4.2248600721359253e-01 -4.2399200797080994e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 304 -2.0247999578714371e-02</internalNodes>
<leafValues>
  -9.1866999864578247e-01 1.8485699594020844e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 305 -3.0683999881148338e-02</internalNodes>
<leafValues>
  -1.5958670377731323e+00 2.5760000571608543e-03</leafValues></_>
<_>
<internalNodes>
  0 -1 306 -2.0718000829219818e-02</internalNodes>
<leafValues>
  -6.6299998760223389e-01 3.1037199497222900e-01</leafValues></_>
<_>

```

```

9 <_>
10 <internalNodes>
11 | 0 -1 11 2.1308999508619308e-02</internalNodes>
12 <leafValues>
13 | -1.9415930509567261e+00 7.0069098472595215e-01</leafValues></_>
14 <_>
15 <internalNodes>
16 | 0 -1 12 9.1660000383853912e-02</internalNodes>
17 <leafValues>
18 | -5.5670100450515747e-01 1.7284419536590576e+00</leafValues></_>
19 <_>
20 <internalNodes>
21 | 0 -1 13 3.6288000643253326e-02</internalNodes>
22 <leafValues>
23 | 2.6763799786567688e-01 -2.1831810474395752e+00</leafValues></_>
24 <_>
25 <internalNodes>
26 | 0 -1 14 -1.9109999760985374e-02</internalNodes>
27 <leafValues>
28 | -2.6730210781097412e+00 4.5670801401138306e-01</leafValues></_>
29 <_>
30 <internalNodes>
31 | 0 -1 15 8.2539999857544899e-03</internalNodes>
32 <leafValues>
33 | -1.0852910280227661e+00 5.3564202785491943e-01</leafValues></_>
34 <_>
35 <internalNodes>
36 | 0 -1 16 1.8355000764131546e-02</internalNodes>
37 <leafValues>
38 | -3.5200199484825134e-01 9.3339198827743530e-01</leafValues></_>
39 <_>
40 <internalNodes>
41 | 0 -1 17 -7.0569999516010284e-03</internalNodes>

```

Do you m

```

9657 <_>
9658 <_>
9659 | 19 13 2 11 2.</_></rects></_>
9660 <_>
9661 <rects>
9662 <_>
9663 | 1 2 4 22 -1.</_>
9664 <_>
9665 | 1 2 2 11 2.</_>
9666 <_>
9667 | 3 13 2 11 2.</_></rects></_>
9668 <_>
9669 <rects>
9670 <_>
9671 | 15 0 2 24 -1.</_>
9672 <_>
9673 | 15 0 1 24 2.</_></rects></_>
9674 <_>
9675 <rects>
9676 <_>
9677 | 3 20 16 4 -1.</_>
9678 <_>
9679 | 11 20 8 4 2.</_></rects></_>
9680 <_>
9681 <rects>
9682 <_>
9683 | 11 6 4 18 -1.</_>
9684 <_>
9685 | 13 6 2 9 2.</_>
9686 <_>
9687 | 11 15 2 9 2.</_></rects></_>
9688 <_>

```

```

616 |         <_>
617 |         | 14 15 9 6 -1.</_>
618 |         <_>
619 |         | 14 17 9 2 3.</_></rects></_>
620 |     <_>
621 |     <rects>
622 |     <_>
623 |     | 0 20 18 4 -1.</_>
624 |     <_>
625 |     | 0 20 9 2 2.</_>
626 |     <_>
627 |     | 9 22 9 2 2.</_></rects></_>
628 | <_>
629 | <rects>
630 | <_>
631 | | 13 18 9 6 -1.</_>
632 | <_>
633 | | 13 20 9 2 3.</_></rects></_>
634 | <_>
635 | <rects>
636 | <_>
637 | | 2 18 9 6 -1.</_>
638 | <_>
639 | | 2 20 9 2 3.</_></rects></_>
640 | <_>
641 | <rects>
642 | <_>
643 | | 6 16 18 3 -1.</_>
644 | <_>
645 | | 6 17 18 1 3.</_></rects></_>
646 | <_>
647 | <rects>
648 | <_>

```

```

64 |         <_>
65 |         | 15 3 2 21 -1.</_>
66 |         <_>
67 |         | 15 3 1 21 2.</_></rects></_>
68 |     <_>
69 |     <rects>
70 |     <_>
71 |     | 7 0 2 23 -1.</_>
72 |     <_>
73 |     | 8 0 1 23 2.</_></rects></_>
74 | <_>
75 | <rects>
76 | <_>
77 | | 15 8 9 4 -1.</_>
78 | <_>
79 | | 15 10 9 2 2.</_></rects></_>
80 | <_>
81 | <rects>
82 | <_>
83 | | 0 8 9 4 -1.</_>
84 | <_>
85 | | 0 10 9 2 2.</_></rects></_>
86 | <_>
87 | <rects>
88 | <_>
89 | | 8 14 9 6 -1.</_>
90 | <_>
91 | | 8 16 9 2 3.</_></rects></_>
92 | <_>
93 | <rects>
94 | <_>
95 | | 0 14 9 6 -1.</_>
96 | <_>

```

```

15134 | <_>
15135 | 4 4 16 12 -1.</_>
15136 | <_>
15137 | 4 10 16 6 2.</_></rects></_>
15138 | <_>
15139 | <rects>
15140 | <_>
15141 | 0 1 4 20 -1.</_>
15142 | <_>
15143 | 2 1 2 20 2.</_></rects></_>
15144 | <_>
15145 | <rects>
15146 | <_>
15147 | 3 0 18 2 -1.</_>
15148 | <_>
15149 | 3 1 18 1 2.</_></rects></_>
15150 | <_>
15151 | <rects>
15152 | <_>
15153 | 1 5 20 14 -1.</_>
15154 | 8 8
15155 | 1 5 10 7 2.</_>
15156 | <_>
15157 | 11 12 10 7 2.</_></rects></_>
15158 | <_>
15159 | <rects>
15160 | <_>
15161 | 5 8 14 12 -1.</_>
15162 | <_>
15163 | 5 12 14 4 3.</_></rects></_>
15164 | <_>
15165 | <rects>

```

```

15166 | <_>
15167 | 3 14 7 9 -1.</_>
15168 | <_>
15169 | 3 17 7 3 3.</_></rects></_>
15170 | <_>
15171 | <rects>
15172 | <_>
15173 | 14 15 9 6 -1.</_>
15174 | <_>
15175 | 14 17 9 2 3.</_></rects></_>
15176 | <_>
15177 | <rects>
15178 | <_>
15179 | 1 15 9 6 -1.</_>
15180 | <_>
15181 | 1 17 9 2 3.</_></rects></_>
15182 | <_>
15183 | <rects>
15184 | <_>
15185 | 11 6 8 10 -1.</_>
15186 | <_>
15187 | 15 6 4 5 2.</_>
15188 | <_>
15189 | 11 11 4 5 2.</_></rects></_>
15190 | <_>
15191 | <rects>
15192 | <_>

```

```

7      | 15 6 4 5 2.</_>
8      |<_>
9      | 11 11 4 5 2.</_></rects></_>
10     |<_>
11     |<rects>
12     |<_>
13     | 5 5 14 14 -1.</_>
14     |<_>
15     | 5 5 7 7 2.</_>
16     |<_>
17     | 12 12 7 7 2.</_></rects></_>
18     |<_>
19     |<rects>
20     |<_>
21     | 6 0 12 5 -1.</_>
22     |<_>
23     | 10 0 4 5 3.</_></rects></_>
24     |<_>
25     |<rects>
26     |<_>
27     | 9 0 6 9 -1.</_>
28     |<_>
29     | 9 3 6 3 3.</_></rects></_>
30     |<_>
31     |<rects>
32     |<_>
33     | 9 6 6 9 -1.</_>
34     |<_>
35     | 11 6 2 9 3.</_></rects></_>
36     |<_>
37     |<rects>
38     |<_>

```

```

18     |<_>
19     | 7 0 6 9 -1.</_>
20     |<_>
21     | 9 0 2 9 3.</_></rects></_>
22     |<_>
23     |<rects>
24     |<_>
25     | 10 6 6 9 -1.</_>
26     |<_>
27     | 12 6 2 9 3.</_></rects></_>
28     |<_>
29     |<rects>
30     |<_>
31     | 8 6 6 9 -1.</_>
32     |<_>
33     | 10 6 2 9 3.</_></rects></_>
34     |<_>
35     |<rects>
36     |<_>
37     | 3 8 18 4 -1.</_>
38     |<_>
39     | 9 8 6 4 3.</_></rects></_>
40     |<_>
41     |<rects>
42     |<_>
43     | 6 0 12 9 -1.</_>
44     |<_>
45     | 6 3 12 3 3.</_></rects></_>
46     |<_>
47     |<rects>
48     |<_>
49     | 0 0 24 6 -1.</_>
50     |<_>

```



```

48     <_>
49     | 0 0 24 6 -1.</_>
50     <_>
51     | 8 0 8 6 3.</_></rects></_>
52     <_>
53     <rects>
54     <_>
55     | 4 7 16 12 -1.</_>
56     <_>
57     | 4 11 16 4 3.</_></rects></_>
58     <_>
59     <rects>
60     <_>
61     | 11 6 6 6 -1.</_>
62     <_>
63     | 11 6 3 6 2.</_></rects></_>
64     <_>
65     <rects>
66     <_>
67     | 0 20 24 3 -1.</_>
68     <_>
69     | 8 20 8 3 3.</_></rects></_>
70     <_>
71     <rects>
72     <_>
73     | 11 6 4 9 -1.</_>
74     <_>
75     | 11 6 2 9 2.</_></rects></_>
76     <_>
77     <rects>
78     <_>
79     | 4 13 15 4 -1.</_>
80     <_>

```

```

    <_>
    | 9 6 6 16 -1.</_>
    <_>
    | 9 14 6 8 2.</_></rects></_>
    <_>
    <rects>
    <_>
    | 6 6 7 12 -1.</_>
    <_>
    | 6 10 7 4 3.</_></rects></_>
    <_>
    <rects>
    <_>
    | 14 6 6 9 -1.</_>
    <_>
    | 14 9 6 3 3.</_></rects></_>
    <_>
    <rects>
    <_>
    | 5 14 6 9 -1.</_>
    <_>
    | 5 17 6 3 3.</_></rects></_>
    <_>
    <rects>
    <_>
    | 10 8 6 9 -1.</_>
    <_>
    | 12 8 2 9 3.</_></rects></_>
    <_>
    <rects>
    <_>
    | 6 6 4 18 -1.</_>
    <_>

```

```
<_>
<internalNodes>
  0 -1 17 -7.0569999516010284e-03</internalNodes>
<leafValues>
  9.2782098054885864e-01 -6.6349899768829346e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 18 -9.8770000040531158e-03</internalNodes>
<leafValues>
  1.1577470302581787e+00 -2.9774799942970276e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 19 1.5814000740647316e-02</internalNodes>
<leafValues>
  -4.1960600018501282e-01 1.3576040267944336e+00</leafValues></_>
<_>
<internalNodes>
  0 -1 20 -2.0700000226497650e-02</internalNodes>
<leafValues>
  1.4590020179748535e+00 -1.9739399850368500e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 21 -1.3760800659656525e-01</internalNodes>
<leafValues>
  1.1186759471893311e+00 -5.2915501594543457e-01</leafValues></_>
<_>
<internalNodes>
  0 -1 22 1.4318999834358692e-02</internalNodes>
<leafValues>
  -3.5127198696136475e-01 1.1440860033035278e+00</leafValues></_>
<_>
<internalNodes>
  0 -1 23 1.0253000073134899e-02</internalNodes>
```

```

179      <_>
180      <internalNodes>
181      | 0 -1 23 1.0253000073134899e-02</internalNodes>
182      <leafValues>
183      | -6.0850602388381958e-01 7.7098500728607178e-01</leafValues></_>
184      <_>
185      <internalNodes>
186      | 0 -1 24 9.1508001089096069e-02</internalNodes>
187      <leafValues>
188      | 3.8817799091339111e-01 -1.5122940540313721e+00</leafValues></_></weakClassifiers></_>
189      <_>
190      <maxWeakCount>27</maxWeakCount>
191      <stageThreshold>-4.6551899909973145e+00</stageThreshold>
192      <weakClassifiers>
193      <_>
194      <internalNodes>
195      | 0 -1 25 6.9747000932693481e-02</internalNodes>
196      <leafValues>
197      | -1.0130879878997803e+00 1.4687349796295166e+00</leafValues></_>
198      <_>
199      <internalNodes>
200      | 0 -1 26 3.1502999365329742e-02</internalNodes>
201      <leafValues>
202      | -1.6463639736175537e+00 1.0000629425048828e+00</leafValues></_>
203      <_>
204      <internalNodes>
205      | 0 -1 27 1.4260999858379364e-02</internalNodes>
206      <leafValues>
207      | 4.6480301022529602e-01 -1.5959889888763428e+00</leafValues></_>
208      <_>
209      <internalNodes>
210      | 0 -1 28 1.4453000389039516e-02</internalNodes>
211      <leafValues>

```

Do you mind taking a quick feedback survey?

```

<features>
  <_>
    <rects>
      <_>
        | 6 4 12 9 -1.</_>
      <_>
        | 6 7 12 3 3.</_></rects></_>
    <_>
      <rects>
        <_>
          | 6 4 12 7 -1.</_>
        <_>
          | 10 4 4 7 3.</_></rects></_>
    <_>
      <rects>
        <_>
          | 3 9 18 9 -1.</_>
        <_>
          | 3 12 18 3 3.</_></rects></_>
    <_>
      <rects>
        <_>
          | 8 18 9 6 -1.</_>
        <_>
          | 8 20 9 2 3.</_></rects></_>
    <_>
      <rects>
        <_>
          | 3 5 4 19 -1.</_>
        <_>
          | 5 5 2 19 2.</_></rects></_>
  </_>

```

Do you mind taking a quick feedback survey?

Take Survey

Remind Me later

```

3 19 9 2 3.</_></rects></_>
<_>
  <rects>
    <_>
      16 2 3 20 -1.</_>
    <_>
      17 2 1 20 3.</_></rects></_>
  <_>
    <rects>
      <_>
        0 13 24 8 -1.</_>
      <_>
        0 17 24 4 2.</_></rects></_>
    <_>
      <rects>
        <_>
          9 1 6 22 -1.</_>
        <_>
          12 1 3 11 2.</_>
        <_>
          9 12 3 11 2.</_></rects></_></features></cascade>
</opencv_storage>

```

Mungkin ini hanya gambarannya saja karena sangat banyak isinya, jika mau ambil bisa cari di github.

Kemudian file hardcascade untuk mata

7 C:\Users\user\Desktop\intel\face\lib\addressbook\_eye.xml

```
<?xml version="1.0">
```

```
<!--
```

```
    Stump-based 20x20 frontal eye detector.
```

```
    Created by Shameem Hameed (http://umich.edu/~shameem)
```

```
////////////////////////////////////
```

```
IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.
```

```
By downloading, copying, installing or using the software you agree to this license.
```

```
If you do not agree to this license, do not download, install,
```

```
copy or use the software.
```

```

| | | | | | | | | | Intel License Agreement
| | | | | | | | | | For Open Source Computer Vision Library
```

```
Copyright (C) 2000, Intel Corporation, all rights reserved.
```

```
Third party copyrights are property of their respective owners.
```

```
Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
```

```

| * Redistribution's of source code must retain the above copyright notice,
|   this list of conditions and the following disclaimer.
```

```

| * Redistribution's in binary form must reproduce the above copyright notice,
|   this list of conditions and the following disclaimer in the documentation
|   and/or other materials provided with the distribution.
```

```

| * The name of Intel Corporation may not be used to endorse or promote products
|   derived from this software without specific prior written permission.
```

```

<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>20</height>
  <width>20</width>
  <stageParams>
    <maxWeakCount>93</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>24</stageNum>
  <stages>
    <_>
      <maxWeakCount>6</maxWeakCount>
      <stageThreshold>-1.4562760591506958e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 1.2963959574699402e-01</internalNodes>
          <leafValues>
            -7.7304208278656006e-01 6.8350148200988770e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 -4.6326808631420135e-02</internalNodes>
          <leafValues>
            5.7352751493453979e-01 -4.9097689986228943e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 -1.6173090785741806e-02</internalNodes>
          <leafValues>
            6.0254341363906860e-01 -3.1610709428787231e-01</leafValues></_>
      <_>
        <internalNodes>

```

```

<_>
<maxWeakCount>12</maxWeakCount>
<stageThreshold>-1.2550230026245117e+00</stageThreshold>
<weakClassifiers>
  <_>
    <internalNodes>
      0 -1 6 -2.1727620065212250e-01</internalNodes>
    <leafValues>
      7.1098899841308594e-01 -5.9360730648040771e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 7 1.2071969918906689e-02</internalNodes>
      <leafValues>
        -2.8240481019020081e-01 5.9013551473617554e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 8 -1.7854139208793640e-02</internalNodes>
      <leafValues>
        5.3137522935867310e-01 -2.2758960723876953e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 9 2.2333610802888870e-02</internalNodes>
      <leafValues>
        -1.7556099593639374e-01 6.3356137275695801e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 10 -9.1420017182826996e-02</internalNodes>
      <leafValues>
        6.1563092470169067e-01 -1.6899530589580536e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 11 2.8973650187253952e-02</internalNodes>

```

```

      1.1587540060281754e-01 -1.2790560722351074e-01</leafValues></_>
    <_>
      <internalNodes>
        0 -1 1064 2.4430730263702571e-05</internalNodes>
      <leafValues>
        -1.6816629469394684e-01 8.0449983477592468e-02</leafValues></_>
    <_>
      <internalNodes>
        0 -1 1065 -5.5345650762319565e-02</internalNodes>
      <leafValues>
        4.5338949561119080e-01 -3.1222779303789139e-02</leafValues></_></weakClassifiers></_></stages>
<features>
  <_>
    <rects>
      <_>
        0 8 20 12 -1.</_>
      <_>
        0 14 20 6 2.</_></rects></_>
    <_>
      <rects>
        <_>
          9 1 4 15 -1.</_>
        <_>
          9 6 4 5 3.</_></rects></_>
    <_>
      <rects>
        <_>
          6 10 9 2 -1.</_>
        <_>
          9 10 3 2 3.</_></rects></_>

```

```

    6 10 9 2 -1.</_>
  <_>
    9 10 3 2 3.</_></rects></_>
<_>
  <rects>
    <_>
      7 0 10 9 -1.</_>
    <_>
      7 3 10 3 3.</_></rects></_>
<_>
  <rects>
    <_>
      12 2 2 18 -1.</_>
    <_>
      12 8 2 6 3.</_></rects></_>
<_>
  <rects>
    <_>
      8 6 8 6 -1.</_>
    <_>
      8 9 8 3 2.</_></rects></_>
<_>
  <rects>
    <_>
      2 0 17 18 -1.</_>
    <_>
      2 6 17 6 3.</_></rects></_>
<_>
  <rects>
    <_>
      10 10 1 8 -1.</_>
    <_>
      10 14 1 4 2.</_></rects></_>

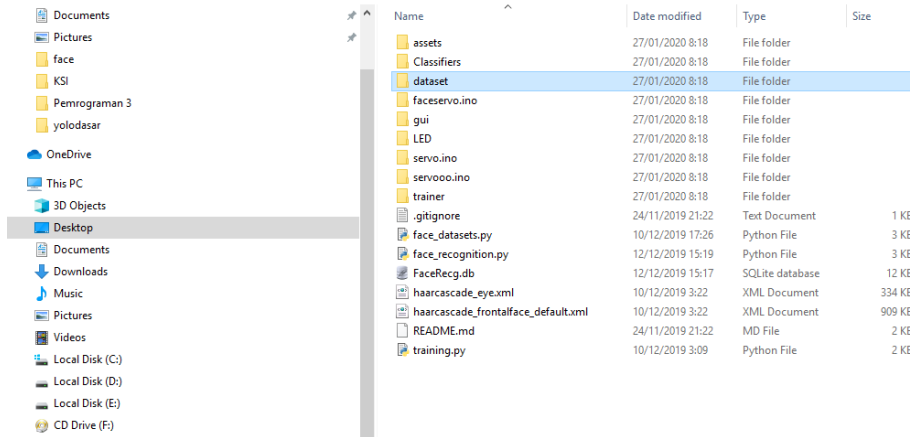
```

```

186 | 6 10 2 1 -1.</_>
187 | <_>
188 | 7 10 1 1 2.</_></rects></_>
189 | <_>
190 | <rects>
191 | <_>
192 | 1 1 18 16 -1.</_>
193 | <_>
194 | 10 1 9 16 2.</_></rects></_>
195 | <_>
196 | <rects>
197 | <_>
198 | 14 4 3 15 -1.</_>
199 | <_>
200 | 15 4 1 15 3.</_></rects></_>
201 | <_>
202 | <rects>
203 | <_>
204 | 19 13 1 2 -1.</_>
205 | <_>
206 | 19 14 1 1 2.</_></rects></_>
207 | <_>
208 | <rects>
209 | <_>
210 | 2 6 5 8 -1.</_>
211 | <_>
212 | 2 10 5 4 2.</_></rects></_></features></cascade>
213 | </opencv_storage>
214 |

```





Jika kalian telah mendownload kedua file tersebut yaitu haarcascade\_frontalface\_default.xml dan juga haarcascade\_eye.xml maka kalian telah siap untuk mengikuti Tutorial Face dan Eye Detector menggunakan OpenCV Python berikut.

## Program Face dan Eye Detector OpenCV Python pada Gambar

Nah, untuk gambar maupun video, kita masukkan terlebih dahulu kedua file yang telah kalian download tadi kedalam script Python kalian. Perlu diketahui disini saya sudah rename file saya menjadi face-detect.xml dan juga eye-detect.xml. Untuk mengimportnya kita gunakan CascadeClassifier() method. Berikut adalah code untuk importnya.

```
import cv2

img = cv2.imread('foto-kamu.jpg')

face = cv2.CascadeClassifier('face-detect.xml')
eye = cv2.CascadeClassifier('eye-detect.xml')
```

Maksud dari code diatas adalah kita memasukkan gambar kita dan juga file kita yang sebagai detector atau pendeteksi kedalam variabel. Hal ini untuk mempermudah lagi proses pemanggilan file ini nantinya. Selanjutnya kita akan terapkan codenya pada gambar kita.

Tapi sebelumnya kita harus ubah terlebih dahulu gambar awal kita menjadi hitam putih. Hal ini dilakukan karena sebuah foto dan juga video lebih mudah diproses menggunakan algoritma dalam keadaan hitam putih.

Untuk membuat gambar kita hitam putih kita gunakan `gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`. Code ini akan mengubah gambar kita menjadi hitam putih yang terletak dalam variabel `gray`. Selanjutnya kita buat Code untuk Face Detector nya dengan menggunakan file yang telah didownload tadi.

Untuk menggunakannya kita cukup tuliskan seperti ini `muka = face.detectMultiScale(gray, 1.3, 5)`. Nah code ini menerapkan algoritma dari face kedalam `gray`, yang merupakan versi hitam putih dari gambar kita. Kemudian 2 parameter terakhir berfungsi sebagai angka keakuratan dari algoritma kita. Sama seperti nilai keakuratan dari Edge Detector.

Kita dapat mengubahnya dan menyesuaikan dengan sesuka hati. Tetapi umumnya digunakan angka 1.3 dan juga 5. Nah, variabel `muka` tersebut akan menghasilkan titik X muka, titik Y muka, lebar muka dan juga tinggi muka.

Untuk menggambar sebuah kotak di muka yang terdeteksi maka kita gunakan `for loop` untuk setiap variabel dalam `muka`. Contoh codenya seperti ini.

```
for (x,y,w,h) in muka:
    cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
```

Nah setelah setiap variabel muka terambil kita akan gambar kotak pada wilayah muka tersebut. OpenCV memiliki banyak fitur Built-In untuk menggambar diatas gambar dan juga video. Tapi kali ini kita gunakan salah satu saja, yaitu `rectangle method` untuk menggambar kotak pada area wajah kita.

Terlihat pada code diatas kita deklarasikan 5 parameter. Parameter pertama adalah letak dimana kita akan menggambar yaitu pada `img` bukan `gray` karena kita gunakan `gray` untuk mendeteksi muka, tetapi kita gambar kedalam gambar yang berwarna atau gambar asli kita.

Kemudian parameter kedua yaitu tuple `(x, y)` yang merupakan titik kiri atas dari kotak persegi kita. Parameter ketiga yaitu tuple `(x+w, y+h)` untuk mendefinisikan titik kanan bawah persegi kita sehingga akan terbentuklah persegi dengan 2 titik tersebut.

Parameter keempat adalah tuple untuk warna. Warna yang dianut OpenCV adalah BGR, yaitu Blue Green Red, kebalikan dari RGB. Valuenya masih sama, warna terkuat adalah value 255, nah karena saya ingin warna hijau maka saya gunakan `(0, 255, 0)` Kemudian parameter kelima dan terakhir adalah tebal garis kita, disini saya gunakan angka 2 saja.

```
import cv2

img = cv2.imread('foto-kamu.jpg')

face = cv2.CascadeClassifier('face-detect.xml')
eye = cv2.CascadeClassifier('eye-detect.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

muka = face.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in muka:
    cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
```

Nah untuk sekarang kita beres untuk face detector, sebelum kita run kita akan membuat untuk eye detector terlebih dahulu. Untuk membuat eye detector kita akan lakukan didalam `for` loop yang telah kita buat diatas.

Hal pertama yang harus kita pahami konsepnya adalah jika mata merupakan komponen dari wajah. Jadi tidak mungkin ada mata diluar wajah. Jadi tidak mungkin mesin akan mendeteksi mata melayang diluar wajah. Tentunya akan menjadi hal yang tidak wajar dan justru menakutkan.

Oleh karena itu kita membutuhkan Region of Image atau yang biasa disingkat RoI. Kita akan gunakan RoI ini untuk menandai bahwa wilayah untuk eye detection hanyalah didalam daerah wajah yang telah terdeteksi sebelumnya.

Maka untuk itu kita buat code seperti ini

```
roi_warna = frame[y:y+h, x:x+w]  
roi_gray = gray[y:y+h, x:x+w]
```

RoI didefinisikan dengan bentuk mirip list pada Python. Didalamnya terdapat 2 parameter yang berupa range. Disini urutan untuk menandai wilayahnya adalah Y, X bukannya X, Y seperti standart pada umumnya. Variabel x, y, w, h yang tertulis diatas adalah milik muka yang sudah diambil menggunakan for loop.

Disini juga kita harus mendefinisikan RoI untuk versi berwarna dan juga versi hitam putih. Kita definisikan dalam 2 variabel berbeda yaitu roi\_warna untuk versi yang berwarna dan juga roi\_gray untuk versi hitam putihnya. Urusan isi parameternya sama saja.

Selanjutnya kita terapkan juga Eye Detector pada script kita. Kita cukup tuliskan seperti ini mata = eye.detectMultiScale(ro\_i\_gray) Masih sama prinsipnya kita terapkan algoritma kita kedalam versi hitam putihnya. Bedanya disini kita tidak perlu deskripsikan lagi nilai keakuratannya.

Hal ini karena kita inherit saja dari variabel muka diatas. Tetapi jika kalian menemukan ketidakakuratan deteksi, kalian bisa deskripsikan lagi nilainya. Disini percobaan run pertama ada sedikit miss dimana mulut juga terdeteksi sebagai mata. Oleh karena itu saya tambahkan codenya hingga seperti ini mata = eye.detectMultiScale(ro\_i\_gray, 1.5, 3) dan miss pun terselesaikan.

Kemudian kita masukkan lagi code untuk for loop. Code kali ini untuk menggambar kotak pada mata. Usahakan agar variabelnya berbeda dengan for loop yang pertama agar tidak terjadi error. Untuk sisanya dan urusan show masih sama saja seperti sebelumnya. Sehingga code kalian akan terlihat seperti ini

```
import cv2

img = cv2.imread('lat-cv.jpg')

face = cv2.CascadeClassifier('face-detect.xml')
eye = cv2.CascadeClassifier('eye-detect.xml')

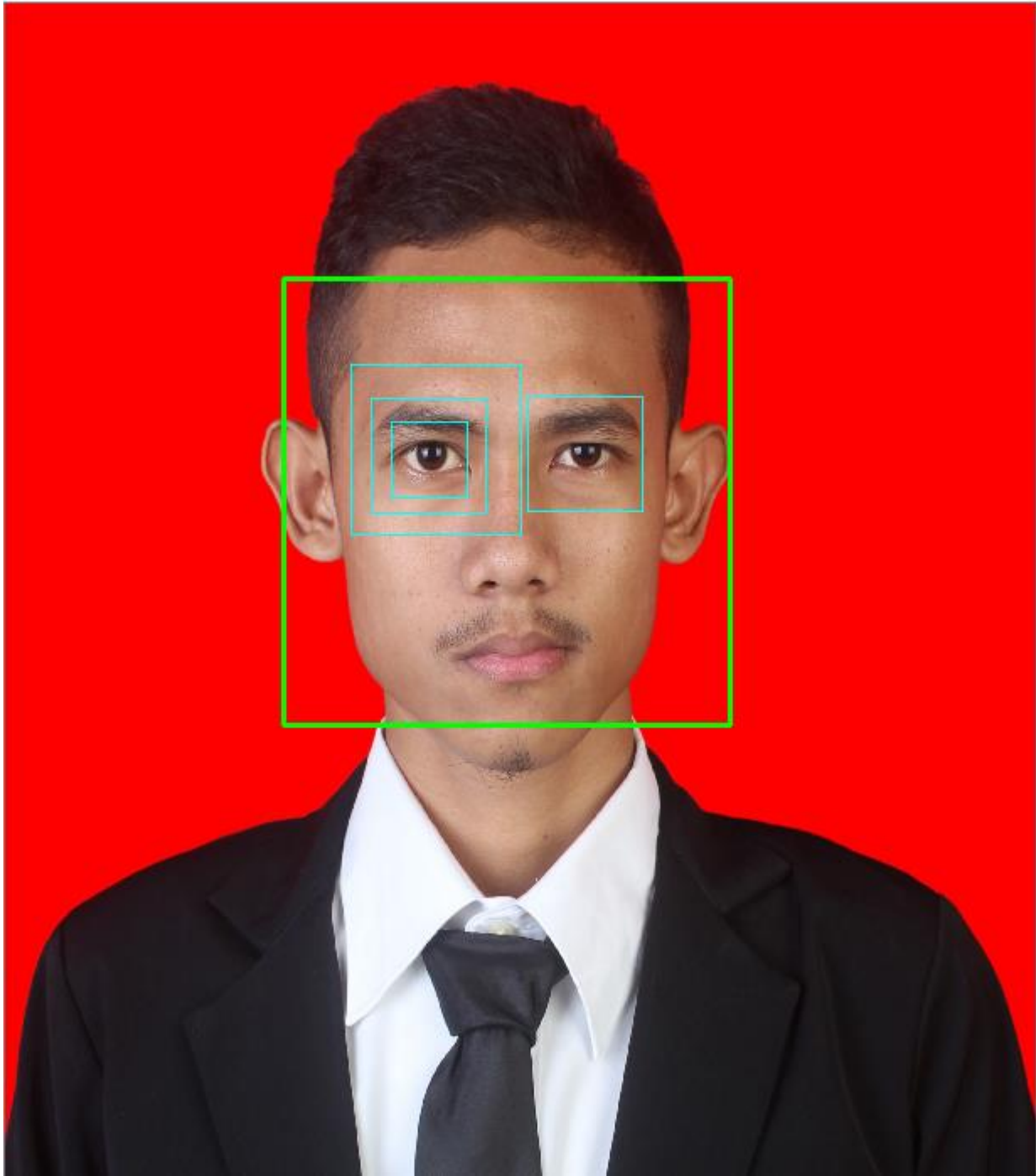
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

muka = face.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in muka:
    cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)

    roi_warna = img[y:y+h, x:x+w]
    roi_gray = gray[y:y+h, x:x+w]
    mata = eye.detectMultiScale(roi_gray, 1.5, 3)
    for (mx,my,mw,mh) in mata:
        cv2.rectangle(roi_warna, (mx,my), (mx+mw, my+mh), (255, 255, 0), 1)

cv2.imshow('Foto Normal', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dan silahkan kalian run code kalian dan hasilnya akan kurang lebih seperti iniDan silahkan kalian run code kalian dan hasilnya akan kurang lebih seperti ini



Jika code kalian belum tepat, silahkan saja otak - atik dibagian nilai keakuratan. Jika sudah akurat. Kalian dapat melanjutkan untuk menerapkan Face dan Eye Detector ini kedalam Live Video dari Kamera kalian.

## Program Face dan Eye Detector OpenCV Python pada Live Video

Misi terakhir kita sudah didepan mata. Kita hanyalah perlu memodifikasi sedikit dan dan hanyalah menambahkan beberapa baris code untuk for loop dalam code kita. Langsung saja, hanyalah seperti ini codenya.

```
import cv2

videoCam = cv2.VideoCapture(0)

face = cv2.CascadeClassifier('face-detect.xml')
eye = cv2.CascadeClassifier('eye-detect.xml')

while True:
    cond, frame = videoCam.read()

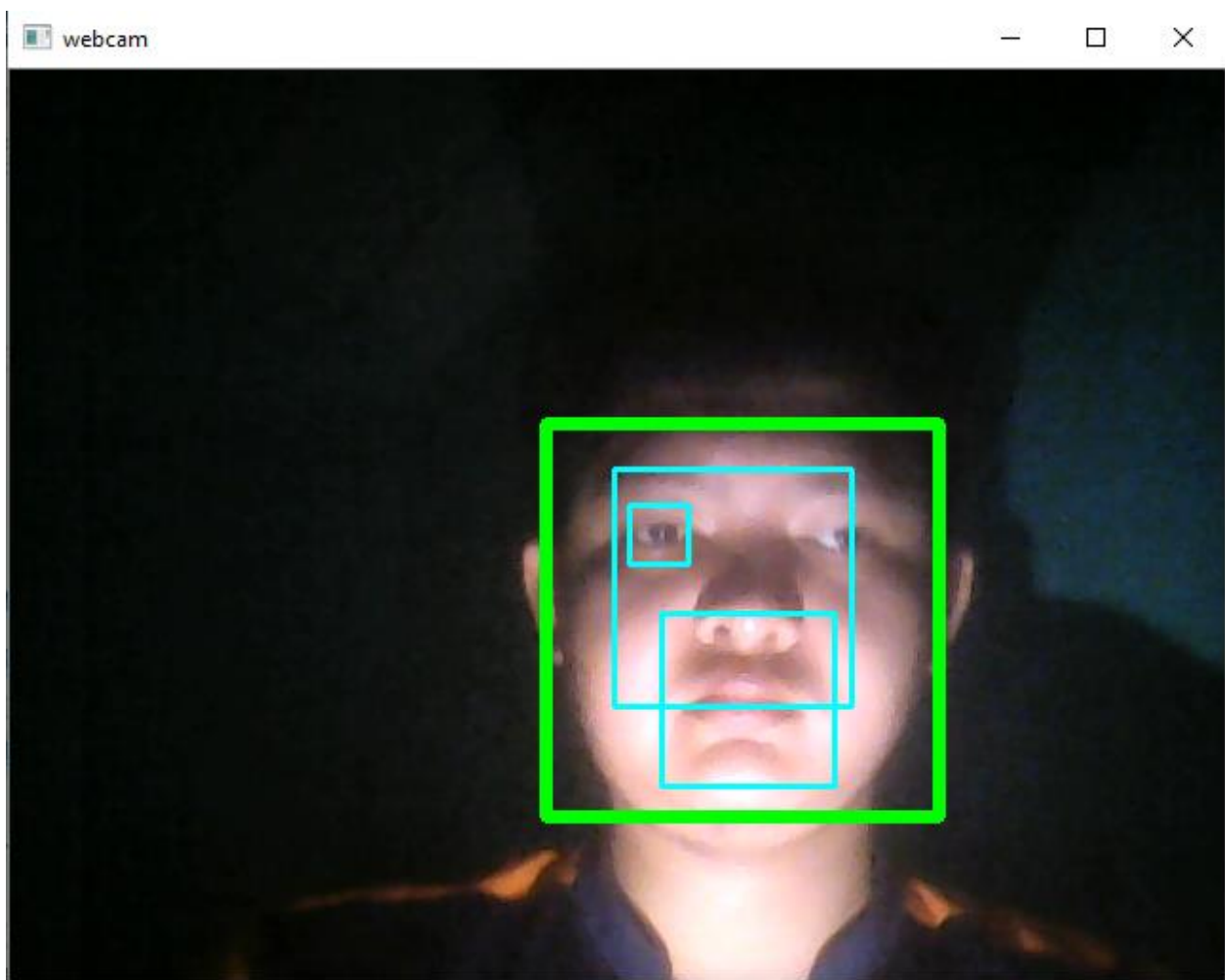
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    muka = face.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in muka:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 255, 0), 5)

        roi_warna = frame[y:y+h, x:x+w]
        roi_gray = gray[y:y+h, x:x+w]
        mata = eye.detectMultiScale(roi_gray)
        for (mx,my,mw,mh) in mata:
            cv2.rectangle(roi_warna, (mx,my), (mx+mw, my+mh), (255,255,0), 2)
```

```
cv2.imshow('Face dan Eye detection', frame)

k = cv2.waitKey(1) & 0xff
if k == ord('q'):
    break

videoCam.release()
cv2.destroyAllWindows()
```



Nah dengan basic skill yang sudah kita mainkan diatas kita bisa mengembangkannya untuk menjadi pengenalan wajah seperti milik Facebook dengan mengawinkannya dengan Machine Learning. Tapi untuk itu mungkin diperlukan langkah yang lebih advance lagi.



Cukup sekian Tutorial Membuat Program Face, Eye dan Edge Detector Menggunakan OpenCV Python.

## **Tutorial Membuat Program Face Recognizer Menggunakan OpenCV Python #1: Membuat Database**

Setelah post untuk OpenCV pertama yang saya memberikan tutorial membuat Face Detection. Nah dengan berbekal pengertian dan kemampuan dalam membuat Face Detection tersebut, kita dapat mengawinkannya dengan Machine Learning untuk membuat Face Recognition sendiri.

Face Recognition kita dapat kembangkan agar sedikit lebih advance daripada milik Facebook yang sering ditampilkan kepada kita saat mengupload foto kita bersama dengan geng kita. Kita dapat mengembangkannya agar dapat bekerja pada live video.

Untuk proses pembuatan Face Recognition ini saya akan bagi menjadi 3 part. Dan setiap partnya ada baiknya kalian juga buat script yang saya ajarkan pada 3 file yang berbeda. Ini digunakan agar saat proses pembuatan dan penambahan Database nantinya tidak bingung dan tidak perlu khawatir akan mengganggu display gambar nantinya.

Pada tutorial yang pertama ini kita akan membuat terlebih dahulu database atau data wajah kita yang nantinya dapat diberi label yang sesuai dengan nama kita untuk ditampilkan oleh mesin OpenCV featuring Python kita. Langsung saja kita mulai pembuatan script untuk databasenya.

### **Membuat Database untuk Face Recognizer OpenCV Python**

Nah rencananya saya ingin membuat database yang berisi gambar - gambar muka manusia kedalam folder terpisah, jadi silahkan kalian buat terlebih dahulu folder khusus untuk menampung database muka kalian. Pada tutorial saya ini saya membuat folder dengan nama dataWajah

Setelah kalian membuat folder khusus Database penampung data wajah, kita akan membuat Face detection terlebih dahulu. Datanya masih sama seperti post OpenCV sebelumnya. Kalian bisa lihat di Tutorial Face, Eye dan Edge Detection OpenCV Python.

Tapi tenang saya akan memberikan penjelasan kembali dalam post saya kali ini agar kalian semakin mengerti lebih dan lebih mengerti untuk setiap code dalam tutorial OpenCV saya.

Sebelumnya agar kalian nyaman untuk mengikuti tutorial ini kedepannya, saya rekomendasikan untuk menginstall terlebih dahulu versi advance dari OpenCV. Yaitu OpenCV yang ditambah dengan kontribusi, didalamnya ada tambahan Machine Learning untuk kita membuat Face Recognizer ini.

Cara menginstallnya juga sangat mudah, cukup gunakan PIP dengan menuliskan `pip install opencv-contrib-python` pada terminal kalian. Masih sama seperti OpenCV sebelumnya, mungkin masih ada beberapa keterbatasan untuk pengguna OS berbasis Unix seperti Linux dan MacOS.

Kalian bisa menguninstall terlebih dahulu versi OpenCV kalian sebelumnya, tapi menurut saya lebih baik dibiarkan saja seperti saya, karena siapa tahu juga jika di unisntall akan ada beberapa fungsi yang tidak bisa digunakan. Dan sekarang kita telah siap memulai Tutorial OpenCV kali ini.

Gampangnya algoritma yang saya pakai adalah seperti ini, pertama kita membuat Face Detection, kemudian kita tangkap gambar wajah kita dan simpan kedalam folder yang telah kita buat diatas.

## **Import Module Untuk Database Face Recognition**

Jangan lupa saya juga masih menggunakan HaarCascade yang sama pada tutorial sebelumnya untuk mencari wajah pada live video kita. Dan seperti biasa hal pertama yang kita lakukan adalah Importing semua hal yang kita butuhkan. Berikut adalah codenya.

```
import cv2  
face = cv2.CascadeClassifier('face-detect.xml')
```

Nah pasti kalian sudah familiar dengan code diatas, pertama kita import modul OpenCV kita. Tenang saja OpenCV biasa ataupun OpenCV + Contribution sama saja dengan mengimportnya dengan import cv2

Selanjutnya pada line kedua saya memasukkan file HaarCascadenya yang akan digunakan untuk mendeteksi wajah pada live video kita nantinya. Lanjut saya akan mendefinisikan parameter yang nantinya akan menjadi batasan gambar kita untuk di capture.

Selanjutnya kita akan membuat VideoCapture() method untuk mengambil video live kita. Tidak lupa juga input user untuk memasukkan Id wajah untuk melabeli gambar kita nantinya. Berikut adalah simpel codenya.

```
cam = cv2.VideoCapture(0)  
jumlah = 0  
id = input("Masukkan ID: ")
```

Karena saya hanya memiliki satu kamera, saya menggunakan parameter 0 pada VideoCapture() . Tetapi jika kalian memilih menggunakan kamera kedua kalian isikan parameter 1 dan begitu seterusnya. Jika sudah kita akan lanjut kedalam inti script ini, apalagi jika bukan Looping untuk menampilkan video dan capture foto.

## **Looping Video untuk Database Face Recognition OpenCV Python**

Simpel saja untuk code ini sebenarnya. Apalagi kalian telah mengikuti Tutorial Face, Eye dan Edge Detection OpenCV Python pada post saya sebelumnya. Hanya saja disini saya tambahkan sedikit untuk mengcapture video dan save dalam bentuk foto kedalam folder yang telah kamu buat sebelumnya. Codenya terlihat seperti ini.

```
while True:
    _, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    wajah = muka.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in wajah:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 5)
        if cv2.waitKey(1) & 0xff == ord('c'):
            cv2.imwrite('dataWajah/User.'+id+'.' + str(jumlah) + ".jpg",
            gray[y:y + h, x:x + w])
            jumlah += 1

    cv2.imshow('Train Face', frame)

    if jumlah > 20:
        break
```

Kali ini saya hanyalah akan menjelaskan bagian ini saja

```
if cv2.waitKey(1) & 0xff == ord('c'):
    cv2.imwrite('dataWajah/User.'+id+'.' + str(jumlah) + ".jpg", gray[y:y +
h, x:x + w])
    jumlah += 1

if jumlah > 20:
    break
```

Bagian diatas adalah kondisi agar ketika user menekan tombol c pada keyboard, maka akan mengcapture foto yang bersatu didalam imwrite() method. Method ini mengambil 2 parameter didalamnya.

Parameter pertama adalah nama file foto kita nantinya dan tak lupa ekstensi videonya. Disana saya memberikan nama dengan format seperti ini User. id . no seri (jumlah).jpg Nah jadi pada awal nama fotonya selalu bernama User, hal ini suka - suka aja sih, tetapi biar lebih sistematis saya pakaikan nama ini saja.

Setelah kata User, dilanjutkan tanda titik (dot) yang berfungsi sebagai pemisah nantinya. Setelah dot pertama tersebut akan diikuti id yang dimasukkan oleh User nantinya saat program ini di run.

Sedangkan setelah dot kedua adalah urutan gambarnya, disini kita memberikan batas 20 gambar untuk setiap label nantinya. Oleh karena itu disana tertera code jumlah += 1 untuk menambahkan value dari jumlah setiap satu foto tertangkap dan juga hal ini agar tidak ada foto yang tertimpa karena bernama sama, jadi dengan begini akan ada urutan dari foto ke satu sampai ke dua puluh.

Kemudian parameter kedua adalah bagian mana yang akan kita save. Karena kita hanya menginginkan bagian wajah saja dan itu adalah yang berformat hitam putih maka kita gunakan Region of Image pada variabel gray seperti ini `gray[y:y + h, x:x + w]`

Jadi dengan code tersebut, gambar yang tersimpan hanyalah bagian dari muka kita saja, dan dalam format Grayscale atau lebih dikenal hitam putih.

Selanjutnya adalah kondisi ketika foto sudah melampaui 20 maka Loop akan berhenti dan keluar. Jadi kita tidak perlu susah - susah menghitung ini sudah foto seberapa. Semuanya otomatis, kalian terima jadi saja.

Terakhir tambahkan penutup yaitu `destroyAllWindows()` method dan juga `release()` method untuk menghabisi task script kita agar tidak memakan banyak memori.

Nah sekarang kita akan coba jalankan codenya. Tapi sebelum itu pastikan code kalian terlihat skurang lebih seperti ini.

```

import cv2

face = cv2.CascadeClassifier('face-detect.xml')

cam = cv2.VideoCapture(0)
jumlah = 0
id = input("Masukkan ID: ")

while True:
    _, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    wajah = muka.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in wajah:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 5)
        if cv2.waitKey(1) & 0xff == ord('c'):
            cv2.imwrite('dataWajah/User.'+id+'.' + str(jumlah) + ".jpg",
            gray[y:y + h, x:x + w])
            jumlah += 1

    cv2.imshow('Train Face', frame)

```

```

            gray[y:y + h, x:x + w])
            jumlah += 1

    cv2.imshow('Train Face', frame)

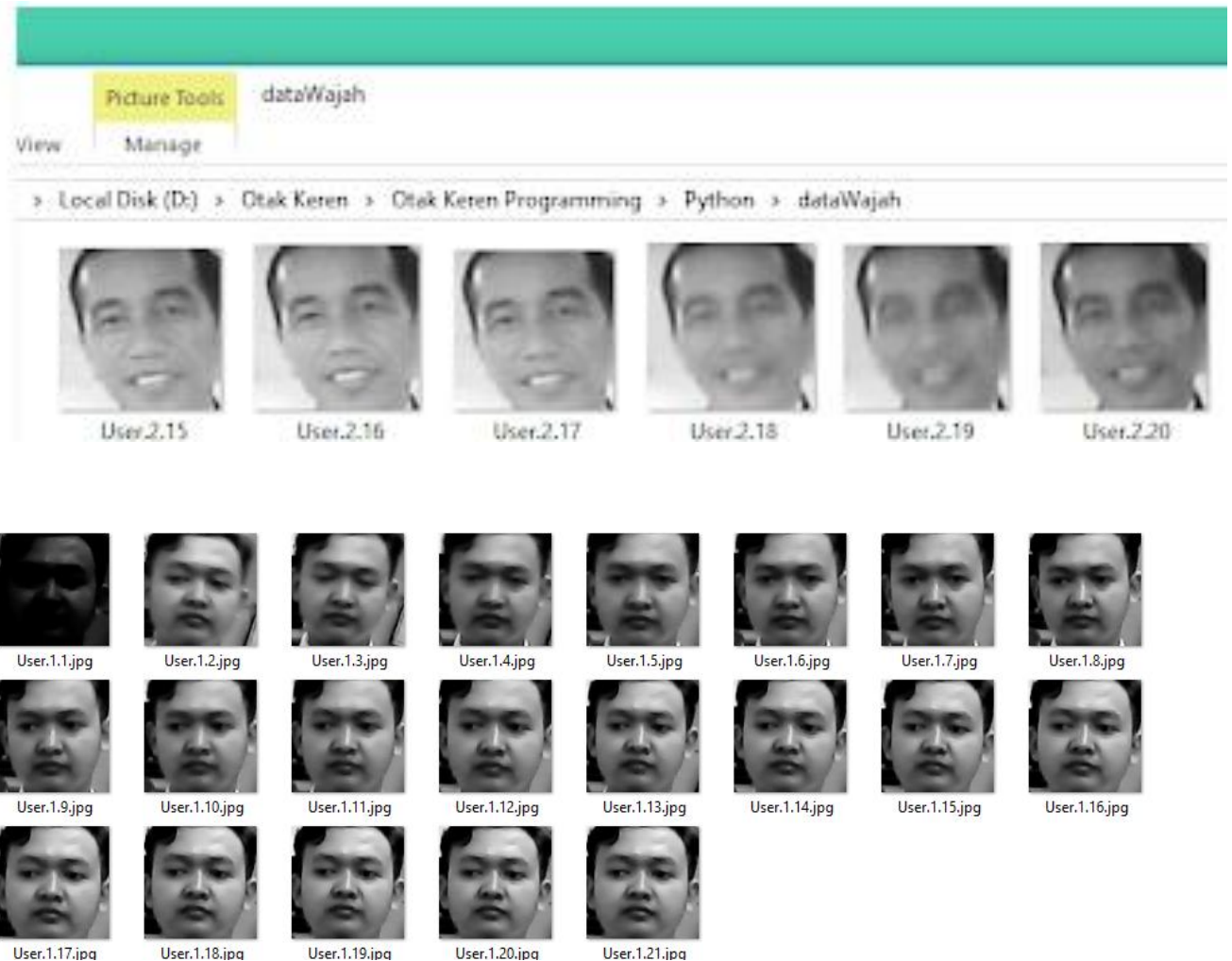
    if jumlah > 20:
        break

cv2.destroyAllWindows()
cam.release()

```

Jika sudah terlihat seperti itu maka kalian siap menjalankan code kalian. Jalankan kemudian tekan tombol c sebanyak 20 kali. Nantinya otomatis wajah kalain akan tercapture dan tersimpan

kedalam folder yang kalian buat sebelumnya, dan jika jumlah foto sudah mencapai angka 20 maka window akan tertutup otomatis. Hasilnya akan terlihat seperti ini.



## Tutorial Memanggil Database

Terkait tutorial sebelumnya yang sudah mendeteksi wajah dan juga menyimpannya, kali ini saya akan menjelaskan bagaimana cara pemanggilan database tersebut sehingga alat bisa mengenali wajah yang terdeteksi di kamera

```

1
2 import cv2
3 import numpy as np
4 import os
5 import pickle
6 import sqlite3
7 import serial
8 import time
9
10 Myserial = serial.Serial('COM5',9600, timeout = 1)
11 time.sleep(2)
12
13 def assure_path_exists(path):
14     dir = os.path.dirname(path)
15     if not os.path.exists(dir):
16         os.makedirs(dir)
17
18 def profile(Id):
19     conn=sqlite3.connect("FaceRecg.db")
20     cmd="SELECT * FROM Data WHERE Id="+str(Id)
21     cursor=conn.execute(cmd)
22     data=None
23     for rows in cursor:
24         data=rows
25     conn.close()
26     return data
27
28 # Create Local Binary Patterns Histograms for face recognition
29 recognizer = cv2.face.LBPHFaceRecognizer_create()
30
31 assure_path_exists("trainer/")
32
33 # Load the trained mode

```



```
# Create Local Binary Patterns Histograms for face recognition
recognizer = cv2.face.LBPHFaceRecognizer_create()

assure_path_exists("trainer/")

# Load the trained mode
recognizer.read('trainer/trainer.yml')

# Load prebuilt model for Frontal Face
cascadePath = "haarcascade_frontalface_default.xml"

# Create classifier from prebuilt model
faceCascade = cv2.CascadeClassifier(cascadePath);

# Set the font style
font = cv2.FONT_HERSHEY_SIMPLEX

# Initialize and start the video frame capture
cam = cv2.VideoCapture(0)

# Loop
while True:
    # Read the video frame
    ret, im =cam.read()

    # Convert the captured frame into grayscale
    gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

    # Get all face from the video frame
    faces = faceCascade.detectMultiScale(gray, 1.2,5)
```

```

gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

# Get all face from the video frame
faces = faceCascade.detectMultiScale(gray, 1.2,5)

# For each face in faces
for(x,y,w,h) in faces:

    # Create rectangle around the face
    cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)

    # Recognize the face belongs to which ID
    Id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

    Name=profile(Id)
    if(Name!=None):
        cv2.putText(im, str(Name[1]) , (x,y-40), font, 1, (255,255,255), 3)

# Display the video frame with the bounded rectangle
cv2.imshow('im',im)

if (len(faces)) == 1:
    if(Id == 1):
        Myserial.write(b'Y')
        print ("Face is detected")
elif (len(faces)) == 0:
    Myserial.write(b'N')
    print ("No face is detected")

k = cv2.waitKey(1) & 0xFF == ord('q')
if k == 1:
    break

```

```

    k = cv2.waitKey(1) & 0xFF == ord('q')
    if k == 1:
        break

# Stop the camera
cam.release()

# Close all windows
cv2.destroyAllWindows()

```

Jadi pertama-tama program akan dijalankan, kemudian meminta id dan nama dari objek, setelah diisi data akan disave di database pada bagian train, ketika data sudah di save maka apabila kamera menemukan wajah tersebut maka akan dibaca sebagai objek yang bersangkutan.

# FACE RECOGNITION WITH PYTHON

## 1. Perkenalan

Dalam dokumen ini saya akan menunjukkan kepada Anda bagaimana menerapkan Eigenfaces [13] dan Fisherfaces [3] metode dengan

[Python](#), jadi Anda akan memahami dasar-dasar Pengenalan Wajah. Semua konsep dijelaskan secara rinci, tetapi a

pengetahuan dasar tentang [Python](#) diasumsikan. Awalnya dokumen ini adalah Panduan untuk Mengenal Wajah

OpenCV. Sejak [OpenCV](#) sekarang hadir dengan [cv :: FaceRecognizer](#), dokumen ini sudah dikerjakan ulang

ke dalam dokumentasi OpenCV resmi di:

- <http://docs.opencv.org/trunk/modules/contrib/doc/facerec/index.html>

Saya melakukan semua ini di waktu luang saya dan saya tidak bisa menyimpan dua dokumen terpisah di Internet

topik yang sama lagi. Jadi saya telah memutuskan untuk mengubah dokumen ini menjadi panduan tentang Pengenalan Wajah dengan python.

Ngomong – ngomong Anda tidak perlu menyalin dan menempelkan cuplikan kode, semua kode telah dimasukkan ke saya

repositori github:

- [github.com/bytefish](https://github.com/bytefish)
- [github.com/bytefish/facerecognition\\_guide](https://github.com/bytefish/facerecognition_guide)

Semua yang ada di sini dirilis di bawah [lisensi BSD](#), jadi silakan menggunakannya untuk proyek Anda. Kamu

Saat ini sedang membaca versi [Python](#) dari Panduan Pengenalan Wajah, Anda dapat mengkompilasi [GNU](#)

[Oktaf](#)/ Versi [MATLAB](#) dengan make oktaf .

1

---

## 2 Pengenalan Wajah

Pengenalan wajah adalah tugas yang mudah bagi manusia. Eksperimen dalam [6] telah menunjukkan, bahwa satu hingga tiga

bayi usia sehari dapat membedakan antara wajah yang dikenal. Jadi seberapa sulitkah itu untuk komputer?

Ternyata kita tahu sedikit tentang pengakuan manusia hingga saat ini. Apakah fitur dalam (mata, hidung, mulut)

atau fitur luar (bentuk kepala, garis rambut) yang digunakan untuk pengenalan wajah yang sukses? Bagaimana kita menganalisis suatu

gambar dan bagaimana otak menyandikannya? Itu ditunjukkan oleh [David Hubel](#) dan [Torsten Wiesel](#), itu milik kita

otak memiliki sel-sel saraf khusus yang menanggapi fitur-fitur 53erak tertentu dari suatu pemandangan, seperti garis, tepi, sudut atau 53erakan. Karena kita tidak melihat dunia sebagai bagian yang tersebar, korteks visual kita harus entah bagaimana menggabungkan berbagai sumber informasi ke dalam pola yang bermanfaat. Pengenalan wajah otomatis adalah segalanya tentang mengekstraksi fitur-fitur yang berarti dari suatu gambar, menempatkannya ke dalam representasi yang bermanfaat dan melakukan semacam klasifikasi pada mereka.

Pengenalan wajah berdasarkan fitur geometris wajah mungkin merupakan pendekatan yang paling intuitif pengenalan wajah. Salah satu sistem pengenalan wajah otomatis pertama dijelaskan dalam [9]: marker titik (posisi mata, telinga, hidung, ...) digunakan untuk membangun 53eraka fitur (jarak antara

poin, sudut di antara mereka, ...). Pengakuan itu dilakukan dengan menghitung jarak 53erakan5353

antara 53eraka fitur probe dan gambar referensi. Metode seperti itu kuat terhadap perubahan dalam

pencahayaan berdasarkan sifatnya, tetapi memiliki kelemahan besar: pendaftaran poin penanda yang akurat

rumit, bahkan dengan algoritma canggih. Beberapa karya terbaru tentang wajah geometris

Pengakuan dilakukan di [4] Vektor fitur 22 dimensi digunakan dan percobaan dilakukan

dataset besar telah menunjukkan, bahwa fitur geometris saja tidak membawa informasi yang cukup untuk wajah

pengakuan.

Metode Eigenfaces dijelaskan dalam [13] mengambil pendekatan 54erakan54

untuk pengenalan wajah: Gambar wajah adalah

titik dari ruang gambar dimensi tinggi dan representasi dimensi rendah ditemukan, di mana

klasifikasi menjadi mudah. Subruang dimensi bawah ditemukan dengan

Komponen Utama

Analisis, yang mengidentifikasi sumbu dengan varians maksimum. Sementara transformasi seperti ini

optimal dari sudut pandang rekonstruksi, tidak memperhitungkan label

kelas. Bayangkan a

situasi di mana varians dihasilkan dari sumber eksternal, biarkan terang. Sumbu dengan maksimum

varians tidak harus mengandung informasi diskriminatif sama sekali, maka klasifikasi menjadi

mustahil. Jadi proyeksi khusus kelas dengan Analisis Diskriminan Linier diterapkan untuk dihadapi

pengakuan dalam [3] Ide dasarnya adalah untuk meminimalkan varians dalam suatu kelas, sekaligus memaksimalkan

varians antara kelas-kelas pada saat yang sama (Gambar 1).

Baru-baru ini berbagai metode untuk ekstraksi fitur 54erak muncul. Untuk menghindari dimensi tinggi

data input hanya daerah 54erak dari suatu gambar yang dijelaskan, fitur yang diekstraksi (semoga) lebih

kuat terhadap oklusi parsial, iluminasi, dan ukuran sampel kecil. Algoritma yang digunakan untuk fitur 54erak

ekstraksi adalah Gabor Wavelets ([14]), Discrete Cosinus Transform ([5]) dan Pola Biner Lokal ([1, 11, 12]). Ini masih merupakan pertanyaan penelitian terbuka bagaimana cara melestarikan informasi spasial ketika menerapkan ekstraksi fitur 55erak, karena informasi spasial adalah informasi yang berpotensi berguna.

## 2.1 Database Wajah

Saya tidak ingin membuat contoh mainan di sini. Kami sedang melakukan pengenalan wajah, jadi Anda perlu wajah gambar-gambar! Anda juga dapat membuat database Anda sendiri atau mulai dengan salah satu database yang tersedia, [wajah-rec.org/databases](http://wajah-rec.org/databases) memberikan ikhtisar terkini. Tiga database menarik adalah 1 : [AT&T Facedatabase](http://AT&T Facedatabase) AT&T Facedatabase, terkadang juga dikenal sebagai ORL Database of Faces, berisi sepuluh gambar berbeda dari masing-masing 40 subjek berbeda. Untuk beberapa subjek, gambar itu diambil pada waktu yang berbeda, memvariasikan pencahayaan, ekspresi wajah (mata terbuka / tertutup, tersenyum / tidak tersenyum) dan detail wajah (kacamata / tanpa kacamata). Semua gambar diambil melawan kegelapan latar belakang homogen dengan subjek dalam posisi tegak lurus (dengan toleransi untuk beberapa 55erakan samping).

[Yale Facedatabase](http://Yale Facedatabase) AT&T Facedatabase bagus untuk tes awal, tetapi ini cukup mudah basis data. Metode Eigenfaces sudah memiliki tingkat pengenalan 97%, jadi Anda tidak akan melihatnya

perbaikan dengan algoritma lainnya. Database Yale Facedatabase A adalah dataset yang lebih tepat

untuk percobaan awal, karena masalah pengenalan lebih sulit. Basis data terdiri dari

15 orang (14 pria, 1 wanita) masing-masing dengan 11 gambar skala abu-abu berukuran  $320 \times 243$  piksel. Ada

1 Bagian dari deskripsi dikutip dari [face-rec.org](http://face-rec.org).

2

---

perubahan kondisi cahaya (cahaya tengah, cahaya kiri, cahaya kanan), ekspresi wajah (bahagia,

normal, sedih, mengantuk, terkejut, mengedipkan mata) dan kacamata (kacamata, tanpa kacamata).

Gambar asli tidak dipotong atau disejajarkan. Saya sudah menyiapkan skrip Python yang tersedia di

src / py / crop\_face.py , yang melakukan pekerjaan untuk Anda.

[Extended Yale Facedatabase B](#) Extended Yale Facedatabase B berisi 2414 gambar dari 38

orang yang berbeda dalam versi yang dipangkas. Fokusnya adalah mengekstraksi fitur yang kuat

iluminasi, gambar hampir tidak memiliki variasi emosi / oklusi / .... Saya pribadi berpikir,

bahwa dataset ini terlalu besar untuk percobaan yang saya lakukan dalam dokumen ini, sebaiknya Anda gunakan

yang [AT & T Facedatabase](#). Versi pertama dari Yale Facedatabase B digunakan dalam [ 3] untuk melihat caranya



metode Eigenfaces dan Fisherfaces (bagian [2.3](#)) tampil di bawah perubahan iluminasi berat.

[10] menggunakan pengaturan yang sama untuk mengambil 16128 gambar dari 28 orang. Database Yale yang Diperluas

B adalah gabungan dari dua basis data, yang sekarang dikenal sebagai Extended Yalefacedatabase B.

Gambar wajah perlu disimpan dalam hierarki folder yang mirip dengan <database name> / <nama subjek> / <nama file

>. <ext> . The [AT & T Facedatabase](#) misalnya datang dalam hirarki tersebut, lihat properti [1](#).

Listing 1:

```
philipp @ mango: ~ / facerec / data / at $ tree
```

```
.
| - README
| - s1
|
| - 1.pgm
|
| - ...
|
| - 10.pgm
| - s2
|
| - 1.pgm
|
| - ...
|
```

| - 10.pgm

...

| - s40

|

| - 1.pgm

|

| - ...

|

| - 10.pgm

### 2.1.1 Membaca gambar dengan Python

Fungsi pada Listing [2](#) dapat digunakan untuk membaca gambar untuk setiap subfolder dari direktori yang diberikan.

Setiap direktori diberi label unik (integer), Anda mungkin ingin menyimpan nama folder juga.

Fungsi mengembalikan gambar dan kelas yang sesuai. Fungsi ini sangat mendasar dan

ada banyak hal yang perlu ditingkatkan, tetapi ia melakukan tugasnya.

Listing 2: [src / py / tinyfacerec / util.py](#)

```
def read_images (path, sz = Tidak Ada):
```

```
    c = 0
```

```
    X, y = [], []
```

```
    untuk dirname, dirnames, nama file di os.walk (jalur):
```

```
        untuk subdirname dalam nama-nama:
```

```
            subject_path = os.path.join (dirname, subdirname)
```

```
            untuk nama file di os.listdir (subject_path):
```

```
                coba :
```

```
                    im = Image.open (os.path.join (subject_path, nama file))
```

```

im = im.convert ( "L" )
# ubah ukuran ke ukuran yang diberikan (jika diberikan)
jika (sz tidak ada):
im = im.resize (sz, Image.ANTIALIAS)
X.append (np.asarray (im, dtype = np.uint8))
y.menambahkan (c)
kecuali IOError:
cetak "kesalahan I / O ({0}): {1}" .format (errno, strerror)
kecuali :
cetak "Kesalahan tak terduga:" , sys.exc_info () [0]
menaikkan
c = c +1
return [X, y]
3

```

---

## 2.2 Eigenfaces

Masalah dengan representasi gambar yang diberikan kepada kita adalah dimensinya yang tinggi. Dua dimensi gambar skala grayscale  $p \times q$  adalah ruang vektor  $pq$ -dimensi, jadi gambar dengan  $100 \times 100$  piksel terletak di ruang gambar 10,000-dimensi sudah. Itu terlalu banyak untuk perhitungan apa pun, tetapi memang demikian semua dimensi benar-benar bermanfaat bagi kita? Kami hanya dapat membuat keputusan jika ada perbedaan dalam data, jadi apa yang kita cari adalah komponen yang bertanggung jawab atas sebagian besar informasi. Kepala sekolah

Analisis Komponen (PCA) secara independen diusulkan oleh [Karl Pearson](#) (1901) dan [Harold Hotelling](#)

(1933) untuk mengubah satu set variabel yang mungkin berkorelasi menjadi sejumlah kecil variabel tidak berkorelasi. Itu

idenya adalah bahwa dataset dimensi tinggi sering dijelaskan oleh variabel berkorelasi dan oleh karena itu hanya a

beberapa dimensi yang bermakna mencakup sebagian besar informasi. Metode PCA menemukan arah

dengan varians terbesar dalam data, yang disebut komponen utama.

### 2.2.1 Deskripsi Algoritma

Misalkan  $X = \{x_1, x_2, \dots, x_n\}$  menjadi vektor acak dengan pengamatan  $x_i \in \mathbb{R}^d$ .

1. Hitung rata-rata  $\mu$

$$\mu =$$

$$\frac{1}{n}$$

$$\sum_{i=1}^n$$

$$x_i$$

$$\sum_{i=1}^n$$

$$x_i$$

$$x_i$$

$$(1)$$

2. Hitunglah Covariance Matrix  $S$

$$S =$$

$$\frac{1}{n}$$

$$\sum_{i=1}^n$$

$$(x_i - \mu)(x_i - \mu)^T$$

$$\sum_{i=1}^n$$

$$i = 1$$

$$(x_i - \mu)(x_i - \mu)^T$$

(2)

3. Hitung nilai eigen  $\lambda_i$  dan vektor eigen  $v_i$  dari  $S$

$$Sv_i = \lambda_i v_i, i = 1, 2, \dots, n$$

(3)

4. Pesan vektor eigen turun dengan nilai eigennya. Komponen utama  $k$  adalah vektor eigen yang sesuai dengan nilai eigen  $k$  terbesar.

Komponen utama  $k$  dari vektor yang diamati  $x$  kemudian diberikan oleh:

$$y = W^T (x - \mu)$$

(4)

di mana  $W = (v_1, v_2, \dots, v_k)$ . Rekonstruksi dari PCA diberikan oleh:

$$x = Wy + \mu$$

(5)

Metode Eigenfaces kemudian melakukan pengenalan wajah dengan:

1. Memproyeksikan semua sampel pelatihan ke dalam subruang PCA

(menggunakan Persamaan 4).

2. Memproyeksikan gambar permintaan ke dalam subruang PCA (menggunakan

Listing 5).

3. Menemukan tetangga terdekat antara gambar pelatihan yang diproyeksikan dan permintaan yang diproyeksikan

gambar.

Masih ada satu masalah yang tersisa untuk diselesaikan. Bayangkan kita diberikan 400 gambar berukuran  $100 \times 100$  piksel. Itu

Analisis Komponen Utama memecahkan matriks kovarian  $S = XX^T$ , di mana ukuran  $(X) = 10.000 \times 400$

dalam contoh kita. Anda akan berakhir dengan matriks  $10.000 \times 10.000$ , kira-kira 0,8GB. Memecahkan masalah ini

tidak layak, jadi kita harus menerapkan trik. Dari pelajaran aljabar linier Anda, Anda tahu bahwa  $M \times N$

matriks dengan  $M > N$  hanya dapat memiliki  $N - 1$  nilai eigen bukan nol. Jadi mungkin untuk mengambil nilai eigen

dekomposisi  $S = X^T X$  ukuran  $N \times N$  sebagai gantinya:

$$X^T X v_i = \lambda_i v_i$$

(6)

dan dapatkan vektor eigen asli  $S = X X^T$  dengan perkalian kiri dari matriks data:

$$X X^T (X v_i) = \lambda_i (X v_i)$$

(7)

Vektor eigen yang dihasilkan adalah ortogonal, untuk mendapatkan vektor eigen ortonormal mereka perlu dinormalisasi

untuk satuan panjang. Saya tidak ingin mengubahnya menjadi publikasi, jadi silakan lihat [\[7\]](#) untuk derivasi

dan bukti persamaan.

4

---

### 2.2.2 Eigenfaces dalam Python

Kita telah melihat, bahwa metode Eigenfaces dan Fisherfaces mengharuskan matriks data dengan pengamatan

demi baris (atau kolom jika Anda suka). Listing [3](#) mendefinisikan dua fungsi untuk membentuk kembali daftar multi-dimensional

data menjadi matriks data. Perhatikan, bahwa semua sampel diasumsikan memiliki ukuran yang sama.

Listing 3: [src / py / tinyfacerec / util.py](#)

```
def asRowMatrix (X):  
    jika len (X) == 0:  
        return np.array ([])  
    mat = np.empty ((0, X [0] .ukuran), dtype = X [0] .dtype)  
    untuk baris dalam X:  
        mat = np.vstack ((mat, np.asarray (baris) .reshape (1, -1)))  
    tika kembali  
def asColumnMatrix (X):  
    jika len (X) == 0:  
        return np.array ([])  
    mat = np.empty ((X [0] .ukuran, 0), dtype = X [0] .dtype)  
    untuk col dalam X:  
        mat = np.hstack ((mat, np.asarray (col) .reshape (-1,1)))  
    tika kembali
```

Menerjemahkan PCA dari deskripsi algoritmik dari bagian [2.2.1](#) ke [Python](#) hampir sepele.

Jangan menyalin dan menempel dari dokumen ini, kode sumber tersedia di folder `src / py / tinyfacerec`

. Listing [4](#) mengimplementasikan Analisis Komponen Utama yang diberikan oleh Persamaan [1](#), [2](#) dan [3](#). Juga

mengimplementasikan formulasi PCA produk dalam, yang terjadi jika ada lebih banyak dimensi daripada

sampel. Anda dapat mempersingkat kode ini, saya hanya ingin menunjukkan cara kerjanya.

Listing 4: [src / py / tinyfacerec / subspace.py](#)

```
def pca (X, y, num_components = 0):  
    [n, d] = X.bentuk  
    jika (num_komponen <= 0) atau (num_komponen > n):  
        num_components = n  
        mu = X. berarti (sumbu = 0)  
        X = X - mu  
    jika n > d:  
        C = np.dot (XT, X)  
        [nilai eigen, vektor eigen] = np.linalg.eigh (C)  
    lain :  
        C = np.dot (X, XT)  
        [nilai eigen, vektor eigen] = np.linalg.eigh (C)  
        vektor eigen = np.dot (XT, vektor eigen)  
    untuk saya di xrange (n):  
        vektor eigen[:, i] = vektor eigen[:, i] / np.linalg.norm (vektor eigen[:, i])  
    # atau cukup lakukan dekomposisi ukuran ekonomi  
    # vektor eigen, nilai eigen, varians = np.linalg.svd (XT, full_matrices = Salah)  
    # Urutkan vektor eigen turun berdasarkan nilai eigennya  
    idx = np.argsort (-eigenvalues)  
    nilai eigen = nilai eigen [idx]  
    vektor eigen = vektor eigen[:, idx]  
    # pilih hanya num_components  
    nilai eigen = nilai eigen [0: num_components] .copy ()  
    vektor eigen = vektor eigen[:, 0: num_components] .copy ()  
    return [nilai eigen, vektor eigen, mu]
```



Pengamatan diberikan oleh baris, sehingga proyeksi dalam Persamaan [4](#) perlu disusun ulang sedikit:

Listing 5: [src / py / tinyfacerec / subspace.py](#)

proyek `def` (W, X, mu = Tidak Ada):

`jika` mu `adalah` None:

`return` np.dot (X, W)

`return` np.dot (X - mu, W)

Hal yang sama berlaku untuk rekonstruksi dalam Persamaan [5](#):

Listing 6: [src / py / tinyfacerec / subspace.py](#)

5

---

`def` merekonstruksi (W, Y, mu = Tidak Ada):

`jika` mu `adalah` None:

`return` np.dot (Y, WT)

`return` np.dot (Y, WT) + mu

Sekarang semuanya sudah ditentukan, inilah saatnya untuk hal-hal yang menyenangkan. Gambar wajah dibaca dengan Listing [2](#) dan kemudian PCA penuh (lihat Listing [4](#)) dilakukan. Saya akan menggunakan pustaka [matplotlib yang](#) bagus untuk merencanakannya [Python](#), silakan instal jika Anda belum melakukannya.

Listing 7: [src / py / script / contoh\\_pca.py](#)

`import` sys

`# tambahkan tinyfacerec ke jalur pencarian modul`

`sys.path.append ( ".." )`

`# import colormaps numpy dan matplotlib`

`import` numpy sebagai np

`# import modul tinyfacerec`

```

dari tinyfacerec.subspace import pca
dari tinyfacerec.util impor dinormalkan, asRowMatrix, read_images
dari tinyfacerec.visual import subplot
# baca gambar
[X, y] = read_images ( "/" home / philipp / facerec / data / at" )
# melakukan pca penuh

```

```
[D, W, mu] = pca (asRowMatrix (X), y)
```

Itu sudah. Cukup mudah, bukan? Setiap komponen utama memiliki panjang yang sama seperti aslinya

gambar, sehingga dapat ditampilkan sebagai gambar. [13] disebut wajah-wajah yang tampak seperti hantu sebagai Eigenfaces,

dari situlah metode Eigenfaces mendapatkan namanya. Kami sekarang ingin melihat Eigenfaces,

tapi pertama-tama kita membutuhkan metode untuk mengubah data menjadi representasi [matplotlib](#) mengerti.

Vektor eigen yang kami hitung dapat berisi nilai negatif, tetapi data gambar dikecualikan sebagai

nilai integer unsigned dalam kisaran 0 hingga 255. Jadi kita perlu fungsi untuk menormalkan data terlebih dahulu

(Listing 8):

Listing 8: [src / py / tinyfacerec / util.py](#)

```
def normalize (X, low, high, dtype = Tidak Ada):
```

```
X = np.asarray (X)
```

```
minX, maxX = np.min (X), np.max (X)
```

```
# normalisasikan ke [0 ... 1].
```

```
X = X - float (minX)
```

```
X = X / float ((maksX - minX))
```

skala ke [rendah ... tinggi].

```
X = X * (tinggi-rendah)
```

```
X = X + rendah
```

jika tipe tidak ada:

```
return np.asarray (X)
```

```
return np.asarray (X, dtype = dtype)
```

Dengan Python kita akan mendefinisikan metode `subplot` (lihat [src / py / tinyfacerec / visual.py](#)) untuk menyederhanakan

merencanakan. Metode ini mengambil daftar gambar, judul, skala warna dan akhirnya menghasilkan subplot:

Listing 9: [src / py / tinyfacerec / visual.py](#)

```
import numpy sebagai np
```

```
import matplotlib.pyplot sebagai plt
```

```
import matplotlib.cm sebagai cm
```

```
def create_font (fontname = 'Tahoma' , fontsize = 10):
```

```
return { 'fontname' : fontname, 'fontsize' : fontsize }
```

```
def subplot (judul, gambar, baris, cols, sptitle = "subplot" , sptitles = [], colormap =  
cm.
```

```
abu-abu, ticks_visible = Benar, nama file = Tidak Ada):
```

```
fig = plt.figure ()
```

```
# judul utama
```

```
fig.text (.5, .95, title, horizontalalignment = 'center' )
```

```
untuk saya di xrange (len (gambar)):
```

```
ax0 = fig.add_subplot (baris, cols, (i + 1))
```

```
plt.setp (ax0.get_xticklabels (), terlihat = Salah)
```

```
plt.setp (ax0.get_yticklabels (), terlihat = Salah)
```

---

if len (sptitles) == len (gambar):

plt.title ( "% s # % s " % (sptitle, str (sptitles [i])), create\_font ( 'Tahoma' , 10))

lain :

plt.title ( "% s # % d " % (sptitle, (i +1)), create\_font ( 'Tahoma' , 10))

plt.imshow (np.asarray (gambar [i]), cmap = colormap)

jika nama file adalah None:

plt.show ()

lain :

fig.savefig (nama file)

Ini menyederhanakan skrip [Python](#) di Listing [10](#) menjadi:

Listing 10: [src / py / script / contoh\\_pca.py](#)

import matplotlib.cm sebagai cm

# ubah 16 vektor eigen pertama (paling banyak) menjadi skala abu-abu

# gambar (catatan: vektor eigen disimpan oleh kolom!)

E = []

untuk saya di xrange (min (len (X), 16)):

e = W[:, i].reshape (X [0].shape)

E.append (menormalkan (e, 0,255))

# plot mereka dan simpan plot ke "python\_eigenfaces.pdf"

subplot (title = "Eigenfaces AT&T Facedatabase" , gambar = E, baris = 4, cols = 4,

sptitle = "

Eigenface " , colormap = cm.jet, nama file = " python\_pca\_eigenfaces.png " )

Saya telah menggunakan jet colormap, sehingga Anda dapat melihat bagaimana nilai skala abu-abu didistribusikan dalam spesifikasi

cific Eigenfaces. Anda dapat melihat, bahwa Eigenfaces tidak hanya menyandikan fitur wajah, tetapi juga

iluminasi dalam gambar (lihat lampu kiri di Eigenface # 4, cahaya kanan di Eigenfaces # 5):

Eigenface # 1

Eigenface # 2

Eigenface # 3

Eigenface # 4

Eigenface # 5

Eigenface # 6

Eigenface # 7

Eigenface # 8

Eigenface # 9

Eigenface # 10

Eigenface # 11

Eigenface # 12

Eigenface # 13

Eigenface # 14

Eigenface # 15

Eigenface # 16

Eigenfaces AT&T Menghadapi basis data

Kami sudah melihat dalam Persamaan [5](#), bahwa kita dapat merekonstruksi wajah dari pendekatan dimensi yang lebih rendah-

mation. Jadi mari kita lihat berapa banyak Eigenfaces yang dibutuhkan untuk rekonstruksi yang baik. Saya akan melakukan subplot dengan

10, 30, ..., 310 Eigenfaces:

Listing 11: [src / py / script / contoh\\_pca.py](#)

dari proyek `import tinyfacerec.subspace` , merekonstruksi

`# langkah rekonstruksi`

```
steps = [i for i in xrange (10, min (len (X), 320), 20)]
```

```
E = []
```

```
untuk saya di xrange (min (len (langkah), 16)):
```

```
numEvs = langkah [i]
```

```
7
```

---

```
P = proyek (W[:, 0: numEvs], X [0] .reshape (1, -1), mu)
```

```
R = merekonstruksi (W[:, 0: numEvs], P, mu)
```

```
# membentuk kembali dan menambahkan plot
```

```
R = R.reshape (X [0] .shape)
```

```
E.append (menormalkan (R, 0,255))
```

```
# plot mereka dan simpan plot ke "python_reconstruction.pdf"
```

```
subplot (title = "Rekonstruksi AT&T Facedatabase" , gambar = E, baris = 4, cols =  
4, sptitle = "
```

```
Vektor eigen " , sptitles = langkah, colormap = cm.gray, nama file = "  
python_pca_reconstruction.png " )
```

10 vektor Eigen jelas tidak cukup untuk rekonstruksi gambar yang baik, 50 vektor Eigen mungkin

sudah cukup untuk menyandikan fitur wajah yang penting. Anda akan mendapatkan rekonstruksi yang baik dengan ap-

sekitar 300 vektor Eigen untuk database AT&T. Ada aturan praktis berapa banyak Eigenfaces Anda harus memilih untuk pengenalan wajah yang sukses, tetapi sangat bergantung pada input

data. [\[15\]](#) adalah titik sempurna untuk mulai meneliti untuk ini.

Vektor Eigen # 10

Vektor Eigen # 30

Vektor Eigen # 50

Vektor Eigen # 70

Vektor Eigen # 90

Vektor Eigen # 110

Vektor Eigen # 130

Vektor Eigen # 150

Vektor Eigen # 170

Vektor Eigen # 190

Vektor Eigen # 210

Vektor Eigen # 230

Vektor Eigen # 250

Vektor Eigen # 270

Vektor Eigen # 290

Vektor Eigen # 310

Rekonstruksi menghadapi database AT&T

Sekarang kita sudah mendapatkan segalanya untuk mengimplementasikan metode Eigenfaces. [Python](#) berorientasi objek dan begitu juga

model Eigenfaces kami. Mari kita rekap: Metode Eigenfaces pada dasarnya adalah Analisis Komponen Pricipal

dengan model Tetangga Terdekat. Beberapa publikasi melaporkan tentang pengaruh metrik jarak

(Saya tidak bisa mendukung klaim ini dengan penelitian saya), jadi berbagai metrik jarak untuk Tetangga Terdekat

harus didukung. Listing [12](#) mendefinisikan AbstractDistance sebagai kelas dasar abstrak untuk setiap jarak

metrik. Setiap subclass mengabaikan operator panggilan `__call__` seperti yang ditunjukkan untuk Euclidean Distance dan

Jarak cosine yang dinegasikan. Jika Anda membutuhkan metrik jarak lebih jauh, silakan lihat jaraknya

metrik diterapkan di <https://www.github.com/bytefish/facerec>.

Listing 12: [src / py / tinyfacerec / distance.py](#)

```
import numpy sebagai np
```

```
kelas AbstractDistance (objek):
```

```
def __init __ (diri, nama):
```

```
self._name = nama
```

```
def __call __ (self, p, q):
```

```
meningkatkan NotImplementedError ( "Setiap AbstractDistance harus
```

```
mengimplementasikan __call__
```

```
metode. " )
```

```
@Properti
```

```
nama def (diri):
```

```
kembalikan self._name
```

```
8
```

---

```
def __repr __ (mandiri):
```

```
kembalikan self._name
```

```
kelas EuclideanDistance (AbstractDistance):
```

```
def __init __ (mandiri):
```

```
AbstractDistance .__ init __ (mandiri, "EuclideanDistance" )
```

```
def __call __ (self, p, q):
```

```
p = np.asarray (p) .flatten ()
```

```
q = np.asarray (q) .flatten ()
```



```

return np.sqrt (np.sum (np.power ((pq), 2))))
kelas CosineDistance (AbstractDistance):
def __init __ (mandiri):
AbstractDistance .__ init __ (mandiri, "CosineDistance" )
def __call __ (self, p, q):
p = np.asarray (p) .flatten ()
q = np.asarray (q) .flatten ()
return -np.dot (pT, q) / (np.sqrt (np.dot (p, pT) * np.dot (q, qT)))

```

Metode Eigenfaces dan Fisherfaces sama-sama berbagi metode umum, jadi kami akan menentukan prediksi basis

model dalam Listing [13](#). Saya tidak ingin melakukan implementasi k-Nearest Neighbor penuh di sini, karena (1)

jumlah tetangga tidak terlalu penting untuk kedua metode dan (2) itu akan membingungkan orang. Jika

Anda menerapkannya dalam bahasa pilihan Anda, Anda harus benar-benar memisahkan ekstraksi fitur

dan klasifikasi dari model itu sendiri. Pendekatan generik nyata diberikan dalam kerangka [facerec](#) saya .

Namun, jangan ragu untuk memperluas kelas dasar ini untuk kebutuhan Anda.

Listing 13: [src / py / tinyfacerec / model.py](#)

```

import numpy sebagai np
dari util import asRowMatrix
dari subruang import pca, lda, fisherfaces, proyek
dari jarak import EuclideanDistance
kelas BaseModel (objek):
def __init __ (mandiri, X = Tidak ada, y = Tidak ada, dist_metric =
EuclideanDistance (), num_components

```

```

= 0):
self.dist_metric = dist_metric
self.num_components = 0
self.projections = []
self.W = []
self.mu = []
jika (X tidak ada) dan (y tidak ada):
    perhitungan sendiri (X, y)
    perhitungan def (self, X, y):
        meningkatkan NotImplementedError ( "Setiap BaseModel harus menerapkan
        metode komputasi." )
    prediksi def (mandiri, X):
        minDist = np.finfo ( 'float' ) .max
        minClass = -1
        Q = project (self.W, X.reshape (1, -1), self.mu)
        untuk saya di xrange (len (self.projections)):
            dist = self.dist_metric (self.projections [i], Q)
            jika dist < minDist:
                minDist = dist
                minClass = self.y [i]
        kembalikan minClass

```

Listing [20](#) kemudian subclass EigenfacesModel dari BaseModel , jadi hanya metode komputasi yang perlu ditimpa dengan ekstraksi fitur khusus kami. Prediksi ini adalah pencarian 1-Nearest Neighbor dengan metrik jarak.

Listing 14: [src / py / tinyfacerec / model.py](#)

kelas EigenfacesModel (BaseModel):

9

---

```
def __init__ (mandiri, X = Tidak ada, y = Tidak ada, dist_metric =  
EuclideanDistance (), num_components  
= 0):  
    super (EigenfacesModel, self) .__ init __ (X = X, y = y, dist_metric = dist_metric,  
num_components = num_components)  
    perhitungan def (self, X, y):  
        [D, self.W, self.mu] = pca (asRowMatrix (X), y, self.num_components)  
        # label toko  
        self.y = y  
        # proyeksi toko  
        untuk xi di X:
```

```
self.projections.append (proyek (self.W, xi.reshape (1, -1), self.mu))
```

Sekarang setelah EigenfacesModel didefinisikan, dapat digunakan untuk mempelajari Eigenfaces dan menghasilkan prediksi.

Dalam Listing [15](#) berikut kami akan memuat Yale Facedatabase A dan melakukan prediksi pada yang pertama gambar.

Listing 15: [src / py / script / contoh model eigenfaces.py](#)

```
import sys  
# tambahkan tinyfacerec ke jalur pencarian modul  
sys.path.append ( ".." )  
# impor colormaps numpy dan matplotlib  
import numpy sebagai np  
# impor modul tinyfacerec
```

```

dari tinyfacerec.util import read_images
dari tinyfacerec.model import EigenfacesModel
# baca gambar
[X, y] = read_images ( "/ home / philipp / facerec / data / yalefaces_recognition" )
# hitung model eigenfaces
model = EigenfacesModel (X [1:], y [1:])
# dapatkan prediksi untuk pengamatan pertama
cetak "diharapkan =" , y [0], "/" , "prediksi =" , model.predict (X [0])

```

## 2.3 Perikanan

Analisis Diskriminan Linier ditemukan oleh ahli statistik hebat [Sir RA Fisher](#), siapa yang berhasil-  
 sepenuhnya menggunakannya untuk mengklasifikasikan bunga dalam makalahnya tahun 1936 Penggunaan berbagai pengukuran dalam taksonomi masalah [8] Tetapi mengapa kita membutuhkan metode pengurangan dimensi lain, jika Principal Component Analysis (PCA) melakukan pekerjaan yang baik?  
 PCA menemukan kombinasi linear dari fitur-fitur yang memaksimalkan total varians dalam data. Sementara ini jelas cara yang ampuh untuk merepresentasikan data, itu tidak mempertimbangkan kelas apa pun dan banyak informasi diskriminatif dapat hilang ketika membuang komponen. Bayangkan situasi di mana varians dihasilkan oleh sumber eksternal, biarlah cahaya. Komponen yang diidentifikasi oleh a PCA tidak harus mengandung informasi diskriminatif sama sekali, jadi sampel yang diproyeksikan adalah dioleskan bersama dan klasifikasi menjadi tidak mungkin.

Untuk menemukan kombinasi fitur yang memisahkan terbaik di antara kelas-kelas, Linear Discriminant

Analisis memaksimalkan rasio antara-kelas ke dalam-kelas pencar. Idennya sederhana: sama

kelas-kelas harus berkelompok dengan erat, sementara kelas-kelas yang berbeda sejauh mungkin dari masing-masing

lain. Ini juga diakui oleh [Belhumeur , Hespanha](#) dan [Kriegman](#) dan karenanya mereka menerapkan a

Analisis Diskriminan untuk menghadapi pengakuan dalam [3]

### 2.3.1 Deskripsi Algoritma

Biarkan  $X$  menjadi vektor acak dengan sampel yang diambil dari kelas  $c$ :

$$X = \{X_1, X_2, \dots, X_c\}$$

(8)

$X_i$

$$= \{x_1, x_2, \dots, x_n\}$$

(9)

Matriks sebar  $S_B$  dan  $S_W$  dihitung sebagai:

10

---

$x$

$y$

$s_{W3}$

$s_{B3}$

$\mu_1$

$\mu_2$

$\mu_3$

$s_{B2}$

$s_{B1}$

$s_{W1}$

$s_{W2}$

$\mu$

Gambar 1: Gambar ini menunjukkan matriks sebar  $S_B$  dan  $S_W$  untuk masalah 3 kelas.  $\mu$  mewakili

total mean dan  $[\mu_1, \mu_2, \mu_3]$  adalah sarana kelas.

$S_B$

=

$c$

$\sum$

$i = 1$

$N_i (\mu_i - \mu) (\mu_i - \mu)^T$

(10)

$S_W$

=

$c$

$\sum$

$i = 1$

$\sum$

$x_j \in X_i$

$(x_j - \mu_i) (x_j - \mu_i)^T$

(11)

, di mana  $\mu$  adalah rata-rata total:

$\mu =$

$\frac{1}{N}$

$N$

N

$\sum$

$i = 1$

$x_i$

(12)

Dan  $\mu_i$  adalah rata-rata kelas  $i \in \{1, \dots, c\}$ :

$\mu_i =$

1

$|X_i|$

$\sum$

$x_j \in X_i$

$x_j$

(13)

Algoritma klasik Fisher sekarang mencari proyeksi  $W$ , yang memaksimalkan kriteria pemisahan kelas:

$W_{opt} = \arg \max W$

$|W^T S B W|$

$|W^T S W W|$

(14)

Mengikuti [3], solusi untuk masalah optimasi ini diberikan dengan memecahkan nilai Eigen Umum

Masalah:

$S B v_i$

$= \lambda_i S w v_i$

$S^{-1}$

$W S B v_i$

$= \lambda_i v_i$

(15)

Ada satu masalah yang tersisa untuk diselesaikan: Peringkat S W paling banyak (N - c), dengan N sampel dan kelas c. Di

masalah pengenalan pola, jumlah sampel N hampir selalu sama dengan dimensi dari data input (jumlah piksel), sehingga matriks pencar S W menjadi singular (lihat [2]). Di

[3] ini diselesaikan dengan melakukan Analisis Komponen Utama pada data dan memproyeksikan

11

---

sampel ke dalam ruang dimensi (N - c). Sebuah Analisis Diskriminan Linier kemudian dilakukan pada

mengurangi data, karena SW tidak lagi tunggal.

Masalah optimisasi dapat ditulis ulang sebagai:

$W_{pca}$

$= \arg \max W | W^T S T W |$

(16)

$W_{fld}$

$= \arg \max W$

$| W^T W T$

$pca S B W_{pca} W |$

$| W^T W T$

$pca S W W_{pca} W |$

(17)

Matriks transformasi W, yang memproyeksikan sampel ke dalam ruang dimensi (c - 1) kemudian diberikan

oleh:



$W = W^T$

fld  $W^T$

pca

(18)

Satu catatan terakhir: Meskipun  $SW$  dan  $S^T B$  adalah matriks simetris, produk dari dua matriks simetris

belum tentu simetris. jadi Anda harus menggunakan pemecah nilai eigen untuk matriks umum. OpenCV

[cv::eigen](#) hanya berfungsi untuk matriks simetris dalam versi saat ini; karena nilai eigen dan nilai singular

tidak setara dengan matriks non-simetris yang tidak dapat Anda gunakan dengan Singular Value Decomposition (SVD)

antara.

### 2.3.2 Fisherfaces dengan Python

Menerjemahkan Analisis Diskriminan Linear ke [Python](#) hampir sepele lagi, lihat Listing [16](#). Untuk

memproyeksikan dan merekonstruksi dari dasar Anda dapat menggunakan fungsi-fungsi dari Listing [5](#) dan [6](#).

Listing 16: [src / py / tinyfacerec / subspace.py](#)

```
def lda (X, y, num_components = 0):
```

```
    y = np.asarray (y)
```

```
    [n, d] = X.bentuk
```

```
    c = np.unique (y)
```

```
    if (num_components <= 0) atau (num_component > (len (c) - 1)):
```

```
        num_components = (len (c) - 1)
```

```
    meanTotal = X.mean (sumbu = 0)
```

```
    Sw = np.zeros ((d, d), dtype = np.float32)
```

```
Sb = np.zeros ((d, d), dtype = np.float32)
```

untuk saya dalam c:

```
Xi = X [np.where (y == i) [0] ,:]
```

```
meanClass = Xi.mean (sumbu = 0)
```

```
Sw = Sw + np.dot ((Xi-meanClass) .T, (Xi-meanClass))
```

```
Sb = Sb + n * np.dot ((meanClass - meanTotal) .T, (meanClass - meanTotal))
```

```
nilai eigen, vektor eigen = np.linalg.eig (np.linalg.inv (Sw) * Sb)
```

```
idx = np.argsort (-eigenvalues.real)
```

```
nilai eigen, vektor eigen = nilai eigen [idx], vektor eigen[:, idx]
```

```
nilai eigen = np.array (nilai eigen [0: num_components] .real, dtype = np.float32,
```

```
copy =
```

```
Benar)
```

```
vektor eigen = np.array (vektor eigen [0:, 0: num_components] .real, dtype =
```

```
np.float32,
```

```
copy = Benar)
```

```
return [nilai eigen, vektor eigen]
```

Fungsi untuk melakukan PCA (Listing 4) dan LDA (Listing 16) sekarang didefinisikan, jadi kita bisa pergi

maju dan mengimplementasikan Fisherfaces from Equation 18.

Listing 17: [src / py / tinyfacerec / subspace.py](#)

```
def fisherfaces (X, y, num_components = 0):
```

```
y = np.asarray (y)
```

```
[n, d] = X.bentuk
```

```
c = len (np.unique (y))
```

```
[eigenvalues_pca, eigenvectors_pca, mu_pca] = pca (X, y, (nc))
```

```
[eigenvalues_lda, eigenvectors_lda] = lda (proyek (eigenvectors_pca, X, mu_pca),
```

```
y,
```

komponen num\_komponen)

vektor eigen = np.dot (eigenvectors\_pca, eigenvectors\_lda)

return [eigenvalues\_lda, vektor eigen, mu\_pca]

Untuk contoh ini saya akan menggunakan Yale Facedatabase A, hanya karena plotnya lebih bagus. Setiap

Fisherface memiliki panjang yang sama dengan gambar asli, sehingga dapat ditampilkan sebagai gambar. Kami akan kembali

memuat data, mempelajari Fisherfaces dan membuat subplot dari 16 Fisherfaces pertama.

12

---

Listing 18: [src / py / script / contoh fisherfaces.py](#)

```
import sys
```

```
# tambahkan tinyfacerec ke jalur pencarian modul
```

```
sys.path.append ( ".." )
```

```
# import colormaps numpy dan matplotlib
```

```
import numpy sebagai np
```

```
# import modul tinyfacerec
```

```
dari tinyfacerec.subspace import fisherfaces
```

```
dari tinyfacerec.util import dinormalkan, asRowMatrix, read_images
```

```
dari tinyfacerec.visual import subplot
```

```
# baca gambar
```

```
[X, y] = read_images ( "/ home / philipp / facerec / data / yalefaces_recognition" )
```

```
# melakukan pca penuh
```

```
[D, W, mu] = fisherfaces (asRowMatrix (X), y)
```

```
#import colormaps
```

```
import matplotlib.cm sebagai cm
```

```

# ubah 16 vektor eigen pertama (paling banyak) menjadi skala abu-abu
# gambar (catatan: vektor eigen disimpan oleh kolom!)
E = []
untuk saya dalam xrange (min (W.shape [1], 16)):
e = W[:, i].reshape (X [0].shape)
E.append (menormalkan (e, 0,255))
# plot mereka dan simpan plot ke "python_fisherfaces_fisherfaces.pdf"
subplot (title = "Fisherfaces AT&T Facedatabase", gambar = E, baris = 4, cols = 4,
sptitle = "
Fisherface", colormap = cm.jet, filename = "python_fisherfaces_fisherfaces.pdf
")

```

Metode Fisherfaces mempelajari matriks transformasi kelas-spesifik, sehingga mereka tidak menangkap

iluminasi sejelas metode Eigenfaces. Analisis Diskriminan malah menemukan wajah

fitur untuk membedakan antara orang-orang. Sangat penting untuk menyebutkan, bahwa kinerja

Periklanan sangat bergantung pada input data juga. Praktis berkata: jika Anda belajar untuk Fisherfaces

hanya gambar yang diterangi dengan baik dan Anda mencoba mengenali wajah dalam adegan yang kurang terang, lalu metode

kemungkinan menemukan komponen yang salah (hanya karena fitur-fitur itu mungkin tidak dominan

gambar iluminasi buruk). Ini agak logis, karena metode ini tidak memiliki kesempatan untuk mempelajari

penerangan.

Fisherface # 1

Fisherface # 2

Fisherface # 3

Fisherface # 4

Fisherface # 5

Fisherface # 6

Fisherface # 7

Fisherface # 8

Fisherface # 9

Fisherface # 10

Fisherface # 11

Fisherface # 12

Fisherface # 13

Fisherface # 14

Database AT&T Fisherfaces

Fisherfaces memungkinkan rekonstruksi gambar yang diproyeksikan, seperti yang dilakukan Eigenfaces. Tapi sejak itu

kami hanya mengidentifikasi fitur untuk membedakan antara subjek, Anda tidak dapat mengharapkan perkiraan yang bagus

dari gambar aslinya. Kita dapat menulis ulang Listing [11](#) untuk metode Fisherfaces ke Listing [19](#), tapi ini

kali kami akan memproyeksikan gambar sampel ke masing-masing Fisherfaces sebagai gantinya. Jadi Anda akan memiliki visualisasi, yang menggambarkan setiap fitur Fisherface.

13

---

Listing 19: [src / py / script / contoh fisherfaces.py](#)

dari proyek `import tinyfacerec.subspace`, merekonstruksi

E = []

untuk saya dalam xrange (min (W.shape [1], 16)):

e = W[:, i].reshape (-1,1)

P = proyek (e, X [0].reshape (1, -1), mu)

R = merekonstruksi (e, P, mu)

# membentuk kembali dan menambahkan plot

R = R.reshape (X [0].shape)

E.append (menormalkan (R, 0,255))

# plot mereka dan simpan plot ke "python\_reconstruction.pdf"

subplot (title = "Rekonstruksi Fisherfaces Yale FDB" , gambar = E, baris = 4, cols  
= 4, sptitle

= "Fisherface" , colormap = cm.gray, filename

= "python\_fisherfaces\_reconstruction.pdf" )

Fisherface # 1

Fisherface # 2

Fisherface # 3

Fisherface # 4

Fisherface # 5

Fisherface # 6

Fisherface # 7

Fisherface # 8

Fisherface # 9

Fisherface # 10

Fisherface # 11

Fisherface # 12

Fisherface # 13

Fisherface # 14

## Rekonstruksi Periklanan Yale FDB

Detail implementasi tidak diulangi di bagian ini. Untuk metode Fisherfaces serupa model ke EigenfacesModel di Listing [20](#) harus ditentukan.

Listing 20: [src / py / tinyfacerec / model.py](#)

```
class FisherfacesModel (BaseModel):
```

```
def __init__ (mandiri, X = Tidak ada, y = Tidak ada, dist_metric =
```

```
EuclideanDistance (), num_components
```

```
= 0):
```

```
super (FisherfacesModel, self) .__ init__ (X = X, y = y, dist_metric = dist_metric,  
num_components = num_components)
```

```
perhitungan def (self, X, y):
```

```
[D, self.W, self.mu] = fisherfaces (asRowMatrix (X), y, self.num_components)
```

```
# label toko
```

```
self.y = y
```

```
# proyeksi toko
```

```
untuk xi di X:
```

```
self.projections.append (proyek (self.W, xi.reshape (1, -1), self.mu))
```

Setelah FisherfacesModel didefinisikan, itu dapat digunakan untuk mempelajari Fisherfaces dan menghasilkan prediksi.

Dalam Listing [21](#) berikut kami akan memuat Yale Facedatabase A dan melakukan prediksi pada yang pertama gambar.

Listing 21: [src / py / script / contoh model fisherfaces.py](#)

14

---

```
import sys
```

```
# tambahkan tinyfacerec ke jalur pencarian modul
```

```
sys.path.append ( ".." )  
# impor colormaps numpy dan matplotlib  
impor numpy sebagai np  
# impor modul tinyfacerec  
dari tinyfacerec.util impor read_images  
dari tinyfacerec.model impor FisherfacesModel  
# baca gambar  
[X, y] = read_images ( "/ home / philipp / facerec / data / yalefaces_recognition" )  
# hitung model eigenfaces  
model = FisherfacesModel (X [1:], y [1:])  
# dapatkan prediksi untuk pengamatan pertama  
cetak "diharapkan =" , y [0], "/" , "prediksi =" , model.predict (X [0])
```

### 3 Kesimpulan

Dokumen ini menjelaskan dan mengimplementasikan Eigenfaces [13] dan Fisherfaces [3] metode dengan

[GNU Octave/ MATLAB, Python](#). Ini memberi Anda beberapa ide untuk memulai dan meneliti ini sangat aktif tema. Saya harap Anda bersenang-senang membaca dan saya harap Anda berpikir [cv :: FaceRecognizer](#) adalah tambahan yang bermanfaat OpenCV.

Lebih mungkin di sini:

- <http://www.opencv.org>
- <http://www.bytefish.de/blog>
- <http://www.github.com/bytefish>



## **ABSTRAK**

Pengenalan wajah adalah salah satu teknologi paling trending yang digunakan di mana-mana di dunia saat ini.

Ini melibatkan bagian yang dipindai dari bagian tubuh kita yang unik untuk semua orang dan disimpan dalam

database yang membuat peretas tidak mungkin mencuri kata sandi atau informasi pribadi apa pun. Ada banyak hal

banyak keuntungan dan banyak membantu para ilmuwan, partai politik untuk melindungi masalah-masalah partai pribadi mereka

dari mencuri oleh pihak lain atau peretas. Pengenalan wajah melibatkan banyak pengkodean dan bahkan

membutuhkan pengetahuan matematika untuk mengambil informasi seseorang. Topik matematika seperti vektor eigen

dan integrasi digunakan untuk mengambil data, karena nilai-nilai eigen dan integrasi ini memakan waktu lama

interaksi bagian tubuh komputer. Sekarang pengenalan wajah membuat peretas tidak mungkin untuk mencuri

kata sandi seseorang dan meretas barang-barang mereka yang harus dijaga dengan aman. Pengenalan wajah bermanfaat

bagi orang-orang seperti wartawan, partai politik untuk memastikan data mereka aman dan melindungi data mereka

mencuri oleh partai politik lain atau oleh para pencuri. Topik kita adalah pengenalan wajah yang mana

teknologi yang paling banyak digunakan dalam kehidupan kita sehari-hari. Ini melibatkan pemindaian wajah dan mendapatkan titik pada wajah yang mana

unik untuk semua orang dan mendapatkan poin menggunakan topik matematika seperti vektor eigen dan integrasi.

Ini juga melibatkan bahasa python dan bahasa matlab. Matlab digunakan untuk mendapatkan nilai matematika dan

python adalah bahasa pengkodean umum yang digunakan dalam biometrik dan juga digunakan dalam kecerdasan buatan

dan pembelajaran mesin. Python memiliki pustaka inbuilt yang kami gunakan darinya dan mendapatkan nilai nominalnya.

Semua wajah ini disimpan dalam database dan setiap kali kita menyimpan wajah yang hidup itu mengenali wajah

orang yang sudah ada dalam database. Bahkan pengenalan wajah digunakan dalam keamanan data dan

laboratorium keamanan tinggi di mana hanya orang terbatas yang memiliki akses untuk mengontrol laboratorium. Jadi pengenalan wajah

bahkan digunakan di sana untuk memastikan keamanan yang tinggi bagi perusahaan. Hampir semua perusahaan perangkat lunak,

lembaga pendidikan memiliki teknologi pengenalan wajah ini untuk memastikan penipuan atau membuat mereka lebih

nyaman. Dan sekarang di proyek kami, kami melihat proses yang kompleks ini akan terjadi dalam hal yang lebih sederhana.

**Kata kunci:** pengenalan wajah, deteksi wajah, python, matlab, database

## **1. PERKENALAN**

Sekarang pengakuan wajah hari berkembang pesat. Begitu banyak orang meneliti tentang itu. Kita perlu memiliki pengetahuan tambahan untuk mengetahuinya.

Kita harus tahu tentang highlight wajah dan invarian geometrik. Tergantung pada revolusi wajah dan distorsi. Kami bahkan memiliki beberapa kelebihan dan kerugian tapi salah satu kelemahan paling aneh adalah ada yang kurang solid invarian.

Deteksi wajah melibatkan prosedur 2 langkah: 1 berisi wajah. ini sulit menemukan wajah sehingga langkah pertama yang terlibat dalam teknologi pengenalan wajah

adalah untuk mendapatkan atau menangkap wajah. Ada beberapa masalah saat menangkap

wajah seperti cahaya, latar belakang, kualitas dan banyak lagi. Jadi kita harus punya detektor wajah ideal yang mendeteksi wajah apa pun pada suatu titik waktu. Setelah mendapatkan

input menggunakan face detector, kita bisa mendapatkan output dalam 2 cara satu cara

menjaga semua gambar di folder sebagai input dan ketika kita memindai wajah maka akan mengatakan apakah gambar yang diberikan ada atau tidak, jika gambar yang diberikan ada

1 Departemen Sistem Perangkat Lunak, RUANG LINGKUP, Universitas VIT, Vellore, TN, India.

2 School of Compting, EGSPillay Engineering College, Nagapattinam, TN, India.

3 Departemen Matematika, Sekolah Tinggi Teknik Universitas Ariyalur, TN, India.

4 Departemen Ilmu dan Teknik Komputer, Fakultas Teknik Universitas Ariyalur, TN, India.

---

**Prosiding Simposium Internasional ke- 9 (Full Paper)** , South Eastern University of Sri Lanka,

Oluvil. 27 th - 28 th November 2019, *ISBN: 978-955-627-189-8*

maka itu akan memberikan output ya atau yang lain itu akan memberikan output no. Metode selanjutnya

ketika kita memberikan wajah sebagai input maka itu akan memeriksa dan mengukur semua

istilah aljabar seperti lebar, tinggi, dan warna dan kemudian akan mengenali wajah input yang diberikan.

## **2. SURVEI SASTRA**

Deteksi wajah dapat dibagi menjadi 2 jenis: Pemrosesan sebelum mereka disimpan dalam database setelah kami memberikan input maka gambar sudah diproses dan kemudian jika gambarnya jelas maka mereka disimpan dalam database kalau tidak mereka harus diambil kembali. Sekarang gambar yang diambil diklasifikasikan menjadi

kategori di mana kami menggunakan matlab untuk klasifikasi. Kita juga bisa menggunakan banyak

sistem jaringan lain tetapi dalam proyek ini kami telah menggunakan python dan matlab, jadi

kami menggunakan matlab untuk menyaring. Menempatkan gambar. Semua gambar itu

disimpan dan diekstraksi disimpan tetapi saatnya untuk mengetahui milik gambar manakepada orang tertentu. Jadi semua gambar disimpan dan diberi nama begitu masuk database, jadi ketika waktu berikutnya ketika kita memindai wajah maka akan ditampilkan

output dari database yang disimpan.

### **3. METODOLOGI**

Ekstraksi gambar untuk memfilter dan mengatur gambar: Sekarang kita menggunakan hari

ton foto yang diambil di DSLR, ponsel atau digital

kamera. Jadi ada kebutuhan untuk membuat aplikasi foto yang telah dibuat. Ini akan membantu kami dalam menjelajah dan mengatur foto secara sederhana. Ini aplikasi akan membantu dalam mengatur foto secara berurutan menggunakan di atas pendekatan otomatis yang akan memilah gambar dan melabeli mereka. Jadi yang ini alat yang paling berguna yang dikembangkan.

#### **3.1 Menggunakan tanda tangan eigen untuk pengenalan wajah umum**

Dalam proses ini kita akan mengekstraksi gambar menggunakan nilai eigen. Ini membantu mendapatkan wajah untuk wajah dengan ekspresi berbeda. Ini juga membantu dalam mengidentifikasi gambar itu nyata atau palsu. Ini akan mendeteksi gambar palsu dan hanya menerima yang asli

wajah. Ia juga mengenali wajah bahkan di bawah perubahan tertentu seperti rotasi, kedalaman. Ini juga mengenali wajah yang bergerak seperti ketika video diputar kita bisa menangkap gambar. Ini sebagian besar tergantung pada pemfilteran dari gambar asing ke gambar akrab.

#### **3.2 Menggunakan pola biner lokal untuk pengenalan wajah**

Makalah ini menjelaskan penggunaan dan mempertimbangkan bentuk dan pola tekstur. Theare dibagi menjadi beberapa tes chi-square. Chi-square adalah ujian yang memiliki kisi atau membagi wajah sedemikian rupa sehingga memiliki jumlah yang sama

baris dan jumlah kolom yang sama. Pengenalan wajah adalah berbasis komputer teknologi di mana kami mendeteksi hanya wajah tetapi benda lain seperti pohon,

bangunan tidak terdeteksi.

---

**Prosiding Simposium Internasional ke- 9 (Full Paper)** , South Eastern University of Sri Lanka,

Oluvil. 27 th - 28 th November 2019, *ISBN: 978-955-627-189-8*

### **3.3 Pengenalan wajah memiliki dua cara untuk mendeteksi gambar**

Salah satu caranya adalah menemukan wajah dan mendeteksi semua titik di dalamnya sehingga wajah itu

diakui dan cara ini disebut pengenalan wajah. Cara lain adalah mendeteksi

wajah menggunakan metode coba-coba di mana kami memberikan banyak input gambar dan

ketika wajah dipindai, ia akan memeriksa setiap wajah kemudian

perlihatkan wajah yang cocok. Jadi dalam hal ini kami memeriksa setiap wajah dan percobaan

dan metode kesalahan digunakan.

## **4. ARSITEKTUR SISTEM**

Struktur arsitektur dari pekerjaan kami diberikan pada Gambar 1. Pertama, kita ambil

gambar input menggunakan detektor pindai wajah maka wajah ini mendapat pemisahan menggunakan

deteksi fitur wajah. Kedua, kami menerapkan pengklasifikasi dan membandingkannya dengan

gambar inbuilt menggunakan algoritma viola jones. Akhirnya, wajah yang dihasilkan adalah terdeteksi atau tidak atau kita mendapatkan nama wajah yang telah kita simpan di Internet masukan basis data. Fitur tepi, fitur garis, dan fitur surround pusat untuk berbeda sudut diperoleh. Gambar 2. Menunjukkan fitur yang disebutkan di atas obyek.

---

**Prosiding Simposium Internasional ke- 9 (Full Paper)** , South Eastern University of Sri Lanka,

Oluvil. 27 th - 28 th November 2019, *ISBN: 978-955-627-189-8*

## **5. ANALISIS SISTEM DAN HASIL**

Gambar input dipandang sebagai gambar piramida. Preprocessing adalah dicapai menggunakan pemerataan histogram dan koreksi untuk pencahayaan tercapai. Lapisan input dan lapisan tersembunyi dianalisis dan diperoleh dengan menggunakan neural jaringan, dan output diperoleh. Langkah-langkah ini diringkas sebagai di bawah.

### **5.1 Algoritma Deteksi Wajah**

Gambar. 3 menunjukkan fungsi algoritma deteksi wajah.

*Gambar.3. Algoritma Deteksi Wajah Ikuti*

- Pemrosesan dilakukan sebelum disimpan dalam database. Begitu input diterima, sedang menjalani pra-pemrosesan, lalu pengecekannya dilakukan untuk kejelasan gambar. Jika gambarnya jelas maka mereka disimpan dalam database jika tidak, permintaan dibuat untuk pengambilan kembali gambar.
- Sekarang gambar ini diklasifikasikan ke dalam kategori. Untuk tujuan ini, kami

gunakan matlab dan klasifikasi dilakukan. Kita juga bisa menggunakan banyak lainnya

sistem jaringan tetapi dalam pekerjaan ini, kami telah menggunakan python dan matlab, jadi kami menggunakan matlab untuk penyaringan.

- Semua gambar yang disimpan dan diekstraksi disimpan tetapi waktunya untuk tahu gambar mana milik orang tertentu. Lokasi gambar adalah

perhatian penting berikutnya dari proses algoritmik kami. Jadi semua gambar disimpan dan dinamai begitu mereka berada di database, sehingga lain kali ketika kita memindai wajah maka akan memproses output dari membandingkan gambar yang disimpan dalam database.

- Dasar skala abu-abu: Wajah kita terdiri dari banyak bagian seperti alis dan peupa. Warna di sekitar wilayah ini lebih gelap dari warna wajah. Jadi tingkat abu-abu membuat semua wajah dengan warna konstan dan kemudian

mengeksktraksi gambar menggunakan matlab.

## **5.2 Analisis Tingkat Rendah**

Ini didasarkan pada warna, di mana warna penyaringan jauh lebih baik daripada menyaring wajah

menggunakan titik koordinat. Dalam hal ini warna wajah difilter dan dikenali tetapi ini bukan pendekatan yang tepat karena orang yang memiliki warna yang sama mungkin

kadang-kadang diakui dan ini dapat menyebabkan tabrakan antara wajah

---

**Prosiding Simposium Internasional ke- 9 (Full Paper)** , South Eastern University of Sri Lanka,

Oluvil. 27 th - 28 th November 2019, *ISBN: 978-955-627-189-8*

orang dan bahkan ketika orang itu bergerak, itu tidak dapat dideteksi karena



kurang cahaya dan dipengaruhi oleh banyak faktor.

Ada 3 cara untuk mendapatkan pendekatan ini:

1. Dapatkan gambar wajah input dan saring.
2. Terapkan semua topeng untuk mendapatkan yang terbaik dari itu.
3. Terapkan teorema konvolusi dan dapatkan koordinat dan ekstrak gambar.

*Gambar.4. Layar Pemrograman Python*

*Gambar.5. Output Sampel*

Gambar 4 dan Gambar 5 menunjukkan tangkapan layar pemrograman Python dan sampel masing-masing.

## **6. KESIMPULAN**

Ada beberapa implementasi yang tidak menghasilkan 3D aplikasi. Seiring perubahan waktu, perangkat lunak harus diperbarui, sementara perangkat lunak yang ketinggalan jaman masih digunakan. Wajah kita tidak hanya mengandung kulit tetapi juga berisi rambut yang algoritmanya mengalami kesulitan untuk memindai gambar sehingga algoritma

---

**Prosiding Simposium Internasional ke- 9 (Full Paper)** , South Eastern University of Sri Lanka,

Oluvil. 27 th - 28 th November 2019, *ISBN: 978-955-627-189-8*

bekerja dengan cara yang berbeda. Ketika kami memindai gambar yang berbeda, dimensi dan intensitas bervariasi. Ini menyebabkan kesulitan untuk mengekstrak gambar. Sampah di penyimpanan dalam database ketika kita mengambil lebih banyak foto dari yang dibutuhkan. Sistem kami meminimalkan atau menghindari pemborosan dalam storage. Beberapa sistem sulit melakukannya menangani perangkat lunak. Tetapi sistem kami memudahkan penggunaan perangkat

lunak. Sekarang kami menemukan aplikasi wajah sedang digunakan di aplikasi media sosial seperti snapchat, Instagram, facebook, dll. Oleh karena itu, sistem kami berkinerja baik hingga yang terbaru





## Referensi

- [1] Ahonen, T., Hadid, A., dan Pietikainen, M. Pengenalan Wajah dengan Pola Biner Lokal.  
Computer Vision - ECCV 2004 (2004), 469–481.
- [2] AK Jain, SJR Efek ukuran sampel kecil dalam pengenalan pola statistik: Rekomendasi  
untuk para praktisi. Transaksi IEEE pada Analisis Pola dan Kecerdasan Mesin 13, 3 (1991),  
252–264.
- [3] Belhumeur, PN, Hespanha, J., dan Kriegman, D. Eigenfaces vs. fisherfaces: Recogni-  
tion menggunakan proyeksi linear spesifik kelas. Transaksi IEEE pada Analisis  
Pola dan Mesin  
Kecerdasan 19, 7 (1997), 711-720.
- [4] Brunelli, R., dan Poggio, T. Pengenalan wajah melalui fitur geometris. Di  
Eropa  
Conference on Computer Vision (ECCV) (1992), hlm. 792–800.
- [5] Cardinaux, F., Sanderson, C., dan Bengio, S. otentikasi pengguna melalui  
statistik yang disesuaikan  
model gambar wajah. Transaksi IEEE pada Pemrosesan Sinyal 54 (Januari 2006),  
361-373.
- [6] Chiara Turati, Viola Macchi Cassia, FS, dan Leo, I. Pengenalan wajah bayi  
baru lahir: Peran  
fitur wajah bagian dalam dan luar. Perkembangan Anak 77, 2 (2006), 297–311.
- [7] Duda, RO, Hart, PE, dan Bangau, Klasifikasi Pola DG (Edisi ke-2), 2 ed.

November 2001.

[8] Fisher, RA Penggunaan berbagai pengukuran dalam masalah taksonomi. *Annals Eugen.* 7 (1936), 179–188.

[9] Kanade, T. Sistem pemrosesan gambar oleh komplek komputer dan pengenalan wajah manusia.

Tesis PhD, Universitas Kyoto, November 1973.

[10] Lee, K.-C., Ho, J., dan Kriegman, D. Memperoleh subruang linier untuk pengenalan wajah berdasarkan pencahayaan variabel. *Transaksi IEEE pada Analisis Pola dan Kecerdasan Mesin (PAMI)* 27, 5 (2005).

[11] Maturana, D., Mery, D., dan Soto, A. Pengenalan wajah dengan pola biner lokal, spasial

histogram piramida dan naif bayes klasifikasi tetangga terdekat. *Konferensi Internasional 2009*

ence dari Masyarakat Ilmu Komputer Chili (SCCC) (2009), 125–132.

[12] Rodriguez, Y. Deteksi Wajah dan Verifikasi menggunakan Pola Biner Lokal. Tesis PhD, École Polytechnique Fédérale De Lausanne, Oktober 2006.

[13] Turk, M., dan Pentland, A. Eigenfaces untuk pengakuan. *Jurnal Ilmu Saraf Kognitif* 3 (1991), 71-86.

[14] Wiskott, L., Fellous, J.-M., Krüger, N., dan Malsburg, CVD Pengenalan wajah oleh pencocokan grafik banyak elastis. *TRANSAKSI IEEE TERHADAP ANALISA POLA DAN KECERDASAN MESIN* 19 (1997), 775-779.

[15] Zhao, W., Chellappa, R., Phillips, P., dan Rosenfeld, A. Pengenalan wajah: Sebuah literatur survei. *Acm Computing Survey (CSUR)* 35, 4 (2003), 399–458.

