

FIRAT ÜNİVERSİTESİ

Mühendislik Fakültesi

Elektrik-Elektronik Mühendisliği Bölümü

Biyometrik veri (iris) ile kişi tanıma

Öğrenci : Yunus YILMAZ 15220055

Öğrenci : Burak ELHAMAN 16220014

Danışman Öğretmen : Dr. Öğr. Üyesi Duygu KAYA

**GİRİŞ**

İlk olarak yapılan çalışmaları inceledik, Raspberry Pi ile başladık, elimizdeki Raspberry Pi 2B'de yapabileceklerimizin kısıtlı ve daha çok uğraştırıcı olduğunu gördük, bilgisayar üzerinden PyCharm IDE ile kütüphanelerin daha etkin olacağını anladık. Kütüphaneleri ekledik ve yüzün ve göz çevresinin tanıma işlemini yaptık.

İris kısmının tanıma işlemini yapacağız, çıkan sonucu gerekli filtrelemelerden geçireceğiz.

* Literatür Araştırması
* Uygun Çalışma ortamının belirlenmesi(Raspberry Pi/Python/Arduino/Matlab)
* Gerekli malzemelerin belirlenmesi ve temini
* Python da gerekli kütüphanelerin belirlenmesi ve eklenmesi
* İris tanımadan önce yüzün ve göz çevresinin belirlenmesi

Projenin Amacı

Bu projenin amacı İris tanıma olacaktır.

Kimlik doğrulayabilen tanıma ve tanımlama sistemidir.

Kullanılan geliştirme dili Python olacak, tanıma yapabilmek için derleyici de kullanılmalı ve göz görüntüsünü yakalamak için gerekli işlemleri Python sağlayacaktır.

Kullanılabilecek Alanlar

* Güvenli Geçiş Sistemleri
* Genel Sayım
* Banka Kasaları vs.

Projenin Detayları

Biyometrik Veri: Her insanın parmak izi ve göz gibi yaşamsal fizyolojik özelliği yüz, el, ses ve imza kişisel yaşamda önemli bir yere sahiptir. Karşı karşıya kalınan sorunların ve zayıflıkların çoğunun azaltılmasında bu yöntemlerle şifreleri kullanarak kimliğin (kişilerin) doğrulanması amaçlanmaktadır. Elde edilen yüksek güvenlik derecesine rağmen bu tekniklerle doğruluk tanıma sistemleri henüz % 100'e ulaşamadı. Bunun için özellik çıkarma ve görüntü tanıma için yeni tekniklerin tasarımının oluşturulması gerekmektedir.

Biyometrik Sınıflandırma

* Parmak İzi Tanıma
* Yüz Tanıma
* Ses Tanıma
* Retina Tanıma
* El Geometrisi
* İmza Dinamikleri
* İris Tanıma

İris Tanıma

İris, gözün iyi korunmuş bir parçasıdır. Kendi kendine oluşturulmuş benzersiz modeli harici olarak görülebilir ve yetişkin yaşamı boyunca sabit kalır. Bu anahtar faktörler İris'i tanımlamak için biyometrik olarak uygun hale getirir.

Görüntü işleme çerçeveleri benzersiz özellik olarak kullanılabilir ve desen çıkarma ile birlikte onu gözün dijital görüntüsünden biyometrik şablon oluşturulur, daha sonra veri tabanında saklanabilir. Bu biyometrik şablon fiziksel-matematiksel bir temsilini içerir, iriste depolanan benzersiz bilgiler ve modeller arasında yapılacak karşılaştırmalardan birisi aşağıdadır.

Bir müşteri ayırt edilmeyi ve tanımlanmayı tercih ettiğinde iris tanıma sistemi:

• Göz imajının elde edilmesi ve fotoğraflanması gerekiyor.

• Gözün iris bölgesi için bir şablon (biyometrik tanımlama) oluşturulur.

• Bu şablon, diğeri ile ilgili olarak incelenmiştir.

Karşılaştırma için veri tabanında depolanan şablonlar

ya eşleşen bir model bulundu ya da hiç eşleşme yok

olarak tespit edilir.

• Bir eşleşme tanınırsa, müşteri ilan edilir. Ardından

“tanımlandı ve kabul edildi” olarak bildirilir.

• Hiçbir eşleşme tanınmazsa, müşteri kalır

“tanımlanamayan ve anonim” olarak bildirilir.

Projenin Aşamaları

* Görüntü edinme
* Filtreleme
* Gözün bölümlere ayrılması
* Segmentasyon
* Normalleştirme
* Özellik çıkarma
* Karşılaştırma
* Değerlendirme

ÇIKTILAR

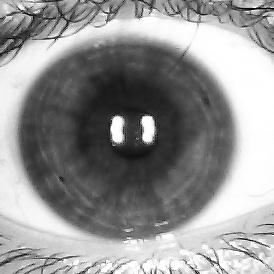
-Görüntü edinme



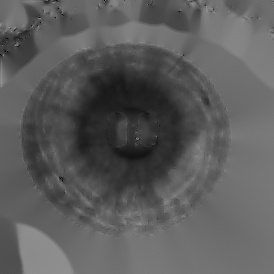
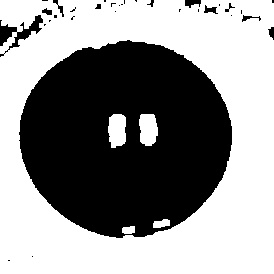
-Filtreleme



-Gözün bölümleri sayesinde fotoğrafın küçültülmesi



-Ayrılan bölümlere tekrar filtre uygulanması



-Normalleştirme



Python Kod

import cv2

import numpy as np

#resim = cv2.imread("foto2.jpg")

#output = resim.copy()

cap = cv2.VideoCapture(1)

def nothing(x):

pass

cv2.namedWindow("Frame")

cv2.createTrackbar("Alt", "Frame", 0, 255, nothing)

cv2.createTrackbar("Ust", "Frame", 0, 255, nothing)

cv2.createTrackbar("CannyAlt", "Frame", 0, 255, nothing)

cv2.createTrackbar("CannyUst", "Frame", 0, 255, nothing)

#gray = cv2.cvtColor(resim, cv2.COLOR\_BGR2GRAY)

#gray = cv2.GaussianBlur(gray, (7, 7), 0)

def processing(image\_path, r):

image\_path = cv2.resize(image\_path, (640, 480), interpolation=cv2.INTER\_LINEAR) #boyutu 640x480 yapar

image\_path = cv2.cvtColor(image\_path, cv2.COLOR\_BGR2GRAY) #graya dönüştürür

gray2 = cv2.medianBlur(image\_path, 11) #Blur yapar ve gürültüleri siler

ret, \_ = cv2.threshold(gray2, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU) #Thresh binnary eşik değeri ile siyah ve beyazı ayırır

cv2.imwrite('processing.jpg', image\_path)

print(ret)

circles = cv2.HoughCircles(gray2, cv2.HOUGH\_GRADIENT, 1, 50, param1=ret, param2=30, minRadius=20,

maxRadius=100) #Çemberleri bulur

print(circles)

circles = circles[0, :, :]

circles = np.int16(np.around(circles)) #Integer (-32768 to 32767) yuvarlama yapar

for i in circles[:]: #çemberleri çizer

image\_path = image\_path[i[1] - i[2] - r:i[1] + i[2] + r, i[0] - i[2] - r:i[0] + i[2] + r]

radus = i[2]/(1.9)

cv2.imwrite('processing.jpg', image\_path)

print(radus)

return image\_path, radus

def recflection\_remove(img):

ret, mask = cv2.threshold(img, 150, 255, cv2.THRESH\_BINARY)

kernel = np.ones((5, 5), np.uint8)

dilation = cv2.dilate(mask, kernel, iterations=1) #beyaz kısımları genişletir

dst = cv2.inpaint(img, dilation, 4, cv2.INPAINT\_TELEA)#gereksiz şeyleri siliyor

cv2.imwrite('kernel.jpg', kernel)

cv2.imwrite('dilation.jpg', dilation)

cv2.imwrite('dst.jpg', dst)

return dst

def daugman\_normalizaiton(image, height, width, r\_in, r\_out): # Daugman 640\*480, width\*height

thetas = np.arange(0, 2 \* np.pi, 2 \* np.pi / width) # Theta değerleri

r\_out = r\_in + r\_out

# Boş Flat oluşturma

flat = np.zeros((height, width, 3), np.uint8)

circle\_x = int(image.shape[0] / 2)

circle\_y = int(image.shape[1] / 2)

for i in range(width):

for j in range(height):

theta = thetas[i] # theta koordinat değerleri

r\_pro = j / height

Xi = circle\_x + r\_in \* np.cos(theta)

Yi = circle\_y + r\_in \* np.sin(theta)

Xo = circle\_x + r\_out \* np.cos(theta)

Yo = circle\_y + r\_out \* np.sin(theta)

Xc = (1 - r\_pro) \* Xi + r\_pro \* Xo

Yc = (1 - r\_pro) \* Yi + r\_pro \* Yo

color = image[int(Xc)][int(Yc)] # color of the pixel

flat[j][i] = color

return flat # liang

while True:

\_, frame = cap.read()

#frame = cv2.line(frame, (0, 0), (640, 480), (255, 0, 0), 4)

#frame = cv2.arrowedLine(frame, (0, 0), (640, 480), (255, 0, 0), 4)

#frame2 = cv2.circle(frame, (320, 240), 100, (0, 0, 255), 1)

#frame2 = cv2.circle(frame2, (320, 240), 35, (0, 0, 0), -1)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

gray = cv2.medianBlur(gray, 11)

Alt = cv2.getTrackbarPos("Alt", "Frame")

Ust = cv2.getTrackbarPos("Ust", "Frame")

CannyAlt = cv2.getTrackbarPos("CannyAlt", "Frame")

CannyUst = cv2.getTrackbarPos("CannyUst", "Frame")

canny = cv2.Canny(gray, CannyAlt, CannyUst)

\_, threshold = cv2.threshold(gray, Alt, Ust, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

cv2.imshow("canny", canny)

cv2.imshow("resim", frame)

cv2.imshow("Gray", gray)

cv2.imshow("Frame", threshold)

if cv2.waitKey(1) & 0xFF == ord('s'):

cv2.imwrite('test.jpg', frame)

cv2.imwrite('gray.jpg', gray)

gelen = cv2.imread('test.jpg')

image\_roi, rounds = processing(gelen, 40)

image\_roi = recflection\_remove(image\_roi)

image\_nor = daugman\_normalizaiton(image\_roi, 60, 360, rounds, 40)

cv2.imwrite('daugman.jpg', image\_nor)

break

cap.release()

cv2.destroyAllWindows()

Not: Kod halen düzenlenme aşamasında olduğundan içerisinde test amaçlı yazılan, kullanılmayan kodlar vardır.