GÖRÜNTÜ İŞLEMEYE GİRİŞ

PYTHON-OPENCV İLE YÜZ TANIMA

BATUHAN KÜÇÜKYILDIZ

İÇİNDEKİLER

GÖRÜNTÜ İŞLEMEYE GİRİŞ:

- **Görüntü işlemeye başlama süreci verilerin bilgisayar tarafından tanınmasıyla başlar.**

- **Görüntü formatındaki veri için öncelikle matris oluşturulur, resimdeki her veri değeri bu matris içinde işlenir.**

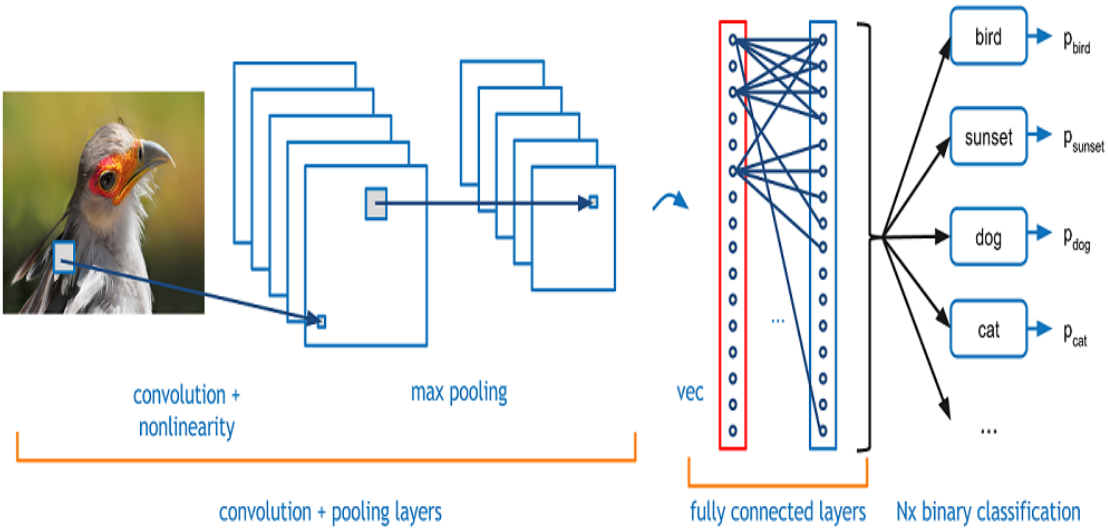
- **640*480 bir fotoğraf için 640*480' lik bir matris açılır kullandığımız resim RGB ise bu değer 640*480*3 olur.**

- **640*480 lik bir görüntüde bulanıklaştırma işlemi yapmak istesek bu işlemi program tüm matrisler üzerinde gezerek her birine tek tek yapacaktır.**

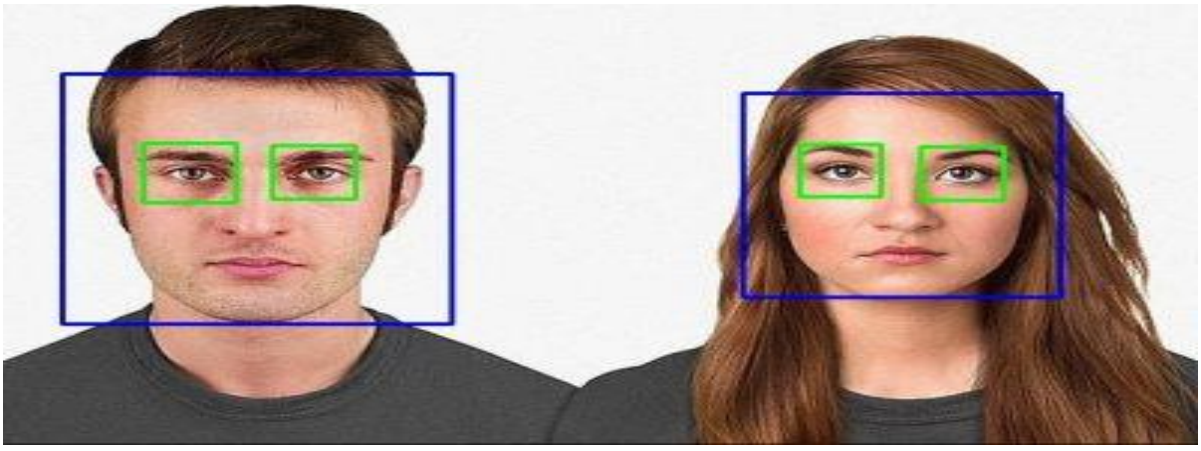
- **Görüntü işlemede kullanılan görüntünün RGB olması çok da önemli değildir hatta bazı durumlarda bu olay daha yavaş bir işleme yapılmasına yol açabilmektedir.**

- **Görüntü işlemenin en çok kullanıldığı derin öğrenme yapılarından birisi Evrişimsel Sinir Ağlarıdır.**

- **Bu model içerinde bulundurduğu Convolutional katmanı ile resim üzerinde eğitim için gerekli öznitelikleri belirler.**



- **Bazı durumlarda keskin hatlar yerine daha yuvarlak hatlar kutlanılırsa eğitim başarı oranının artmasını sağlayabiliriz.**



- **Bu tarz durumlar karşısında görüntü işleme teknikleri kullanılması gerekir.**

- **Histogram değerlerinin düzenlenip değerlendirilmesi ile zor olan nesnelerin daha kolay bir şekilde tespit edilmesi sağlanmaktadır.**

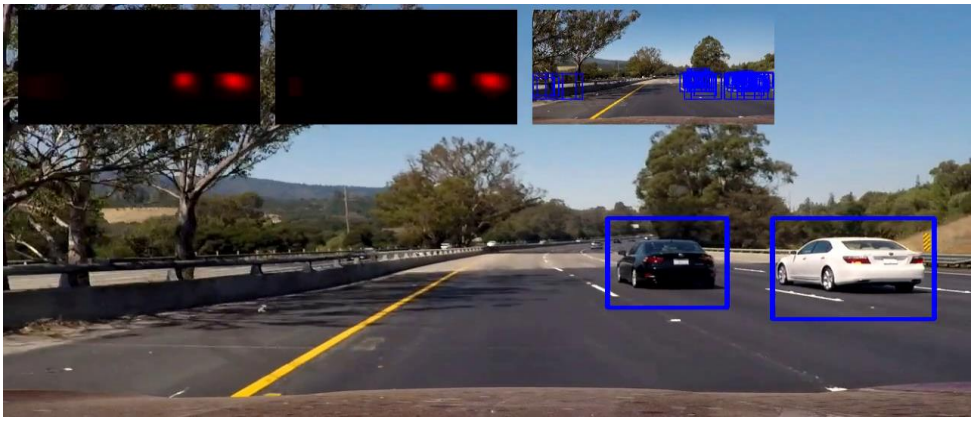
OpenCv Kütüphanesi ve Uygulamaları:

- **Görüntü işlemek isteyen kişilerin en çok başvurduğu kütüphanelerden birisi OpenCv ‘ dir.**

- **Dünya üzerinde çok yaygın kullanılan bu kütüphanenin en büyük kullanıcıları arasında Google, Microsoft, Yahoo gibi büyük şirketler gösterilebilir.**

- **OpenCv ‘yi C++, Python ve Matlab gibi birçok yazılım diliyle kullanabiliriz.**

- **OpenCv’ nin kullanım alanları araç plakalarını okumak yüz tanıma sistemlerinde kullanma ve birçok alanda kullanılma durumu vardır.**

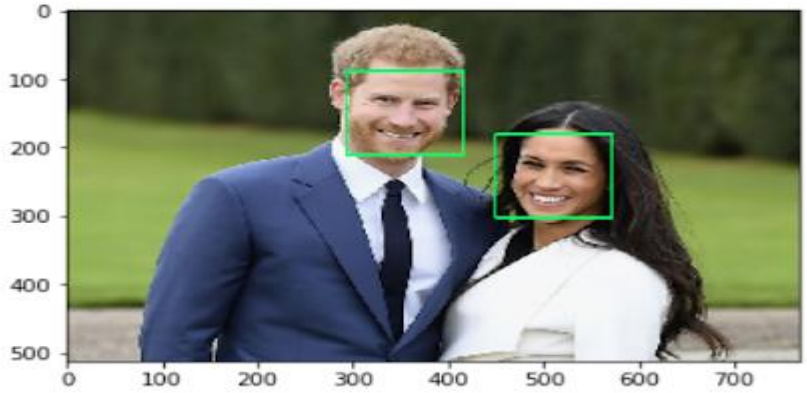


Python İle OpenCv' ye Giriş:

Yüz Algılama Programını Yapmak;

- **Yüz tanıma sistemi yapmak için en temel görev programın Yüz Algılama kısmını yazmaktır.**

```
Yüz Tespiti:  2
Out[17]: <matplotlib.image.AxesImage at 0x3f82e18b38>
```



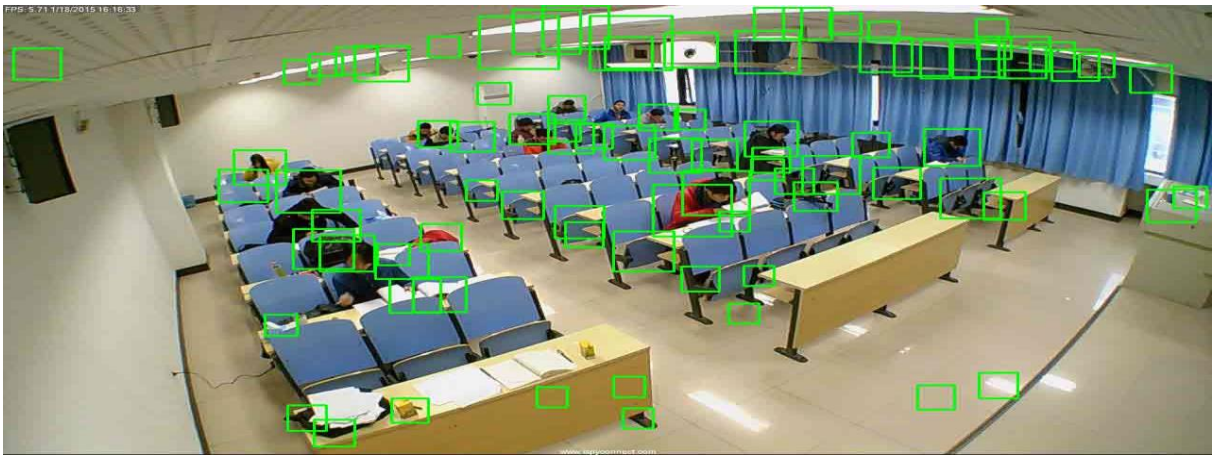
- **Bir yüzü veya istediğiniz bir nesneyi yakalamak için kullanılan yollardan en yaygın olanı "Haar –Cascade Detection" dur.**

- **Haar Cascade nesne algılama özelliği 2001 yılında Paul Viola ve Micheal Jones tarafından bulunmuştur.**

Haar Cascade Sınıflandırıcısı Nedir?

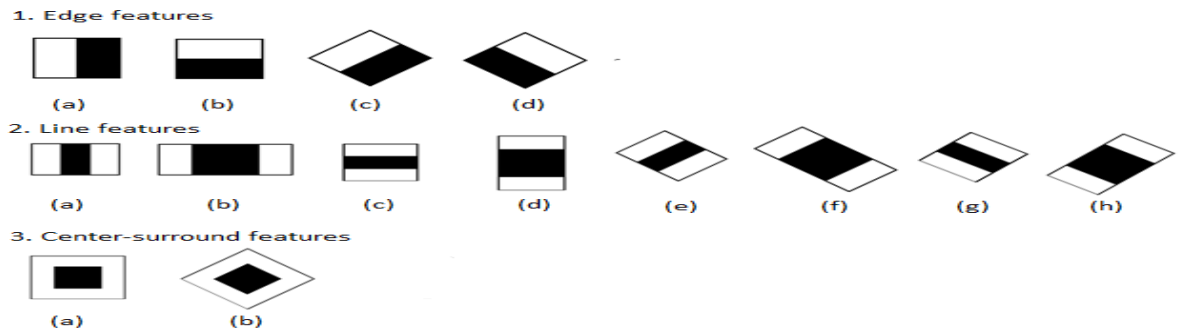
- **OpenCv içerisinde bulabileceğimiz haarcascade sınıflandırıcısı Paul ve Micheal tarafından bulunan nesne bulma yapısı olarak da bilinir.**

- **En temel manada belirli bir algoritma göre bulunması gereken nesneler önce bilgisayara tanıtılır ve daha sonra ona benzer şekillerin bulunduğu resimler veya video sahneleri taranarak o nesne bulunmaya çalışılır.**



- **Eğitim için içerisinde aranılan nesnenin bulunduğu pozitif resimlere ve aynı zaman da negatif resimlere ihtiyaç duyar.**

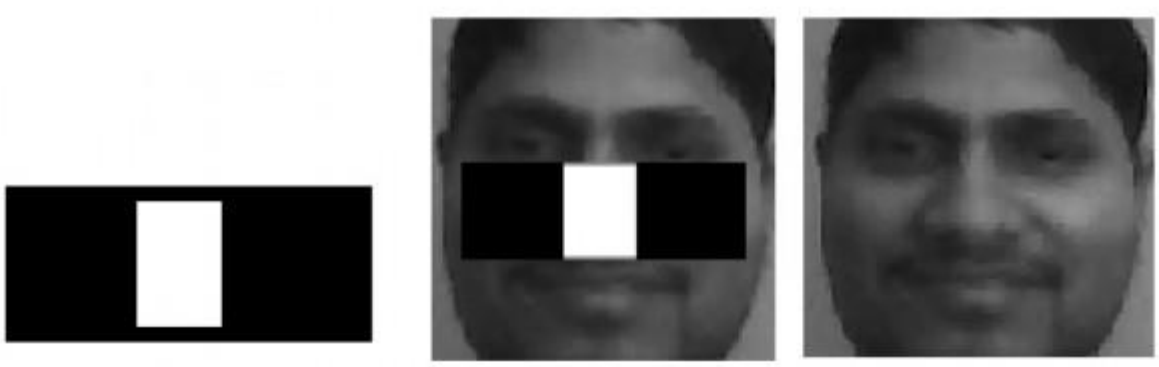
- **Sınıflandırıcı eğitimde pozitif resimlerdeki nesneleri ağadaki görseldeki gibi belirli çerçevelere ayırarak belirli hedefler bulmaya çalışır ve değerler oluşturur.**



- **Feature denilen bu çerçevelere zayıf sınıflandırıcılar denmektedir, çünkü tek başlarına doğru ve güçlü bir sınıflandırıcı olmazlar.**

- **Sınıflandırıcı temek olarak şöyle çalışmaktadır:**

- **Çerceveler aşağıdaki gibi örnek pozitif resimler üzerinde taranır.**



- **Yukarıdaki çerçeve için yanakların parlaklık oranının burun bölgesindeki parlaklık oranından daha düşük olmasından burun kısmı bu yöntemle seçilebilir.**

- **Aynı zaman da yukarıdaki görselde ise burun dışında kalan kısımlar bu yöntemle seçilir.**

Yüz Algılama Sistemine Başlangıç:

Yüz Tanıtma;

- **Programda OpenCv kütüphanesi kullanacağımız için negatif ve pozitif yönleri çıkarmamıza gerek yoktur OpenCv bunu otomatik olarak gerçekleştirir.**

- **Yüz tanıma sistemi yapmak için hazır sınıflandırıcı kullanacaksak sisteme XML dosyaları entegre etmek gerekir.**

```
CascadeClassifier('C:\\opencv\\build\\etc\\haarcascades\\haarcascade_frontalface_default.xml')
```

- **Yukarıda gördüğümüz kod bloğu sınıflandırıcımızı tanıttığımız bölümdür.**

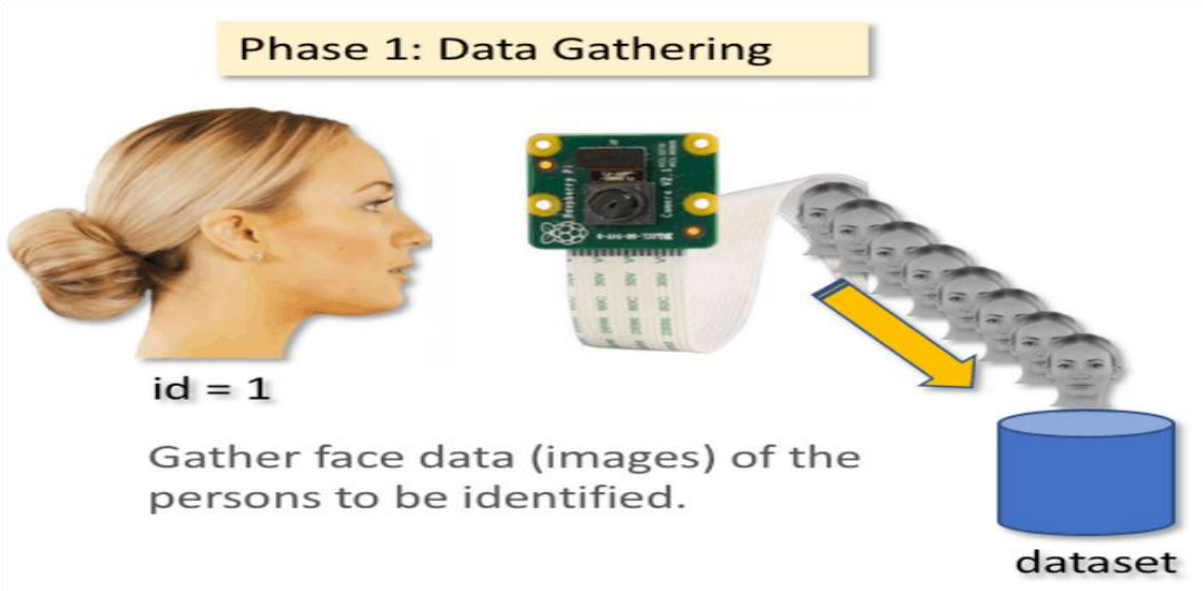
- **Bu işlemden sonra kameramızı ayarlayacağımız ve döngünün içerisinde karelerin gri mod da gözükmesini sağlayacağız.**

```
faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    )
```

- **Gray Komutu: Resimi gri tona çeviren komuttur.**
  Scale Factor: Her bir görüntünün ne kadar küçüleceğini belirten fonksiyondur.
  minNeighbors: Her bir kişinin dikdörtgen içerisinde gözükmesini sağlar.
  minSize: Yüzün etrafındaki dikdörtgenin minimum boyutudur.

Veri Toplama Adımı:

- **Projenin ilk aşamasında tanıttığımız yüzü bu sefer ise her kimlik için tek tek arşivlememiz gerekmektedir.**



- **İlk önce dizinimize OpenCv kütüphanemizi çağırıyoruz.**

```
import cv2
```

- **Daha sonra ise kameramızı tanıtıp XML dosyamızı ekliyoruz.**

```
kamera = cv2.VideoCapture(0)
kamera.set(3, 640)
kamera.set(4, 480)
face_detector = cv2.CascadeClassifier('C:\\opencv\\build\\etc\\haarcascades\\haarcascade_frontalface_default.xml')
```

- **Data dosyamızda kaç adet fotoğrafımızın saklanacağı kodu belirleyip kaç kişinin saklanacağı kod bloğunu dizinimize yazıyoruz.**

```
MAXFOTOSAY = 50
face_id = 1
print("\n [INFO] Kayıtlar başlıyor. Kameraya poz ver :)")

say = 0
```

```
while(True):
    ret, img = kamera.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    yuzler = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in yuzler:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        say += 1
        # Yakalanan imajı veriseti klasörüne kaydet
        cv2.imwrite("veriseti1/" + str(face_id) + '.' + str(say) + ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('imaj', img)
        print("Kayıt no: ",say)
    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif say >= MAXFOTOSAY:
         break
print("\n [INFO] Program sonlanıyor ve bellek temizleniyor.")
kamera.release()
```

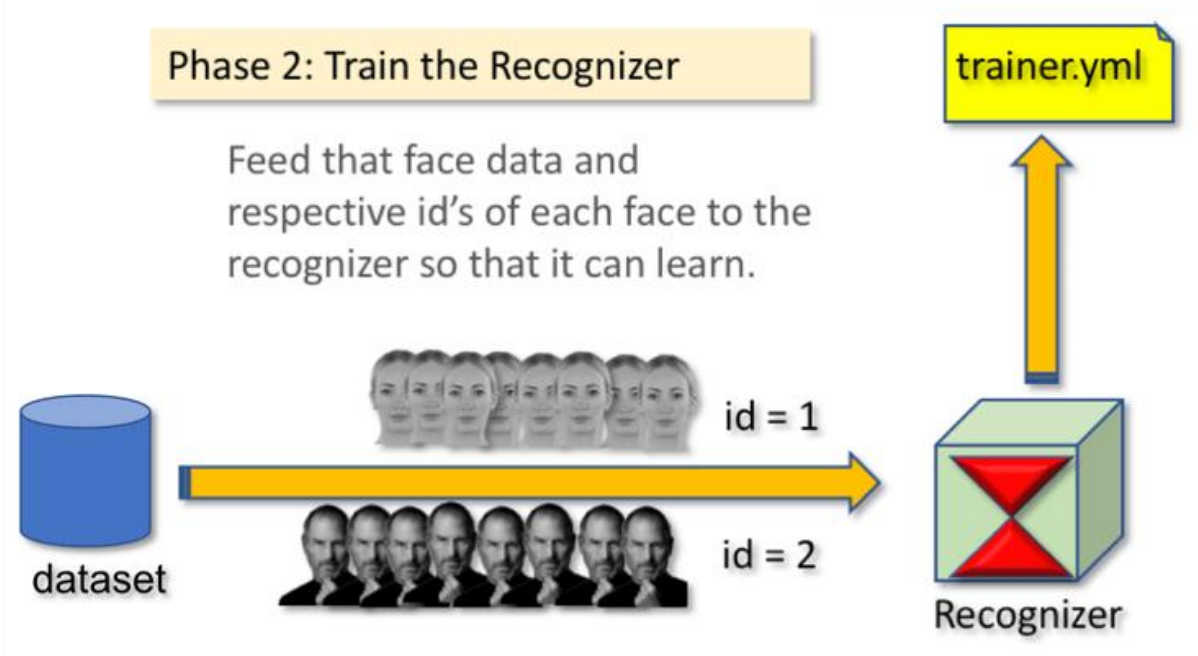
- **Yukarıdaki kod bloğunda ise kameranın konumları ayarlanmış olup verileri hangi klasöre ne tür uzantında olacağına ve boyutlarına karar verilir.**

- **Bu işlemlerden sonra elimizde olması gereken görseller şu şekilde olması gerekir:**

- **Görseller belirlediğimiz klasörlerde 1.1, 1.2,1.3… sırasıyla kodlanırlar.**

Yüz Eğitimi Adımı:

- **Oluşturduğumuz bu verileri veri setimizden alıp OpenCv' yi eğitmeliyiz. Bu tamamen OpenCv içerinde yer alan bir özelliktir.**



- **Bulduğu sonuçları kaydetmesi için bir klasör açıp içerisine eğitmen.yml adlı bir dosya açmamız gerekir.**

- **Bu eğitim için " Pillow" , "Numpy" ve "OpenCv" gerekir. Sınıflandırıcılarımızı ilk önce kod dosyamıza eklememiz gerekir.**

```
import os
import cv2
import numpy as np
from PIL import Image
```

- **Kütüphaneleri çağırdıktan sonra görselleri hangi klasöre kaydettiysek bu klasörü programa tanıtmamız gerekir.**

```
path = 'veriseti1'
cv2.face.LBPHFaceRecognizer_create()
```

- **Yukarıdaki kod bloğunda aynı zamanda bir tanıyıcı olan LBPH(LOCAL BINARY PATTERNS HISTOGRAMS) kodunu tanıtmış olduk.**

```
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    ornekler=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')    # gri
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[0])
```

- **Bu blokta ise dizideki tüm fotoğrafları çekmiş olup kimlikler ve yüzler olarak eğitmiş olacağız.**

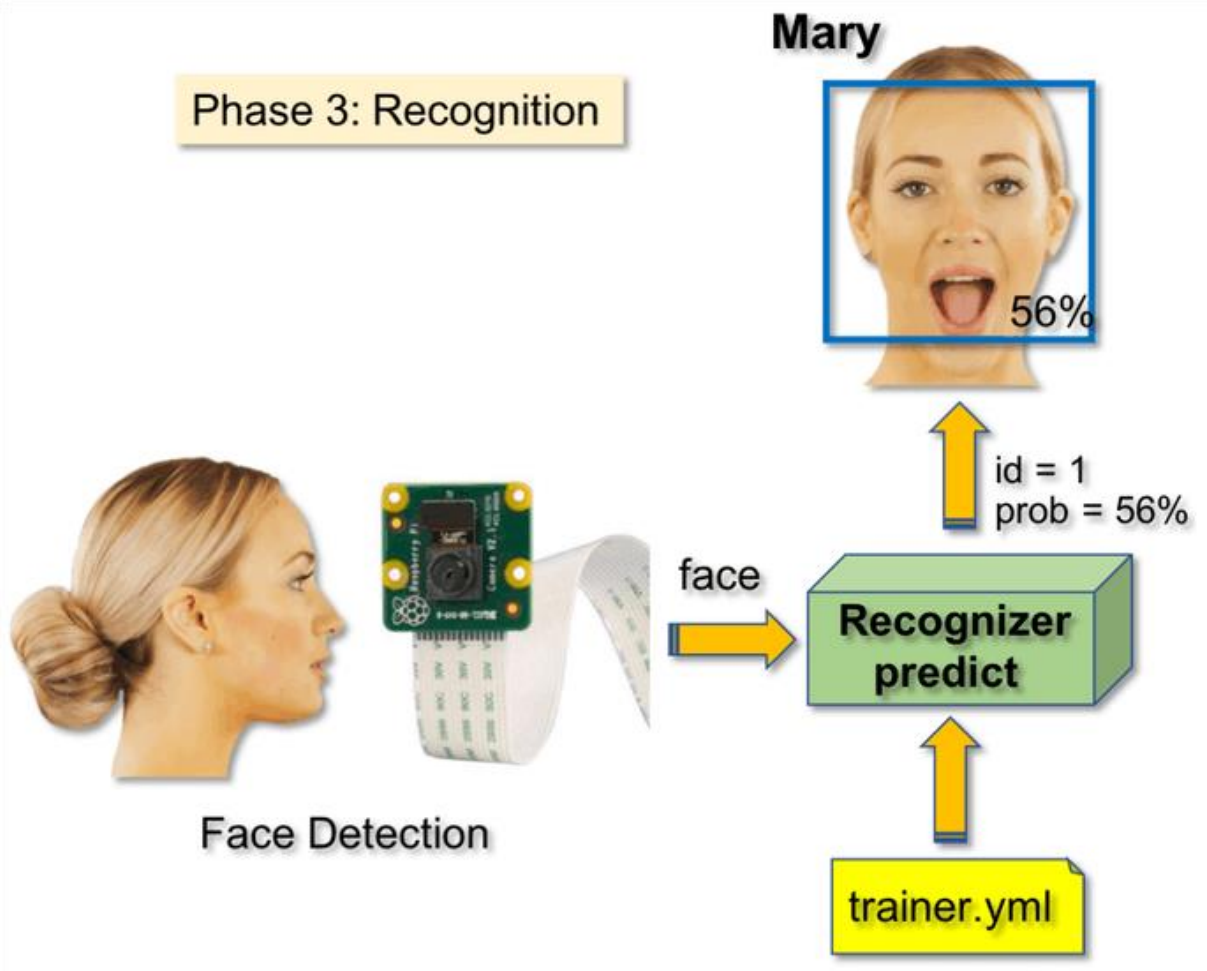
- **Sonuç olarak "eğitim.yml" dosyasına eğitmiş olduğumuz yüzler kaydedilmiş olacaktır.**

- **Eğitim yapmak için Python dosyası ile .yml uzantılı dosyanın aynı klasör içerisinde yer almasına dikkat etmek gerekmektedir.**

Tanıyıcı Adımı:

- **Projemizin son adımı olan tanıyıcı kısmını çalıştırdığımız zaman kamerada yeni bir yüz yakalama işlemi gerçekleştireceğiz.**

- **Eğer yakaladığımız bu yüz daha önceden kaydedilmiş ve eğitilmiş bir yüz ise sistem bu yüzü tanıyacak ve daha sonrada bir uyumluluk yüzdesi çıkartacaktır.**



- **Eğittiğimiz yüzleri kullanıcıya göstermek için dizine kullanıcın isimlerini önceden girmemiz gerekir.**

```
names = ['None', 'Batuhan','Umut']
```

- **Bu kısımda ID numarasına bakan program sırayla numaralara girdiğimiz isimleri birbirleriyle eşleştirecektir.**

- **Daha sonra ilk başta yaptığımız gibi yüz tanıma kodumuzla kişinin yüzünü tespit edeceğiz.**

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('egitim/egitim.yml')
cascadePath = "C:\\opencv\\build\\etc\\haarcascades\\haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
names = ['None', 'Batuhan','Umut']


kamera = cv2.VideoCapture(0)
kamera.set(3, 1000)
kamera.set(4, 800)

minW = 0.1 * kamera.get(3)
minH = 0.1 * kamera.get(4)
```

- **Tanınan yüzün üstünde kişinin adı yazacağı için bu yazının stilini programa .ttf dosyası olarak tanıtmamız gerek.**

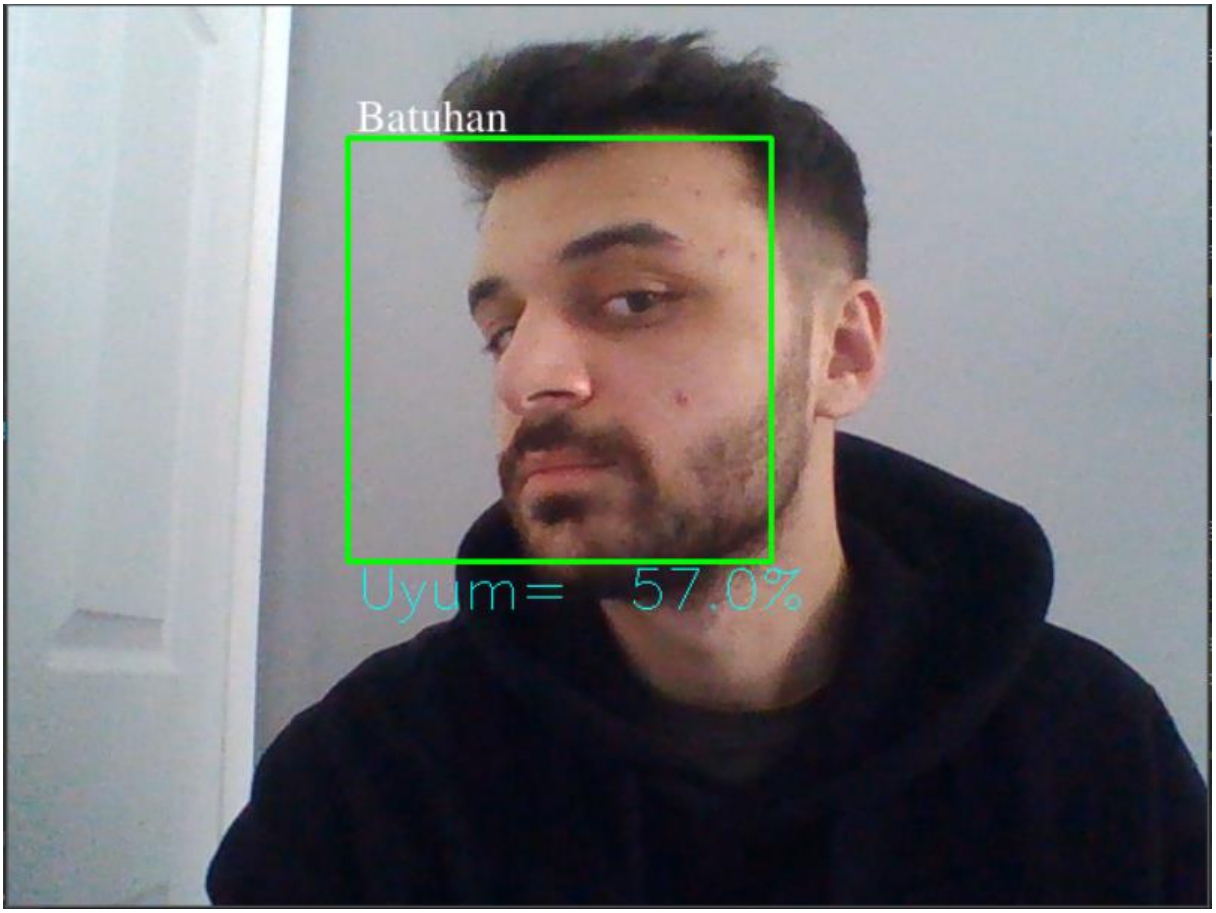
```
def print_utf8_text(image, xy, text, color):
    fontName = 'FreeSerif.ttf'
    font = ImageFont.truetype(fontName, 24)
    img_pil = Image.fromarray(image)
```

- **Ekranda gözüken kişiyle eğitilmiş olan kişi birbiriyle eşleşiyorsa program bir uyumluluk yüzdesi çıkarır ve bunu ekranda bize gösterir.**

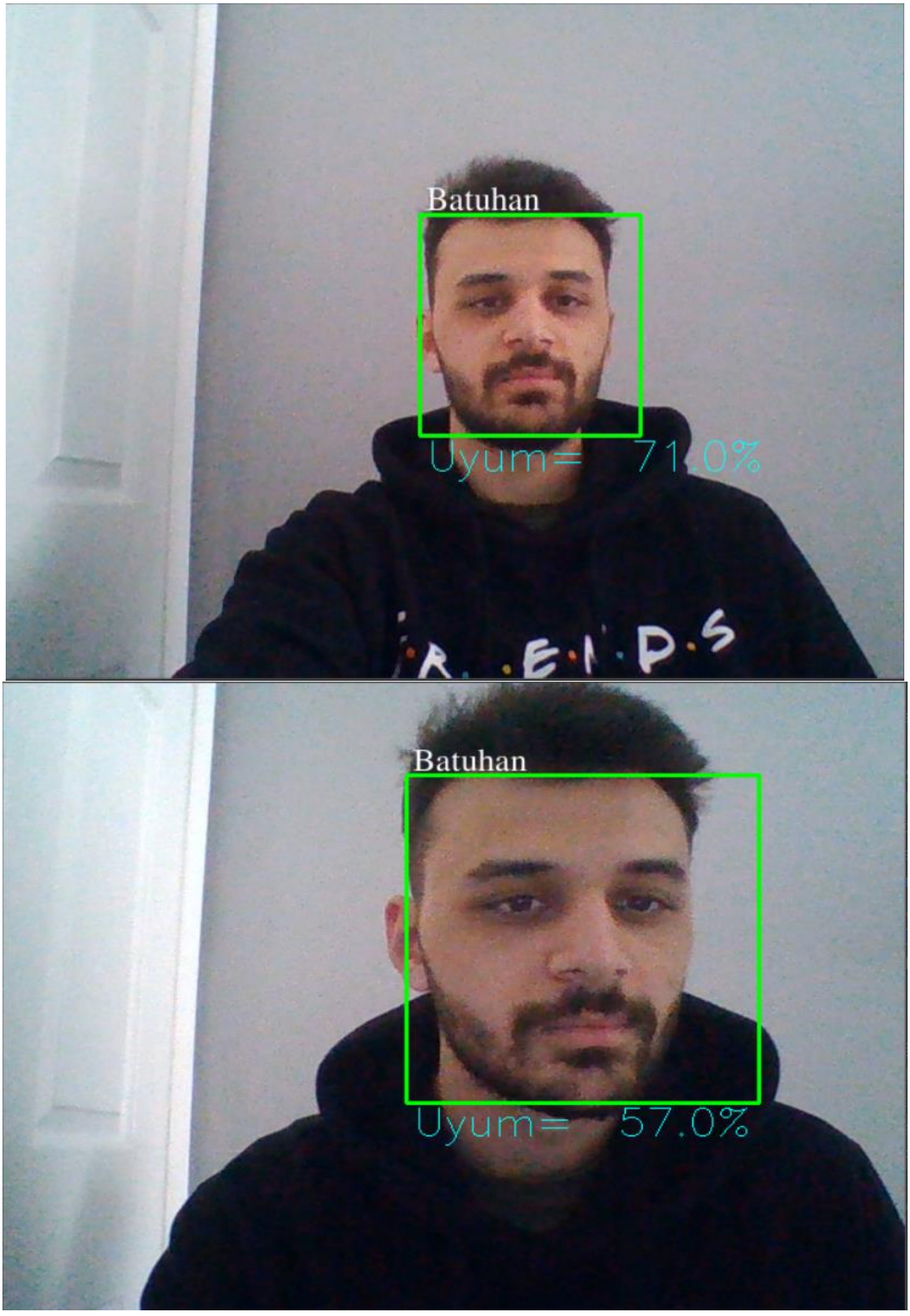
```
for (x, y, w, h) in yuzler:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    id, uyum = recognizer.predict(gri[y:y + h, x:x + w])

    if (uyum < 100):
        id = names[id]
        uyum = f"Uyum=  {round(uyum,0)}%"
    else:
        id = "bilinmiyor"
        uyum = f"Uyum=  {round(uyum,0)}%"
```

- **Bu yazdığımız kodla programa aslında olasılık hesabı hesaplatmış oluruz.**



Programın Çalışmasına Dair Görseller:



Program Dosyasına Ait Kodlar:

01_yuz_veriseti.py:

```
import cv2

kamera = cv2.VideoCapture(0)
kamera.set(3, 640)
kamera.set(4, 480)
face_detector =
cv2.CascadeClassifier('C:\\opencv\\build\\etc\\haarcascades\\haarcascade_fr
ontalface_default.xml')
MAXFOTOSAY = 50
face_id = 1
print("\n [INFO] Kayıtlar başlıyor. Kameraya poz ver :)")

say = 0

while(True):
    ret, img = kamera.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    yuzler = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in yuzler:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        say += 1
        # Yakalanan imajı veriseti klasörüne kaydet
        cv2.imwrite("veriseti1/" + str(face_id) + '.' + str(say) + ".jpg",
gray[y:y+h,x:x+w])
        cv2.imshow('imaj', img)
        print("Kayıt no: ",say)
    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif say >= MAXFOTOSAY:
         break
print("\n [INFO] Program sonlanıyor ve bellek temizleniyor.")
kamera.release()
cv2.destroyAllWindows()
```

02_yuz_egitimi:

```
import os
import cv2
import numpy as np
from PIL import Image


path = 'veriseti1'
cv2.face.LBPHFaceRecognizer_create()
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector =
cv2.CascadeClassifier("C:\\opencv\\build\\etc\\haarcascades\\haarcascade_fr
ontalface_default.xml")


def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    ornekler=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')    # gri
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[0])
        yuzler = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in yuzler:
            ornekler.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return ornekler,ids
print ("\n [INFO] yuzler eğitiliyor. Birkaç saniye bekleyin ...")
yuzler,ids = getImagesAndLabels(path)
recognizer.train(yuzler, np.array(ids))
recognizer.write('egitim/egitim.yml')
print(f"\n [INFO] {len(np.unique(ids))} yüz eğitildi. Betik
sonlandırılıyor.")

# print(yuzler)
```

03_yuz_tanıma:

```
import cv2
import numpy as np
from PIL import Image, ImageDraw, ImageFont

def print_utf8_text(image, xy, text, color):
    fontName = 'FreeSerif.ttf'
    font = ImageFont.truetype(fontName, 24)
    img_pil = Image.fromarray(image)
    draw = ImageDraw.Draw(img_pil)
    draw.text((xy[0],xy[1]), text, font=font,
              fill=(color[0], color[1], color[2], 0))  # b,g,r,a
    image = np.array(img_pil)
    return image

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('egitim/egitim.yml')
cascadePath =
"C:\\opencv\\build\\etc\\haarcascades\\haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
names = ['None', 'Batuhan','Umut']


kamera = cv2.VideoCapture(0)
kamera.set(3, 1000)
kamera.set(4, 800)

minW = 0.1 * kamera.get(3)
minH = 0.1 * kamera.get(4)
while True:
    ret, img = kamera.read()
    gri = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    yuzler = faceCascade.detectMultiScale(
        gri,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(int(minW), int(minH)),
    )
    for (x, y, w, h) in yuzler:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        id, uyum = recognizer.predict(gri[y:y + h, x:x + w])

        if (uyum < 100):
            id = names[id]
            uyum = f"Uyum=  {round(uyum,0)}%"
        else:
            id = "bilinmiyor"
            uyum = f"Uyum=  {round(uyum,0)}%"
```

```
  color = (255,255,255)
        img=print_utf8_text(img,(x + 5, y - 25),str(id),color) # Türkçe
karakterler
        # cv2.putText(img, str(id), (x + 5, y - 5), font, 1, (255, 255,
255), 2)
        cv2.putText(img, str(uyum), (x + 5, y + h + 25), font, 1, (255,
255, 0), 1)

    cv2.imshow('kamera', img)
    k = cv2.waitKey(10) & 0xff  # Çıkış için Esc veya q tuşu
    if k == 27 or k==ord('q'):
        break

print("\n [INFO] Programdan çıkıyor ve ortalığı temizliyorum")
kamera.release()
cv2.destroyAllWindows()
```

Kaynakça:

- P.Erbao, Z.Guotong, “Image Processing Technology Research of On-Line Thread Processing”, 2012 International Conference on Future Electrical Power and Energy System, April 2012.

- H.Singh, **Practical Machine Learning and Image Processing,** pp.63–88, January 2019.

- Q.R.Zhang, P.Peng, Y.M.Jin, “Cherry Picking Robot Vision Recognition System Based on OpenCV”, MATEC Web of Conferences, January 2016.

- Y.Xu, L.Zhang, “Research on Lane Detection Technology Based on OPENCV”, Conference: 2015 3rd International Conference on Mechanical Engineering and Intelligent Systems, January 2015.

- https://www.superdatascience.com/blogs/opencv-face-recognition
- https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826
- https://medium.com/@adem.akdogan/opencv-k%C3%BCt%C3%BCphanesi-ile-g%C3%B6r%C3%BCnt%C3%BC-i%CC%87%C5%9Fleme-uygulamal%C4%B1-af50033f7d8
- http://python.gurmezin.com/python-ve-opencv-ile-yuz-tanima/
- https://sogrekci.com/blog/pythonda-opencv-ile-yuz-tanima/
- https://jn7.net/python-ve-opencv-kullanarak-yuz-tanima-uygulamasi/