# JOBSHEET 1
# KONSEP KECERDASAN BUATAN

## A. TUJUAN

1. Mahasiswa memahami konsep kecerdasan buatan
2. Mahasiswa mampu menjelaskan implementasi kecerdasan buatan
3. Mahasiswa mampu membangun proyek sederhana untuk implementasi kecerdasan buatan

## B. PETUNJUK

Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan. Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar. Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur. Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

## C. DASAR TEORI

### i. Citra Digital

Citra digital juga dapat diproses dan dimanipulasi menggunakan bahasa pemrograman Python. Python adalah bahasa pemrograman yang populer untuk pengolahan citra digital karena memiliki banyak modul dan paket yang berguna untuk pengolahan citra. Beberapa modul dan paket populer yang digunakan dalam pengolahan citra digital dengan Python adalah NumPy, OpenCV, Matplotlib, dan PIL (Python Imaging Library). NumPy digunakan untuk mengolah data array multidimensi, OpenCV adalah library populer untuk pengolahan citra, Matplotlib digunakan untuk visualisasi data, dan PIL digunakan untuk membaca, memodifikasi, dan menyimpan gambar.

Dengan menggunakan Python, pengguna dapat mengimplementasikan algoritma pengolahan citra seperti pengukuran intensitas, deteksi tepi, segmentasi gambar, dan

penapisan gambar. Python juga memungkinkan pengguna untuk mengembangkan aplikasi pengolahan citra berbasis web dengan menggunakan kerangka kerja web seperti Flask atau Django. Selain itu, Python dapat digunakan dalam pembuatan aplikasi machine learning dan deep learning untuk pengolahan citra.

Namun, penggunaan Python dalam pengolahan citra digital juga memiliki beberapa kelemahan. Pertama, Python adalah bahasa pemrograman yang lebih lambat dibandingkan dengan bahasa pemrograman lain seperti C atau C++. Kedua, beberapa paket dan modul Python untuk pengolahan citra digital membutuhkan pengaturan dan konfigurasi yang rumit. Terakhir, beberapa algoritma pengolahan citra digital yang kompleks memerlukan penggunaan hardware yang lebih kuat untuk mengolah data dan meningkatkan kecepatan pemrosesan.

ii. **OpenCV**

OpenCV (Open Source Computer Vision Library) adalah sebuah library open source yang digunakan untuk memproses citra dan video pada bahasa pemrograman Python. OpenCV memiliki banyak fitur yang dapat digunakan untuk mengolah citra seperti deteksi objek, segmentasi, pengenalan wajah, pelacakan, dan pengolahan citra secara umum. OpenCV telah digunakan dalam berbagai aplikasi seperti pemrosesan citra medis, industri otomotif, robotika, dan penglihatan komputer.

Dalam pengolahan citra dengan OpenCV Python, terdapat beberapa konsep dasar yang perlu dipahami. Konsep pertama adalah representasi citra dalam format array NumPy, dimana setiap piksel dalam citra direpresentasikan sebagai elemen dalam array multidimensi. Konsep kedua adalah pengolahan citra dengan menggunakan filter, dimana filter digunakan untuk meningkatkan kejernihan, mengurangi noise, atau menekankan fitur-fitur dalam

citra. Konsep ketiga adalah deteksi fitur dalam citra seperti tepi, sudut, dan titik.

Selain itu, OpenCV Python juga menyediakan berbagai algoritma machine learning seperti SVM, Naive Bayes, dan KNN yang dapat digunakan untuk klasifikasi citra. OpenCV juga menyediakan fitur pelacakan objek, dimana objek dapat dipantau dan dilacak pergerakannya dalam suatu video atau rentetan citra. Selain itu, OpenCV juga memiliki fitur Face Detection dan Face Recognition yang dapat digunakan untuk pengenalan wajah.

iii. **Haar Cascade**

Haar Cascade adalah sebuah algoritma deteksi objek yang digunakan dalam pengolahan citra dan pengenalan pola. Algoritma ini digunakan untuk mendeteksi objek dalam citra atau video dengan cara membandingkan pola citra dengan pola wajah atau objek yang telah disimpan dalam file XML. Algoritma Haar Cascade merupakan salah satu algoritma deteksi objek paling populer dalam pengenalan wajah.

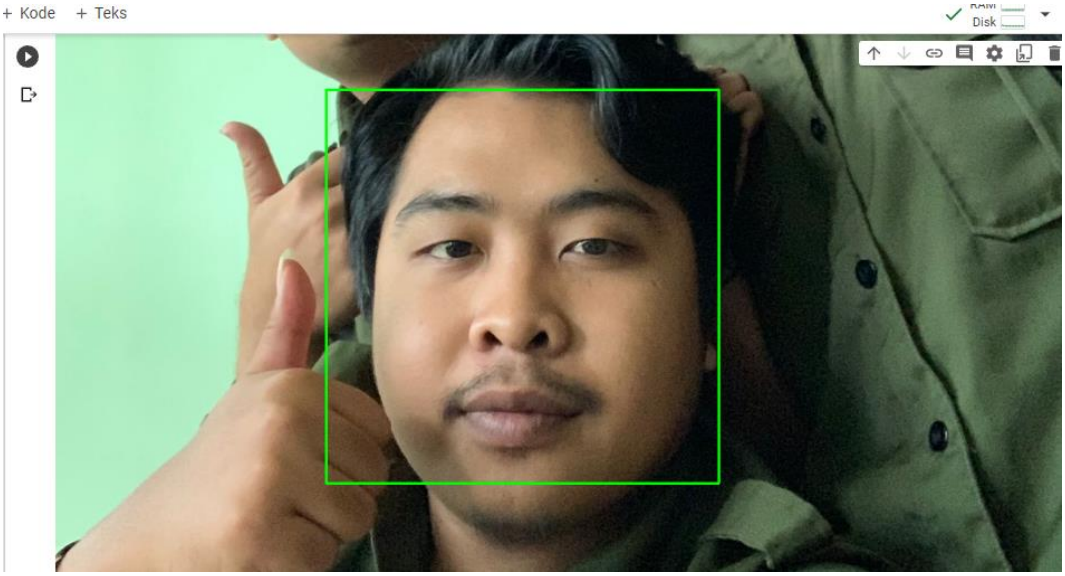
Proses deteksi objek dengan algoritma Haar Cascade dimulai dengan pengambilan gambar dan mengubahnya ke dalam bentuk grayscale. Selanjutnya, gambar tersebut diukur dengan menggunakan kernel Haar Cascade, yang merupakan sejumlah kernel atau filter yang terdiri dari blok pola hitam dan putih dengan ukuran yang berbeda. Kemudian, algoritma Haar Cascade memproses setiap blok dalam gambar dan menghitung nilai integralnya. Nilai integral digunakan untuk menghitung fitur Haar dan kemudian fitur-fitur tersebut digunakan untuk mengklasifikasikan gambar sebagai gambar dengan objek atau gambar tanpa objek.

Salah satu keunggulan dari algoritma Haar Cascade adalah kemampuannya untuk mendeteksi objek dalam waktu yang cepat dan akurat. Algoritma ini juga dapat digunakan dalam berbagai jenis objek seperti wajah, mobil, atau bahkan benda-benda yang lebih kompleks. Selain itu, pengguna dapat membuat file XML kustom untuk mendeteksi objek khusus yang tidak termasuk dalam file XML bawaan OpenCV. Namun, algoritma Haar Cascade juga memiliki beberapa kelemahan seperti sensitif terhadap perubahan skala dan rotasi, serta kurang akurat ketika menghadapi objek dengan pola yang kompleks dan berbeda dari pola yang telah disimpan dalam file XML.

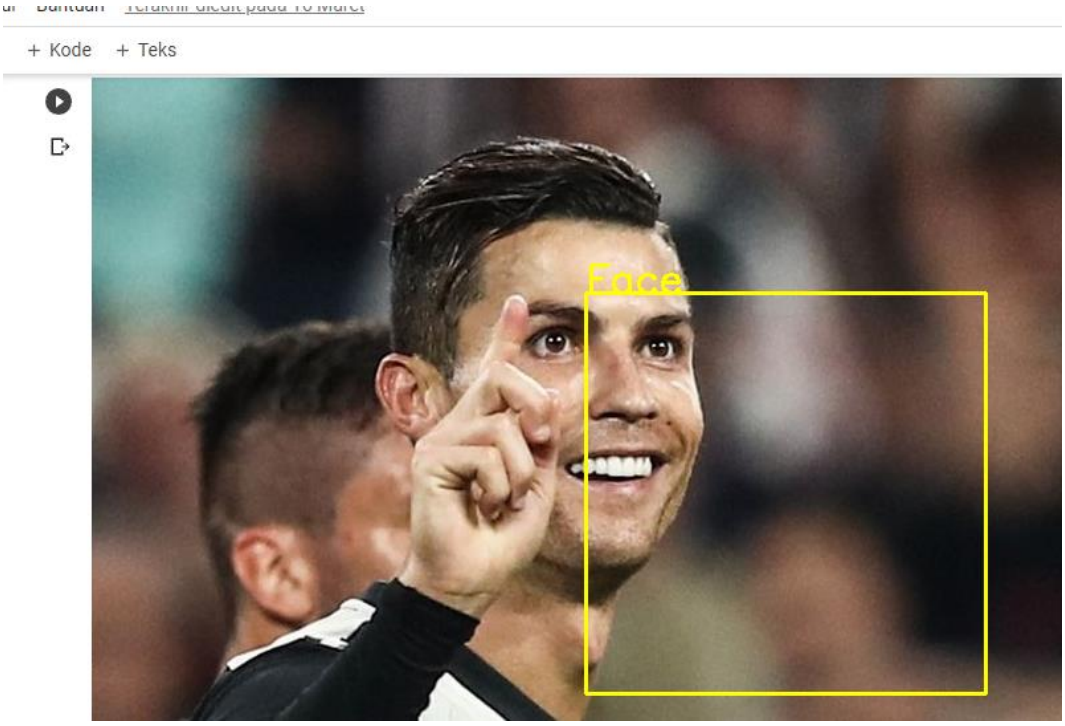
## D. LATIHAN

### Hasil Praktikum setelah dijalankan

#### Deteksi wajah




#### Deteksi Senyuman




1. **Deteksi Wajah Menggunakan Citra Digital**
   - Buka Jupyter Notebook / Jupyter Lab
   - Rename : **face_detector.ipynb**
   - Masukan Library yang digunakan

     ```
     import cv2
     ```

     Library diatas digunakan untuk memanggil fungsi yang terdapat pada OpenCV
   - Buat objek untuk membaca citra digital yang dipakai sebagai inputan

     ```
     img = cv2.imread('mypicture.jpg')
     ```

     fungsi `imread()` berfungsi untuk load citra dan mengubahnya dalam pixel. Kemudian disimpan pada objek `img`.
   - Konversi citra tersebut kedalam Grayscale

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

     Ubah citra berwarna tersebut menjadi grayscale menggunakan fungsi `cvtColor()`. Flag COLOR_BGR2GRAY digunakan untuk mengubah citra berwarna RGB ke Grayscale.

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi wajah

```
haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Berkas haar casecade dapat diunduh pada tautan berikut : https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

- Menerapkan metode deteksi wajah pada citra Grayscale

```
faces_rect = haar_cascade.detectMultiScale(gray, 1.1, 9)
```

Fungsi `detectMultiScale()` berfungsi untuk mendeteksi objek yang memiliki ukuran yang berbeda dengan data yang telah di-*training* dalam *classifier*. Obyek kita adalah gambar. Pada parameter *scaleFactor* (1.1), proses kinerja pendeteksi gambar terjadi. Jika ukuran gambar diperkecil akan mempercepat kinerja deteksi, tetapi akurasi akan semakin berkurang. Sedangkan parameter *minNeighbors* (9) akan mendeteksi nilai positif dalam *scaleFactor* yang akan mendeteksi wajah dalam gambar tersebut.

- Lakukan *looping* untuk menggambar persegi ketika mendeteksi wajah

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

Untuk menggambar persegi, digunakan fungsi `rectangle()`. Struktur dari fungsi rectangle adalah sebagai berikut :

- img = digunakan untuk menentukan dimana persegi akan diletakan
- x, y = koordinat sudut kiri atas persegi
- x+w, yh = koordinat sudut kanan bawah persegi
- (0, 255, 0) = nilai antara 0 – 255 untuk membuat warna, BGR merupakan kembalikan dari RGB sehingga perlu membiasakan diri

- 2 = mengatur ketebalan garis penyusun persegi

- Tampilkan jendela yang menampilkan citra wajah, dan tambahkan fungsi untuk menutup jendela tersebut

```
cv2.imshow('Detected faces', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fungsi `imshow()` digunakan untuk menampilkan citra hasil pemrosesan. Fungsi `waitKey(0)` digunakan untuk menunggu perintah selanjutnya, dengan menggunakan sinyal keyboard. Fungsi `destroyAllWindows()` digunakan untuk menutup semua jendela yang aktif.

## 2. Deteksi Wajah dan Senyum Menggunakan Citra Digital

- Buka Jupyter Notebook / Jupyter Lab
- Rename : **smile_detector.ipynb**
- Masukan Library yang digunakan

```
import cv2
```

Library diatas digunakan untuk memanggil fungsi yang terdapat pada OpenCV

- Buat objek untuk membaca citra digital yang dipakai sebagai inputan

```
img = cv2.imread('mypicture.jpg')
```

fungsi `imread()` berfungsi untuk load citra dan mengubahnya dalam pixel. Kemudian disimpan pada objek `img`.

- Konversi citra tersebut kedalam Grayscale

```
gray_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

Ubah citra berwarna tersebut menjadi grayscale menggunakan fungsi `cvtColor()`. Flag COLOR_BGR2GRAY digunakan untuk mengubah citra berwarna RGB ke Grayscale.

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi wajah

```
face_cascade = cv2.CascadeClassifier('haarc
ascade_frontalface_default.xml')
```

Berkas haar casecade dapat diunduh pada tautan berikut :

https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi senyum

```
smile_cascade = cv2.CascadeClassifier('haar
cascade_smile.xml')
```

Berkas haar casecade dapat diunduh pada tautan berikut :

https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_smile.xml

- Menerapkan metode deteksi wajah pada citra Grayscale

```
faces = face_cascade.detectMultiScale(gray,
1.1, 9)
print('Number of detected faces:', len(face
s))
```

Fungsi `detectMultiScale()` berfungsi untuk mendeteksi objek yang memiliki ukuran yang berbeda dengan data yang telah di-*training* dalam *classifier*. Obyek kita adalah gambar. Pada parameter *scaleFactor* (1.1), proses kinerja pendeteksi gambar terjadi. Jika ukuran gambar diperkecil akan mempercepat kinerja deteksi, tetapi akurasi akan semakin berkurang. Sedangkan parameter *minNeighbors* (9) akan mendeteksi nilai positif dalam *scaleFactor* yang akan mendeteksi wajah dalam gambar tersebut. Fungsi `print()` digunakan untuk mentak jumlah wajah yang terdeteksi oleh sistem.

- Lakukan *looping* untuk menggambar persegi ketika mendeteksi wajah dan senyuman.

```
for (x,y,w,h) in faces:
    # Menggambar persegi pada wajah
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
    cv2.putText(img, "Face", (x, y), cv2.FONT_HERSHEY_
SIMPLEX, 1, (0, 255, 255), 2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    # mendeteksi senyum pada ROI wajah
    smiles = smile_cascade.detectMultiScale(roi_gray,
1.8, 20)
    if len(smiles) > 0:
        print("smile detected")
        for (sx, sy, sw, sh) in smiles:
            cv2.rectangle(roi_color, (sx, sy), ((sx + sw
), (sy + sh)), (0, 0, 255), 2)
            cv2.putText(roi_color, "smile", (sx, sy),
            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    else:
        print("smile not detected")
```

- Tampilkan jendela yang menampilkan citra wajah, dan tambahkan fungsi untuk menutup jendela tersebut

```
cv2.imshow('Detected faces', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fungsi `imshow()` digunakan untuk menampilkan citra hasil pemrosesan. Fungsi `waitKey(0)` digunakan untuk menunggu perintah selanjutnya, dengan menggunakan sinyal keyboard. Fungsi `destroyAllWindows()` digunakan untuk menutup semua jendela yang aktif.

### 3. Deteksi Wajah Menggunakan Webcam

- Buka Jupyter Notebook / Jupyter Lab
- Rename : **face_detector_webcam.ipynb**
- Masukan Library yang digunakan

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

Cv2 digunakan untuk memanggil fungsi-fungsi pada OpenCV. Sedangkan numpy digunakan untuk melakukan komputasi matriks.

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi wajah

```
faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Berkas haar casecade dapat diunduh pada tautan berikut : https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml.

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi wajah

```
camera = cv2.VideoCapture(0)
```

Fungsi `VideoCapture()` digunakan untuk meng-*capture* video secara real-time menggunakan kamera. Nilai 0 artinya kamera yang digunakan adalah kamera/webcam bawaan laptop.

- Lakukan *looping* untuk menggambar persegi ketika mendeteksi wajah

```
while(True):
    ret,img =camera.read()
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceDetect.detectMultiScale(gray,1.1,5)
    for (x,y,w,h) in faces:
       cv2.rectangle(img,(x,y), (x+w,y+h),(0,255,0),2)
    cv2.imshow("Face",img)
    if(cv2.waitKey(1) ==ord('q')):
        break
```

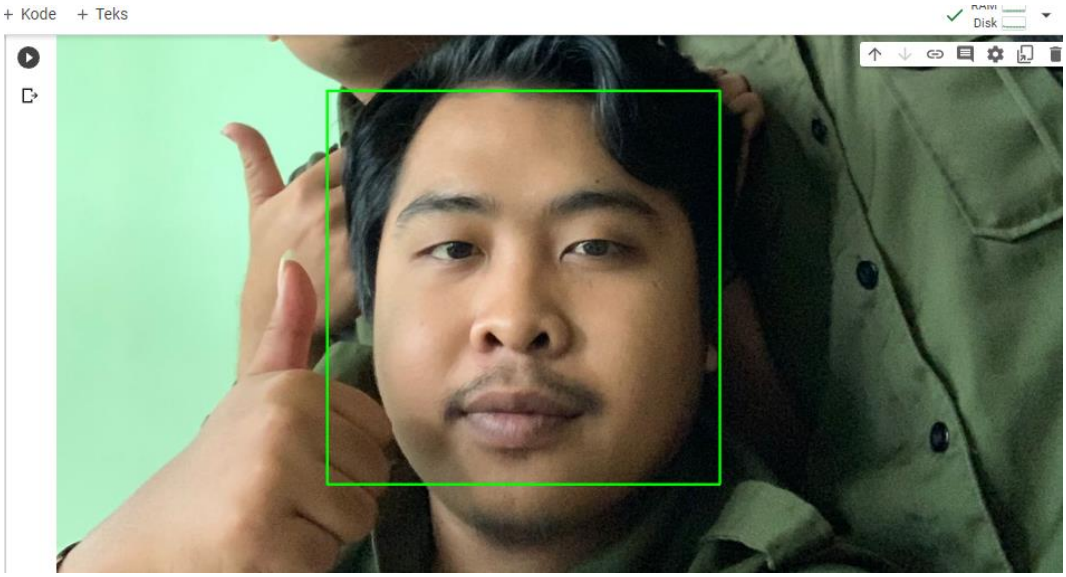
Persegi yang digambar pada wajah berwarna hijau dan memiliki ketebalan 2 pt. Untuk menutup jendela tekan tombol ‘q’ di keyboard.

- Buat objek untuk membaca haar cascade yang dapat digunakan untuk mendeteksi wajah

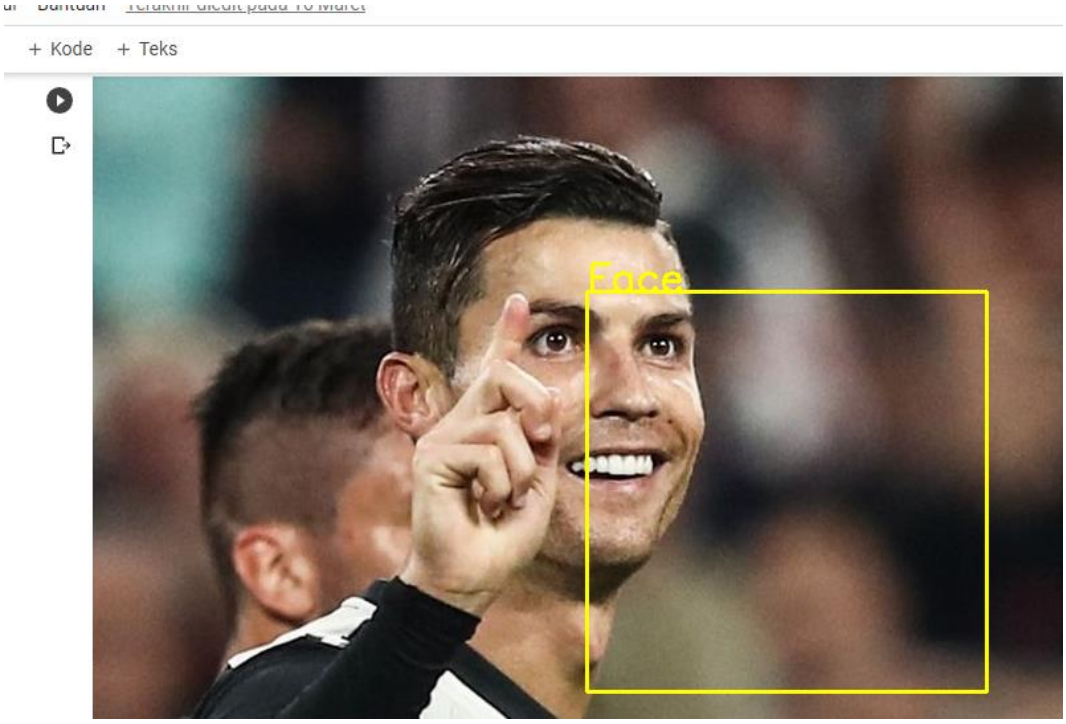
```
camera.release()
cv2.destroyAllWindows()
```

Fungsi `camera.release()` digunakan untuk melepaskan semua resource dari kemera. Fungsi `destroyAllWindows()` digunakan untuk menutup semua jendela yang aktif

**Deteksi wajah**



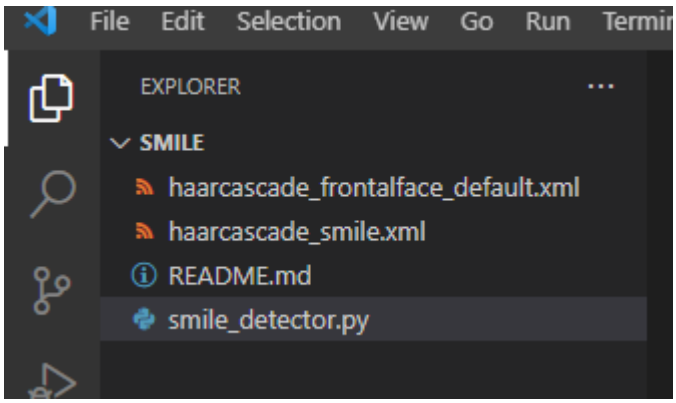

**Deteksi senyuman**



## E. TUGAS PRAKTIKUM

1. Buatlah sebuah program menggunakan python dan opencv yang digunakan untuk mendeteksi wajah dan senyuman secara *real-time* menggunakan webcam.

   Tampilan Folder, memanggil dua file dan membuat file dengan nama smile_detector.py




Source Code Tugas Praktikum

```
import numpy as np
import cv2

faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smileCascade = cv2.CascadeClassifier('haarcascade_smile.xml')

cap = cv2.VideoCapture(0)

while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.3,
        minNeighbors=5,
        minSize=(30, 30)
    )

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]

        smile = smileCascade.detectMultiScale(
```

```
            roi_gray,
            scaleFactor= 1.5,
            minNeighbors=15,
            minSize=(25, 25),
            )

        for i in smile:
            if len(smile)>1:
                cv2.putText(img,"Smiling",(x,y-
30),cv2.FONT_HERSHEY_SIMPLEX,
                    2,(0,255,0),3,cv2.LINE_AA)

    cv2.imshow('video', img)
    k = cv2.waitKey(30) & 0xff
    if k == 27: # press 'ESC' to quit
        break

cap.release()
cv2.destroyAllWindows()
```

**Penjelasan Source Code**

Kode ini adalah program deteksi wajah dan senyum menggunakan OpenCV dan webcam. Pertama, program mengimpor pustaka yang diperlukan yaitu NumPy dan OpenCV. Kemudian, program memuat file XML untuk deteksi wajah dan senyum menggunakan cv2.CascadeClassifier().

Program membuka kamera menggunakan cv2.VideoCapture(0) dan memulai loop untuk membaca setiap frame dari kamera menggunakan cap.read(). Setiap frame kemudian dikonversi ke skala abu-abu menggunakan cv2.cvtColor().

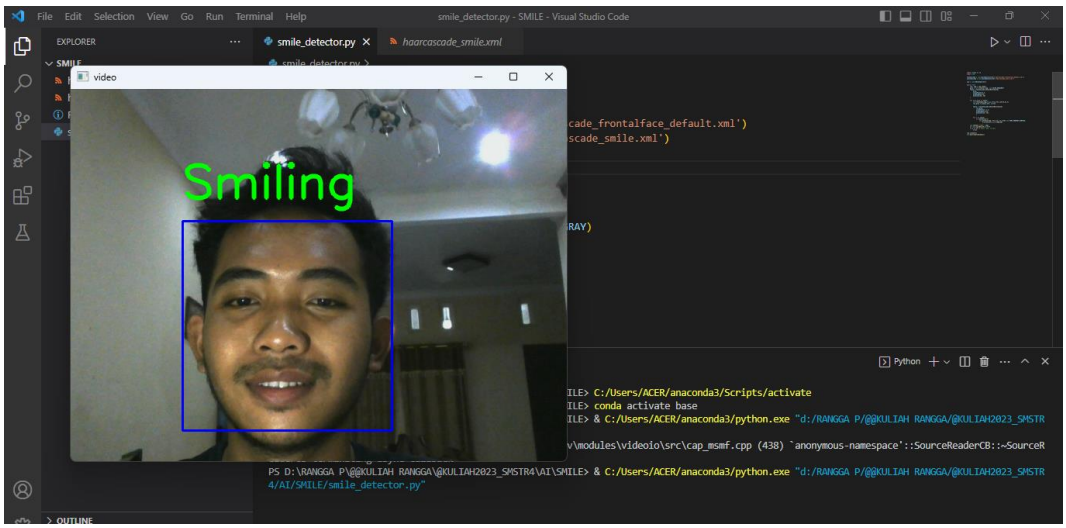
Untuk setiap frame, program mencari wajah dalam gambar menggunakan faceCascade.detectMultiScale() dan menggambar kotak di sekitar wajah yang terdeteksi menggunakan cv2.rectangle().

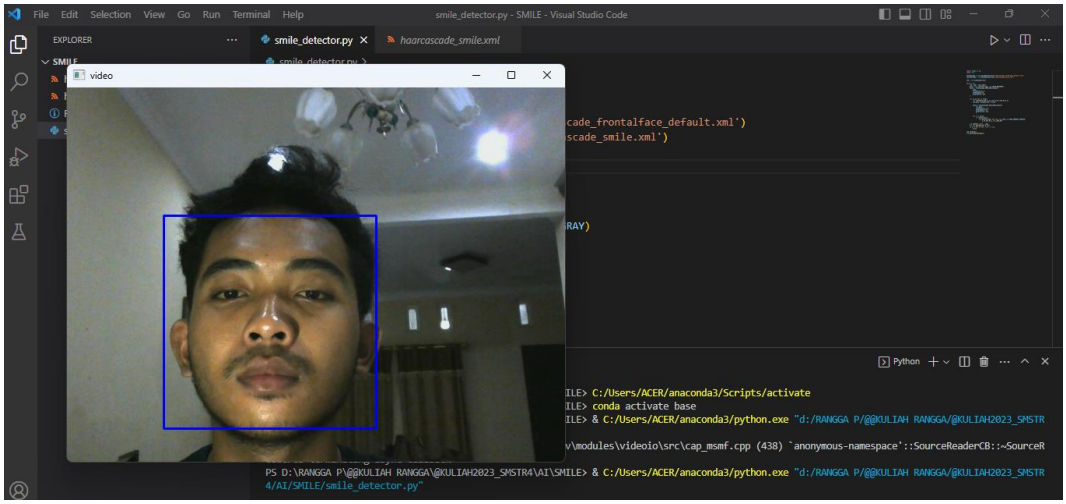
Setelah wajah terdeteksi, program mengambil ROI (region of interest) pada wajah menggunakan gray[y:y+h, x:x+w]. Kemudian, program mencari senyum dalam ROI menggunakan smileCascade.detectMultiScale().

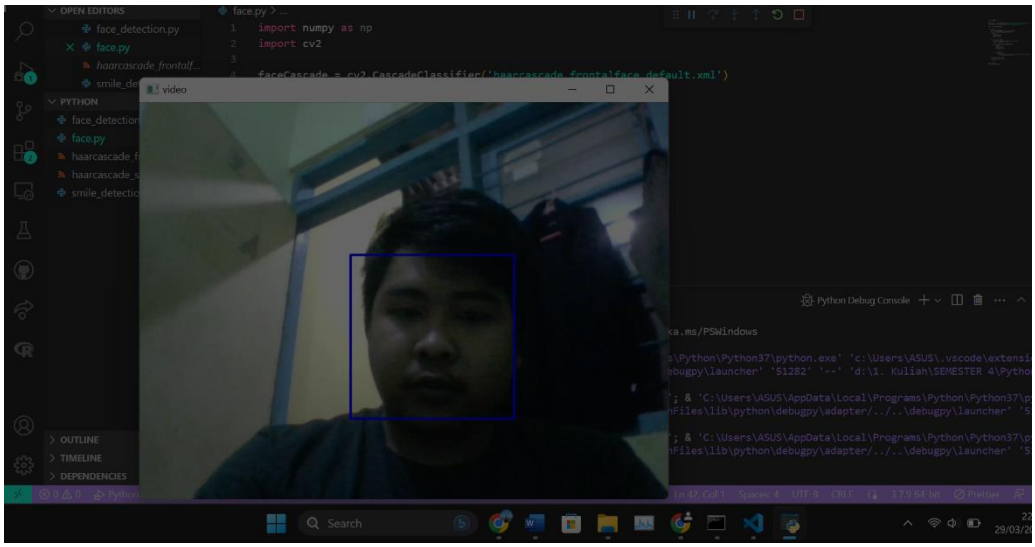
Jika program mendeteksi satu atau lebih senyum, maka akan menampilkan teks "Smiling" di atas kotak wajah menggunakan cv2.putText().

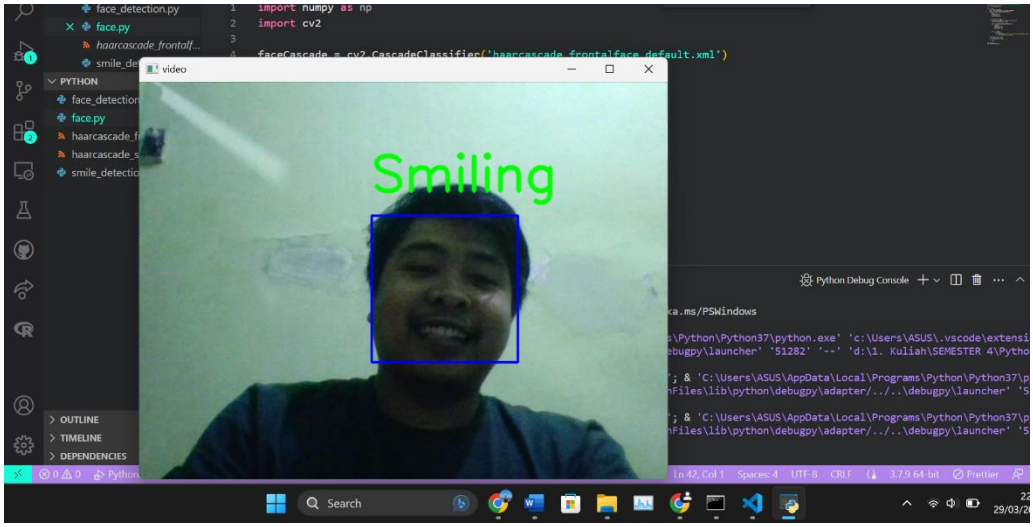
Akhirnya, program menampilkan frame yang dimodifikasi di jendela cv2.imshow(). Jika tombol ESC ditekan, program akan keluar dari loop dan menutup jendela menggunakan cap.release() dan cv2.destroyAllWindows().

Hasil Dokumentasi Praktikum








2. Jelaskan kelebihan dan kekurangan deteksi wajah menggunakan metode haar casecade ?

Deteksi wajah menggunakan metode Haar Cascade merupakan salah satu metode populer dalam pengolahan citra untuk mendeteksi keberadaan wajah pada sebuah gambar atau video. Metode ini memiliki beberapa kelebihan dan kekurangan, di antaranya:

Kelebihan:

1. Cepat dan efisien: Metode Haar Cascade dapat mendeteksi wajah dengan sangat cepat dan efisien, bahkan pada gambar atau video yang memiliki resolusi tinggi.
2. Akurasi yang cukup tinggi: Metode Haar Cascade mampu menghasilkan deteksi wajah yang cukup akurat, dengan tingkat kesalahan yang relatif rendah.
3. Dapat diimplementasikan pada berbagai platform: Metode ini dapat diimplementasikan pada berbagai platform, termasuk perangkat seluler dan sistem embedded.
4. Mudah diimplementasikan: Metode Haar Cascade dapat diimplementasikan dengan mudah menggunakan berbagai library atau framework yang tersedia, seperti OpenCV.

Kekurangan:

1. Sensitif terhadap pencahayaan: Metode Haar Cascade rentan terhadap perubahan pencahayaan pada gambar atau video, sehingga kemampuannya dalam mendeteksi wajah dapat berkurang.
2. Tidak mampu mendeteksi wajah dalam posisi ekstrim: Metode Haar Cascade tidak mampu mendeteksi wajah dengan baik dalam posisi ekstrim, seperti wajah yang condong atau wajah yang terpotong.

3. Sangat tergantung pada kualitas dataset: Metode ini sangat tergantung pada kualitas dataset yang digunakan untuk melatih model deteksi, sehingga kualitas deteksi dapat dipengaruhi oleh kualitas dataset yang digunakan.

## 1.4. PENUTUP

### 1. Kesimpulan

1. Kita Dapat memahami konsep kecerdasan buatan
2. kita Dapat mampu menjelaskan implementasi kecerdasan buatan
3. Kita Dapat mampu membangun proyek sederhana untuk implementasi kecerdasan buatan

### 2. Saran

Kita perlu terus berlatih menjalankan sorce code satu satu, dan melatih agar dapat menemukan code yang eror, dan open cv tidak hanya untuk wajah, namun dapat untuk pembacaan citra warna dll