|  |  |  |
| --- | --- | --- |
|  |  |  |

**T.C.**

**CUMHURİYET ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

|  |  |  |
| --- | --- | --- |
|  | **YÜZ TANIMA SİSTEMİ**    **Seyit Can BAĞ**  **2018141019**  **LİSANS BİTİRME PROJESİ** |  |

**Temmuz-2022**

**SİVAS**

**TEZ BİLDİRİMİ**

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Seyit Can BAĞ

Tarih: 27.07.2022

# 

ÖZET

**Yüz Tanıma Sistemi**

**Öğrenci: Seyit Can BAĞ**

**Danışman: Dr. Öğr. Üyesi Kali GÜRKAHRAMAN**

Projede olabildiğince yüksek doğrulukta yüz tanıma yapıldı. Projenin içerisinde 2 adet örnek proje var.

1. örneği anlatmakla başlayalım, öncelikle projede 2 adet resim kullanıyoruz ilk resim programın referans alması gereken resim ikincisi ise test resmi. Oluşturduğumuz program bu test resmine bakarak bu test resmindeki kişinin referans olarak verdiğimiz kişi ile aynı kişi olup olmadığını bize söylüyor (True, False şeklinde) ve aralarındaki benzerliği de yine test resmi üzerine bastırıyoruz. Tabii bunları yaparken iki resimde de insanların yüzlerini tanıyıp bir dikdörtgen içerisine alıyoruz.

2. Örneğimizde ise biraz farklı şeyler yapıyoruz. İkinci örnek projemizde biz kameramızı kullanmaya başlıyoruz ve canlı olarak kameranın karşısındaki kişinin görüntüsünü kullanarak o kişinin kim olduğunu buluyoruz. Projenin devamında sisteme yeni bir yüz eklemek istediğimizde, eklemek istediğimiz kişinin resmini resimlerimizin tutulduğu dosyaya aktarıyoruz ve artık bu kişiyi de kameramız canlı olarak tanıyabilecek olacak.

**Anahtar Kelimeler:** Histogram of Oriented Gradients (HOG), Yüz Tanıma sistemi, kütüphaneler

İÇİNDEKİLER

[ÖZET iv](#_Toc495327139)

[İÇİNDEKİLER v](#_Toc495327140)

[KISALTMALAR vi](#_Toc495327141)

[1. GİRİŞ 1](#_Toc495327142)

[1.1. Yüz Tanıma Nedir? 1](#_Toc495327143)

[1.2. Yüz Tanıma Sistemi Nasıl Çalışır? 1](#_Toc495327144)

[2. YÜZ TANIMA 2](#_Toc495327145)

[2.1. Yüz Tanımanın Tarihi 2](#_Toc495327146)

[2.2. Yüz Tanımanın Uygulama Alanları 3](#_Toc495327147)

2.2.1 Sağlık Hizmeti ……………………………………………………………… 3

2.2.2 Eğitim ………………………………………………………………………. 4

2.2.3 Bankacılık ve Finans………………………………………………………... 4

2.2.4 Mağaza Güvenliği ve Dolandırıcılık Önleme………………………………. 4

2.3. Yüz Tanıma Teknolojisinin Faydaları …………………………………………. 5

2.3.1.Kayıp kişileri bulmak ve failleri belirlemek………………………………... 5

2.3.2. Daha iyi tıbbi tedavi ……………………………………………………….. 5

2.3.3. Fotoğrafları düzenlemek için daha iyi araçlar ……………………………... 5

2.3.4. İnsan temas noktalarını büyük ölçüde azaltması …………………………... 5

2.3.5. Alışverişi çok daha verimli hale getirmesi ………………………………… 6

[3. MATERYAL VE YÖNTEM 7](#_Toc495327148)

[3.1. Yazılım Geliştirme Ortamı ve Yazılım Kütüphaneleri 7](#_Toc495327149)

[3.1.1. Python 7](#_Toc495327150)

3.1.2. PyCharm……………………………………………………………………. 7

3.1.3. OpenCv……………………………………………………………………... 7

3.1.4. Dlib…………………………………………………………………………. 8

3.1.5. NumPy……………………………………………………………………… 8

3.2. Yöntem………………………………………………………………………….. 9

3.2.1. HOG (Histogram of Oriented Gradients) Nedir?........................................... 9

[4. YÜZ TANIMA UYGULAMA 11](#_Toc495327151)

[4.1. İlk Proje (ResimleTanima) 11](#_Toc495327152)

[4.2. İkinci Proje (VideoluTanima) 13](#_Toc495327153)

[5. SONUÇLAR 17](#_Toc495327154)

6.[KAYNAKLAR 18](#_Toc495327157)

KISALTMALAR

**Kısaltmalar**

HOG: Histogram of Oriented Gradients (Yönlendirilmiş Histogramların Diyagramı), bu proje için kullanılan bir çeşit yüz tanıma algoritmasıdır.

IDE: **programlama işlemlerini organize edebilmenize yardımcı olan platformlardır.**

# GİRİŞ

## 1.1. Yüz Tanıma Nedir?

Yüz tanıma, bir kişinin yüzünü kullanarak kimliğini tanımlamanın veya onaylamanın bir yoludur. Yüz tanıma sistemleri, insanları fotoğraflarda, videolarda veya gerçek zamanlı olarak tanımlamak için kullanılabilir. Yüz tanıma, bir biyometrik güvenlik kategorisidir. Diğer biyometrik yazılım biçimleri arasında ses tanıma, parmak izi tanıma ve göz retinası veya iris tanıma bulunur. Diğer kullanım alanlarına artan ilgi olsa da, teknoloji çoğunlukla güvenlik için kullanılıyor.

* 1. **Yüz Tanıma Sistemi Nasıl Çalışır?**

Birçok kişi, iPhone'ların kilidini açmak için kullanılan FaceID aracılığıyla yüz tanıma teknolojisine aşinadır (ancak bu, yüz tanımanın yalnızca bir uygulamasıdır). Tipik olarak, yüz tanıma, bir bireyin kimliğini belirlemek için büyük bir fotoğraf veri tabanına dayanmaz. Yalnızca bir kişiyi cihazın tek sahibi olarak tanımlar ve tanır, diğerlerine erişimi sınırlar. Yüz teknolojisi sistemleri değişebilir, ancak genel olarak aşağıdaki gibi çalışma eğilimindedirler:

1. Adım: Yüz algılama
2. Adım: Yüz analizi
3. Adım: Görüntüyü verilere dönüştürme
4. Adım: Bir eşleşme bulma

# 2. YÜZ TANIMA

**2.1. Yüz Tanımanın Tarihi**

**1960’larda yüz tanıma:**

Yüz tanımanın ilk öncüleri Woody Bledsoe, Helen Chan Wolf ve Charles Bisson idi. 1964 ve 1965 yıllarında Bledsoe , Wolf ve Bisson ile birlikte insan yüzünü tanımak için bilgisayarları kullanarak çalışmaya başladı. İsmi açıklanmayan bir istihbarat teşkilatından gelen projenin finansmanı nedeniyle, çalışmalarının çoğu asla yayınlanmadı. Bununla birlikte, daha sonra, ilk çalışmalarının, göz merkezleri, ağız vb. gibi yüzdeki çeşitli yer işaretlerinin manuel olarak işaretlenmesini içerdiği ortaya çıktı.

**1970’lerde yüz tanıma:**

Bledsoe’nun ilk çalışmasından devam eden baton, 1970’lerde Goldstein, Harmon ve Lesk tarafından seçildi ve tanımayı otomatikleştirmek için çalışmayı saç rengi ve dudak kalınlığı dahil olmak üzere 21 belirteç içerecek şekilde genişletti. Doğruluk ilerlemiş olsa da ölçümlerin ve konumların hala manuel olarak hesaplanması gerekiyordu.

**1980’lerde yüz tanıma:**

İşletmeler için geçerli bir biyometrik olarak Yüz Tanıma yazılımının geliştirilmesinde daha fazla ilerleme görmemiz 1980'lerin sonlarına kadar değildi. 1988'de Sirovich ve Kirby, yüz tanıma problemine lineer cebir uygulamaya başladılar.

**1990’larda yüz tanıma:**

İleri Savunma Araştırma Projeleri Ajansı (DARPA) ve Ulusal Standartlar ve Teknoloji Enstitüsü (NIST), ticari yüz tanıma pazarını teşvik etmek için 1990’ların başında Yüz Tanıma Teknolojisi (FERET) programını başlattı. Proje, yüz görüntülerinden oluşan bir veri tabanı oluşturmayı içeriyordu. Test setine 856 kişiyi temsil eden 2.413 hareketsiz yüz görüntüsü dahil edildi. Umut, yüz tanıma için geniş bir test görüntüleri veritabanının yeniliğe ilham vermesi ve daha güçlü yüz tanıma teknolojisine yol açmasıydı.

**2000’lerde yüz tanıma:**

Ulusal Standartlar ve Teknoloji Enstitüsü (NIST), 2000'lerin başında Yüz Tanıma Satıcı Testlerine (FRVT) başladı.

**2006’da yüz tanıma:**

2006'da başlatılan Yüz Tanıma Büyük Mücadelesinin (FRGC) birincil amacı, ABD Hükümeti'ndeki mevcut yüz tanıma çabalarını desteklemek için tasarlanmış yüz tanıma teknolojisini teşvik etmek ve ilerletmekti. FRGC, mevcut en son yüz tanıma algoritmalarını değerlendirdi. Testlerde yüksek çözünürlüklü yüz görüntüleri, 3 boyutlu yüz taramaları ve iris görüntüleri kullanılmıştır. Sonuçlar, yeni algoritmaların 2002'deki yüz tanıma algoritmalarından 10 kat ve 1995'tekilerden 100 kat daha doğru olduğunu gösterdi ve son on yılda yüz tanıma teknolojisindeki ilerlemeleri gösterdi.

**2010 – Günümüz yüz tanıma:**

2010'da Facebook, Facebook kullanıcılarının günlük olarak güncellediği fotoğraflarda yüzleri bulunan kişileri tanımlamaya yardımcı olan yüz tanıma işlevini uygulamaya başladı. Bu özellik, haber medyasında anında tartışmalı hale geldi ve gizlilikle ilgili bir dizi makaleyi ateşledi. Ancak, Facebook kullanıcıları genel olarak umursamıyor gibi görünüyordu. Web sitesinin kullanımı veya popülaritesi üzerinde belirgin bir olumsuz etkisi olmayan, her gün yüz tanıma kullanılarak 350 milyondan fazla fotoğraf yükleniyor ve etiketleniyor.

## 2.2. Yüz Tanımanın Uygulama Alanları

Yüz tanıma sistemleri günümüzde birçok sektörde kullanılmaktadır ve sektörün ihtiyaçlarını gidermektedir.

**2.2.1 Sağlık Hizmeti:**

Bir hastanenin video gözetim sistemine dahil edildiğinde, yüz tanıma, hasta girişini basitleştirebilir, hastane çalışanlarını ve hastaları evrak işlerinden kurtarabilir ve insan hatasını önleyebilir. Bir yüz tanıma sistemi, bir hastaya "bakarak" onların kimlik ve sigorta verilerini doğrulayabilir, böylece kabul sürecini hızlandırabilir, kişiselleştirilmiş bir deneyim için temel oluşturabilir ve sahtekarlığı önleyebilir.

Yüz tanıma da dahil olmak üzere biyometrik teknolojiler, cerrahi hastaların kimliklerini doğrulamak, bir sağlık çalışanının refakatinde olmayan hastaları belirlemek ve güvenlik tehditlerini önlemek için tesislere giren ve çıkan insanları izlemek için de kullanılabilir.

**2.2.2 Eğitim:**

Devamlılığı takip etmek eskiden uzun ve sıkıcı bir süreçti ve her dersin başında harcanan oldukça uzun bir süreye rağmen manuel olarak yapıldığında kaçınılmaz boşluklara ve eksikliklere yol açardı. Bunu düzeltmek için eğitimciler yapay zeka destekli eğitim çözümlerine yönelmeye başladılar. Bu şekilde, yüz tanıma uygulamaları kimin bulunduğunu takip etmenin daha hızlı ve kesintiye uğramayan bir yolunu sunar. Değerli öğrenme zamanından tasarruf etmekle kalmaz, aynı zamanda müfredat tasarımcılarının daha uygun öğrenme ortamları ve mükemmel sınıf planlaması oluşturmasına da olanak tanır. Akıllı katılım izleme çözümleri, Birleşik Krallık ve Avustralya'daki eğitimcilerden özellikle ilgi gördü. Örneğin, Victoria's Department of Education öğrencilerin nerede olduğunu izlemek için yüz tanımaya başvurdu ve öğretmenlerin ve personelin bir web panosu veya mobil uygulama aracılığıyla katılım verilerine erişmesine izin verdi. Avustralyalı eğitimciler arasında popüler olan bir yüz tanıma tabanlı katılım izleme çözümü olan LoopLearn'in yaratıcıları, yüz tanıma kullanmanın haftada 2,5 saat öğretmene zaman kazandırdığını ve sahte katılım sorunlarının etkili bir şekilde üstesinden gelmeye yardımcı olduğunu iddia ediyor.

**2.2.3 Bankacılık ve Finans:**

Bugün, suçlular ATM'lere girmek için rutin olarak gözden geçirme cihazlarını kullanıyor. Yüz tanıma, sahtekarlığı önlemek için daha güvenli bir seçenek olarak potansiyel olarak plastik kartların ve PIN'lerin yerini alabilir. Birkaç banka şimdiden yüz tanıma çözümlerini test etmeye başladı. Örneğin, National Australia Bank, banka müşterilerinin ATM hizmetlerine erişmek için yüzlerini taramalarına olanak tanıyan bir sistem geliştirmek için Sydney merkezli OCR Labs ile ortaklık kurdu. Çözüm, müşterilerin kimlik belgelerinin görüntülerini yakalar ve bunları, makineye entegre bir kamera tarafından çekilen müşterilerin fotoğraflarıyla eşleştirir. Kavram kanıtı 2020'de başarıyla uygulandı ve banka, teknolojiyi federal hükümetin veri etiği çerçevesini karşılayacak şekilde mükemmelleştirmeye devam ediyor.

**2.2.4 Mağaza Güvenliği ve Dolandırıcılık Önleme:**

Yakın tarihli bir araştırmaya göre, ankete katılan alıcıların neredeyse yarısı amacı hırsızlık vakalarını önlemekse, mağazalarda yüz tanımanın uygulanmasını istiyor. Hırsızlığı önleyen sistemler genellikle fotoğrafları zaten bir veri tabanında saklanan mükerrer suçluları belirlemeyi hedefler. Bu nedenle bir yüz tanıma sistemi, bir mağaza ziyaretçisinin yüzüne kişisel olarak tanımlanabilir herhangi bir bilgi atfetmez, ancak bilinen suçluların veri tabanında eşdeğerini arar.

**2.3. Yüz Tanıma Teknolojisinin Faydaları:**

Yüz tanımanın teknolojisinin topluma sunabileceği, suçları önlemek ve güvenliği artırmaktan gereksiz insan etkileşimini ve emeğini azaltmaya kadar birçok fayda vardır. Bazı durumlarda, tıbbi çabaları desteklemeye bile yardımcı olabilir. Biz şimdi birkaç tanesini örnekleyerek bu konuyu açıklayalım.

**2.3.1. Kayıp kişileri bulmak ve failleri belirlemek:**

Emniyet teşkilatları, başka hiçbir kimlik tespit aracı olmaksızın suçluları tespit etmek ve canlı kamera yayınlarındaki yüzleri izleme listesindeki yüzlerle karşılaştırarak kayıp kişileri bulmak için yüz tanıma özelliğini kullanır. Ayrıca bunu kayıp çocukları bulmak için kullandılar. Yüz tanımayı yaşlanma yazılımıyla birleştirerek, çocukların birkaç yıl sonra nasıl görüneceklerini tahmin ederek onları yıllardır kayıp olsalar bile bulabilecek yüz tanıma teknolojisini geliştirdiler.

**2.3.2. Daha iyi tıbbi tedavi:**

Yüz tanıma teknolojisinin şaşırtıcı bir kullanımı, genetik bozuklukların tespitidir.

Yüz tanıma yazılımı, ince yüz özelliklerini inceleyerek bazı durumlarda belirli genetik mutasyonların belirli bir sendroma nasıl neden olduğunu belirleyebilir. Teknoloji, geleneksel genetik testlerden daha hızlı ve daha ucuz olabilir.

**2.3.3. Fotoğrafları düzenlemek için daha iyi araçlar:**

Yüz tanıma, bulut depolama alanınızdaki fotoğrafları Apple veya Google aracılığıyla da etiketleyebilir. Bu, fotoğraflarınızı organize etmeyi, bulmayı ve paylaşmayı kolaylaştırır. Ayrıca Facebook'ta etiket önermede de rol oynar.

**2.3.4. İnsan temas noktalarını büyük ölçüde azaltması:**

Yüz tanıma, parmak izi alma gibi diğer güvenlik önlemleri türlerinden daha az insan kaynağı gerektirir. Ayrıca fiziksel temas veya doğrudan insan etkileşimi gerektirmez. Bunun yerine, otomatik ve sorunsuz bir süreç haline getirmek için yapay zekayı kullanır.

Ayrıca kapıların ve akıllı telefonların kilidini açarken, ATM'den nakit alırken veya genellikle bir PIN, şifre veya anahtar gerektiren diğer görevleri gerçekleştirirken temas noktalarını sınırlar.

**2.3.5. Alışverişi çok daha verimli hale getirmesi:**

Kayıp kişileri ve suçluları tespit etmek ve bulmak yüz tanımanın tartışmasız en önemli faydaları olsa da güvenliğin ötesinde rahatlığa uzanıyor.

Mağazalarda nakit veya kredili alışveriş yapmak yerine yüz tanıma teknolojisi yüzünüzü tanıyabilir ve ürünleri hesabınıza yükleyebilir. Bunun kullanımı, pandemi sırasında hem kolaylık hem de güvenlik amaçlarına hizmet etmek ve ayrıca daha küçük personel/müşteri oranını yönetmeye yardımcı olmak için arttı.

# 3. MATERYAL VE YÖNTEM

## 3.1. Yazılım Geliştirme Ortamı ve Yazılım Kütüphaneleri

3.1.1. Python:

Python, üst düzey, genel amaçlı, yorumlanmış nesne yönelimli bir programlama dilidir. PERL'ye benzer şekilde Python, C++ ve Java programları arasında popüler olan bir programlama dilidir. Python otuz yıl önce ortaya çıktı. Mucidi ise Hollandalı programcı Guido van Rossum'dur. Guido van Rossum o sırada en sevdiği komedi grubu olan Monty Python's Flying Circus'un adını verdi.

Python, güçlü, hızlı ve programlamayı daha eğlenceli hale getirmesiyle bilinir. Python kodlayıcıları, değişkenin ne olması gerektiğini açıklamak zorunda kalmadan değişkenleri dinamik olarak yazabilir. Kullanıcılar Python'u ücretsiz olarak indirebilir ve onunla hemen kodlamayı öğrenmeye başlayabilir. Kaynak kodu serbestçe kullanılabilir ve değişiklik ve yeniden kullanıma açıktır.

3.1.2. PyCharm:

PyCharm, verimli Python, web ve veri bilimi geliştirme için uygun bir ortam oluşturmak üzere sıkı bir şekilde entegre edilmiş Python geliştiricileri için çok çeşitli temel araçlar sağlayan özel bir IDE'dir.

Professional ve Community adı altında 2 adet indirme seçeneği vardır. Siz community seçeneğini seçerek ücretsiz bir şekilde kullanabilirsiniz.

3.1.3. OpenCV:

OpenCV (Open Source Computer Vision Library), açık kaynaklı bir bilgisayarlı görü ve makine öğrenimi yazılım kitaplığıdır. OpenCV, bilgisayarla görü uygulamaları için ortak bir altyapı sağlamak ve ticari ürünlerde makine algısının kullanımını hızlandırmak amacıyla oluşturulmuştur.

Kütüphane hem klasik hem de son teknoloji bilgisayarlı görme ve makine öğrenimi algoritmalarından oluşan kapsamlı bir set içeren 2500'den fazla optimize edilmiş algoritmaya sahiptir. Bu algoritmalar yüzleri algılamak ve tanımak, nesneleri tanımlamak, videolarda insan eylemlerini sınıflandırmak, kamera hareketlerini izlemek, hareketli nesneleri izlemek, nesnelerin 3B modellerini çıkarmak, stereo kameralardan 3B nokta bulutları üretmek, yüksek çözünürlük elde etmek için görüntüleri bir araya getirmek için kullanılabilir.

3.1.4. Dlib:

Herhangi bir tür görüntü işleme, bilgisayarla görme veya makine öğrenimi ile ilgileniyorsanız, yolculuğunuzun bir yerinde dlib ile karşılaşma/kullanma şansınız yüksektir.

Dlib, C++'da gerçek dünya makine öğrenimi ve veri analizi uygulamaları yapmak için bir araç takımıdır. Kitaplık orijinal olarak C++ ile yazılmış olsa da iyi ve kullanımı kolay Python bağlamalarına sahiptir.

3.1.5. NumPy:

NumPy, Python'da bilimsel hesaplama için temel pakettir. Çok boyutlu bir dizi nesnesi, çeşitli türetilmiş nesneler (maskelenmiş diziler ve matrisler gibi) ve diziler üzerinde matematiksel, mantıksal, şekil işleme, sıralama, seçme, G/Ç dahil olmak üzere hızlı işlemler için çeşitli rutinler sağlayan bir Python kitaplığıdır. ayrık Fourier dönüşümleri, temel doğrusal cebir, temel istatistiksel işlemler, rastgele simülasyon ve çok daha fazlası.

NumPy paketinin merkezinde ndarray  nesnesi bulunur. Bu, performans için derlenmiş kodda gerçekleştirilen birçok işlemle birlikte, homojen veri türlerinin n boyutlu dizilerini kapsar. NumPy dizileri ile standart Python dizileri arasında birkaç önemli fark vardır:

* NumPy dizileri, Python listelerinin (dinamik olarak büyüyebilen) aksine, oluşturulurken sabit bir boyuta sahiptir. Bir ndarray'in boyutunu değiştirmek, yeni bir dizi oluşturacak ve orijinali silecektir.
* NumPy dizisindeki öğelerin hepsinin aynı veri türünde olması gerekir ve bu nedenle bellekte aynı boyutta olacaktır. İstisna: bir (Python, NumPy dahil) nesne dizilerine sahip olabilir, böylece farklı büyüklükteki öğelerin dizilerine izin verilir.
* NumPy dizileri, çok sayıda veri üzerinde gelişmiş matematiksel ve diğer türdeki işlemleri kolaylaştırır. Tipik olarak, bu tür işlemler Python'un yerleşik dizileri kullanılarak mümkün olandan daha verimli ve daha az kodla yürütülür.
* Artan sayıda bilimsel ve matematiksel Python tabanlı paketler NumPy dizilerini kullanıyor; Bunlar tipik olarak Python dizisi girişini desteklese de bu girişi işlemeden önce NumPy dizilerine dönüştürürler ve genellikle NumPy dizilerinin çıktısını alırlar. Başka bir deyişle, günümüzün bilimsel/matematiksel Python tabanlı yazılımlarının çoğunu (belki de çoğunu) verimli bir şekilde kullanmak için, yalnızca Python'un yerleşik dizi türlerini nasıl kullanacağınızı bilmek yetersizdir. Ayrıca NumPy dizilerinin nasıl kullanılacağını bilmek gerekir.

Dizi boyutu ve hızı ile ilgili noktalar, bilimsel hesaplamada özellikle önemlidir.

3.2. Yöntem

Bu yüz tanıma projesi HOG (Histogram of Oriented Gradients) algoritması kullanılarak hazırlanmıştır. Sıradaki bölümde bu algoritmayı inceleyeceğiz.

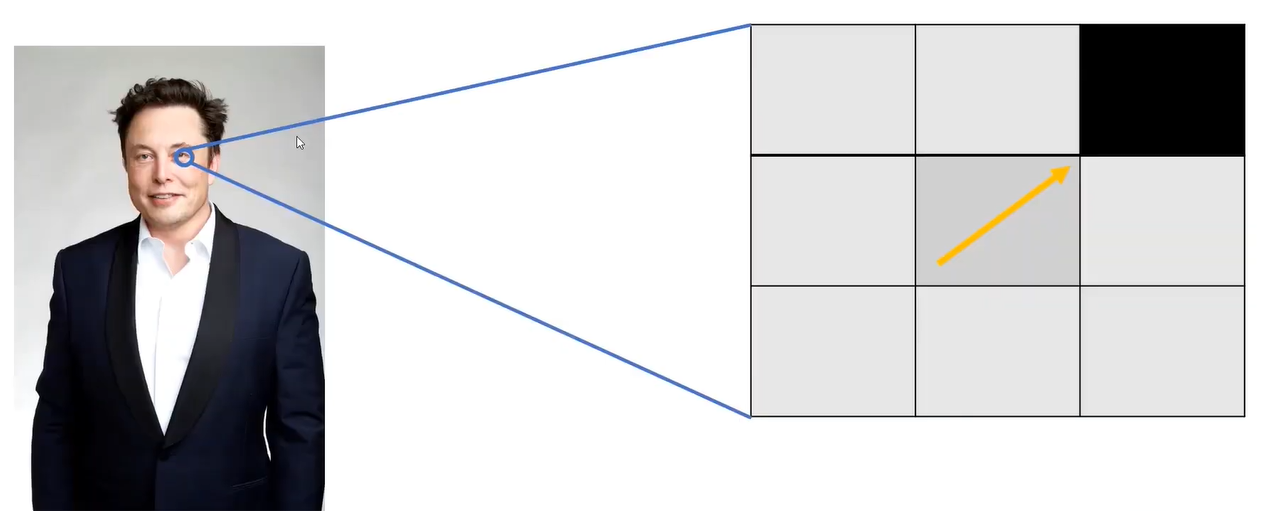
3.2.1. HOG (Histogram of Oriented Gradients) Nedir?

Navneet Dalal ve Bill Triggs, 2005 yılında Histogram of Oriented Gradients(HOG) özelliklerini tanıttı. Histogram of Oriented Gradients (HOG), görüntü işlemede, özellikle nesne tespiti için kullanılan bir özellik tanımlayıcıdır. Özellik tanımlayıcı, bir görüntünün temsilidir veya görüntüden yararlı bilgiler çıkararak görüntüyü basitleştiren bir görüntü yamasıdır.

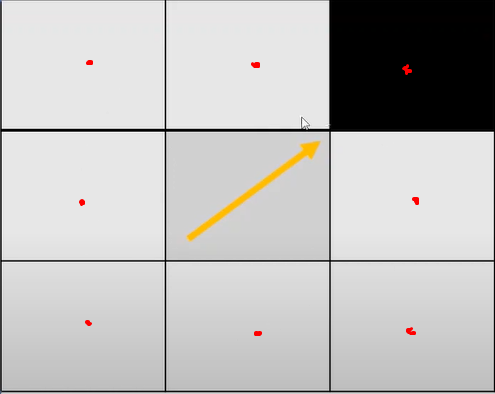
Yönlendirilmiş gradyan tanımlayıcısının histogramının arkasındaki ilke, bir görüntü içindeki yerel nesne görünümü ve şeklinin, yoğunluk gradyanlarının veya kenar yönlerinin dağılımı ile tanımlanabilmesidir. Bir görüntünün x ve y türevleri (Degradeler) yararlıdır çünkü yoğunluktaki ani değişiklik nedeniyle kenarlar ve köşelerdeki gradyanların büyüklüğü büyüktür ve kenarların ve köşelerin nesne şekli hakkında düz bölgelere göre çok daha fazla bilgi içerdiğini biliyoruz. Bu nedenle, degradelerin yönlerinin histogramları bu tanımlayıcıda özellik olarak kullanılır.

HOG algoritması nasıl çalışır?

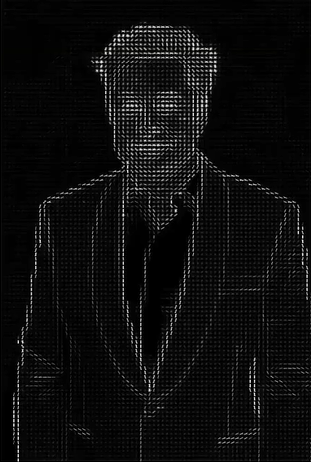
HOG dediğimiz şey aslında görüntüdeki aydınlık ve karanlık noktaların yoğunluklarının bir ok ile belirlenmesi gibi düşünebiliriz. Örneğin biz aşağıdaki (a) görselinde de göreceğiniz gibi göz kısmındaki çok küçük bir bölgeyi ele alıyor olalım ve bu bölge 9 piksellik bir bölge olsun.

 (a)

Bu piksellerin renkleri de görüntüdeki gibi olsun ve biz örnek olarak bir pikseli alalım. Mesela bu pikseller arasından ortadakini seçelim. Ve şimdi biz ortadaki pikselin tüm komşu piksellerini inceliyoruz aşağıdaki (b) görselinde de göreceğiniz gibi 8 adet komşusu var.

 (b)

Ve gördüğünüz gibi sağ üst komşu pikseli diğerlerine göre daha karanlık. İşte bu yüzden biz yönü o piksele doğru olan bir ok çiziyoruz. Şimdi ise biz aynı bu 9 piksel için yaptığımız her şeyi tüm görüntüyü dolaşarak tamamı için yapıyoruz ve resim dosyamızın her pikseli için bir ok belirlemiş oluyoruz. Sonrasında biz tüm bunları yaptığımız zaman aşağıdaki gibi bir görüntü elde ediyoruz.ve bu görüntü aslında bizim belirli örüntüleri yakalayabilmemizi sağlıyor.

 (c)

Şimdi sadece yüz görüntüsüne odaklanalım ve biz bu okları yüzün tüm bölümlerine yerleştirdiğimizi düşünelim. Biz yüz kısmına baktığımızda orada bir yüz görüntüsü olduğunu anlayabildiğimiz gibi bilgisayar da orada böyle bir örüntü olduğunu anlayacaktır böylelikle biz tüm yüzler için belirli bir örüntü yakalamış olacağız. Yani biz HOG algoritması ile yüz görüntülerini algılayabiliyor olacağız.

4. YÜZ TANIMA UYGULAMA

Uygulama bölümünde biraz kodların içine gireceğiz ve hangi kod ne için kullanıldı bundan bahsedeceğiz. Daha önce özet kısmında da anlatıldığı gibi bizim ödevimizin içinde 2 adet projemiz mevcut.

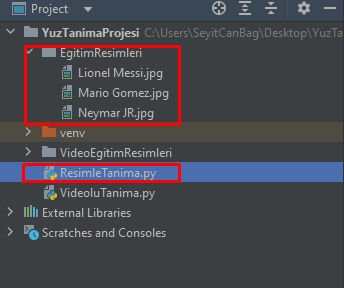
İlk projemiz girdi olarak 2 adet resim alıyor ve bu iki resimdeki kişilerin aynı kişi olup olmadığını kontrol ediyor aynı zamanda resimlerdeki yüzleri algılıyor, yüzleri algılıyor ve yüzün etrafına bir çerçeve çiziyor bu çerçeve üzerine ise aynı kişi olup olmadığını yazıyor (True, False) şeklinde. Aynı kişi olup olmadığını, resimdeki insanların yüzlerindeki dudak burun arası mesafe gibi birçok mesafeyi hesaplayarak yapıyor bu mesafeler 2 resim arasında çok yakınsa aynı kişidir sonucunu veriyor ama değilse aynı kişi olmadığını bize söylüyor.

İkinci projemizde ise bizim kullanmasını istediğimiz resimleri sisteme tanıtıyoruz ve yine ilk projede olduğu gibi bu resimlerde de mesafe hesaplaması kullanılıyor. Sistem video kamera karşısındaki kişinin yüzünü algıladıktan sonra bu kişinin de ağız-burun vs. arası birçok mesafeyi hesaplıyor ve bizim sisteme tanıttığımız tüm resim görüntüleriyle karşılaştırıyor bu mesafelerin birbirine en yakın olduğu resimdeki kişinin adını bizim video kamera karşısındaki kişinin yüz görüntüsünün adının altına bastırıyor ve böylelikle biz bir tanıma yapmış oluyoruz.

Şimdi ilk projemizin kodlarıyla başlayalım…

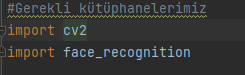
* 1. İlk Proje (ResimleTanima):

Öncelikle projemizin bulunduğu kısmı ve gerekli olan resimlerimizin tutulduğu klasörü göstermekle başlayalım.



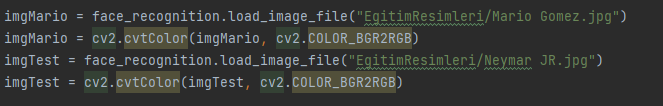
Şimdi ise kod kısmına girebiliriz.

Öncelikle projemize gerekli kütüphanelerimizin eklenmesiyle başlıyoruz.

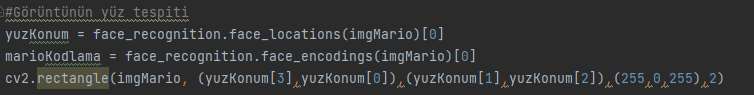


Daha sonra karşılaştırmak istediğimiz 2 resmimizi içe aktarıyoruz ve görüntülerimizi RGB’ye dönüştürüyoruz çünkü kütüphane görüntüleri RGB olarak algılayabiliyor.

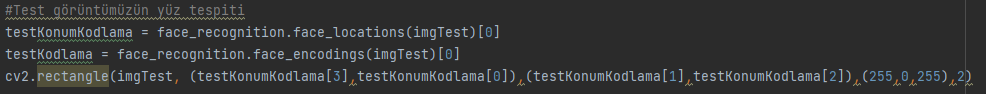
Elimizde 2 adet görüntümüz var biz imgTest olan görüntümüzün diğer görüntüyle aynı olup olmadığını test görüntümüzün üzerine yazdırmak istiyoruz.



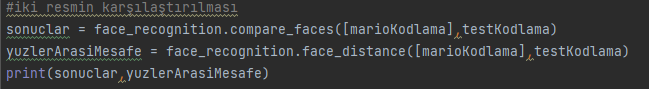
Şimdi ise yüz tespiti yapıyoruz. İlk satırda yüzün konumunu buluyoruz ve yuzKonum’un içerisine aktarıyoruz. İkinci satırda ise imgMario resmini kodlamak istediğimizi söylüyoruz ve marioKodlama değişkenine aktarıyoruz. Üçüncü satırda ise daha önce bulduğumuz konuma bir çerçeve çiziyoruz ve çerçevenin rengi kalınlığı gibi ayarlamalar yapıyoruz.



Şimdi aynı işlemleri test görüntüsü için yapıyoruz.



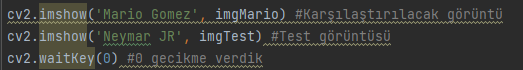
Sıradaki adımımızda bu iki yüz görüntüsünü karşılaştıracağız bunu göz burun aralıkları vs. mesafe ölçümü ile yapacağız ve iki resim aynı kişi ise true değilse false dönecek. İlk satırda iki resim arasında karşılaştırma yapıyoruz ikinci satırda ise iki resim arasındaki mesafeyi ölçüyoruz. Bu mesafe bu proje için 0,5 ve yaklaşık değerlerinde veya daha az ise bu iki kişi aynı kişi sonucunu dönecektir. Son satırda ise sadece bu bilgileri konsola bastırıyoruz.



Bu aşamada test görüntümüze text yazıyoruz. Text’in içinde true veya false değeri ve iki resim arasındaki mesafenin değeri basılı olacak ve yine aynı zamanda text’in büyüklüğü, fontu ve kalınlığı gibi ayarlamalar olacak.

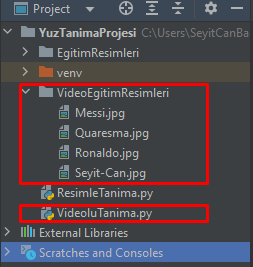


Son olarak görüntülerimizi ekrana bastırmak kaldı

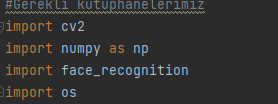


4.2 İkinci Proje (VideoluTanima):

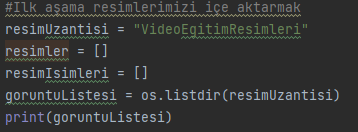
Yine bu projede de projemizin ve resim dosyalarımızın tutulduğu klasörü göstererek başlayalım.



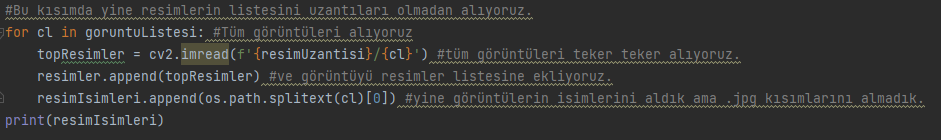
Kullanacağımız kütüphaneleri eklemekle devam edelim.



Şimdi ise yapacağımız ilk şey resimlerimizin tutulduğu uzantıyı almak ve resimlerimizin tamamını içe aktarmak. 4. Satırda ise tüm resimlerimizin isimlerinin listesini aldık ve bunu goruntuListesei’ne aktardık.

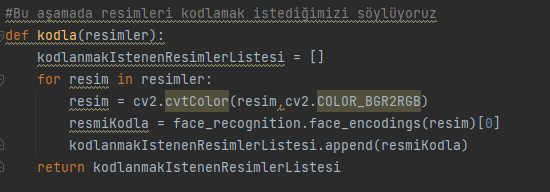


Bu kısımda yine resimlerimizin isimlerinin listesini aldık fakat bu kez farklı olarak uzantılarını almadık çünkü bu isimleri kameradaki yüz görüntümüze bastıracağız. İlk satırda tüm görüntüleri dolaşarak başlıyoruz 2.Satırda tüm resimleri teker teker alıyoruz ve 3. Satırda görüntüyü resimler listesine ekliyoruz. 4. Satırda uzantıları olmadan listeyi almış olduk ve son olarak da 5. Satırda bu listeyi bastırdık.



Bu aşamada resimlerimizin hepsini kodlamak istediğimizi söylüyoruz ve dönüş değeri olarak yine kodlamak istediğimiz resimlerin listesini alıyoruz.

İlk olarak listeyi alabilmek için boş bir liste açıyoruz ve resimleri teker teker dolaşmaya başlıyoruz. İlk olarak resimleri RGB’ye dönüştürüyoruz. Sonrasında resim değişkenindeki resmi kodlamak istediğimizi söylüyoruz ve resmiKodla değişkenine aktarıyoruz ve return’den önceki son satırda kodlamak istediğimiz tüm resimleri listemize aktarıyoruz.



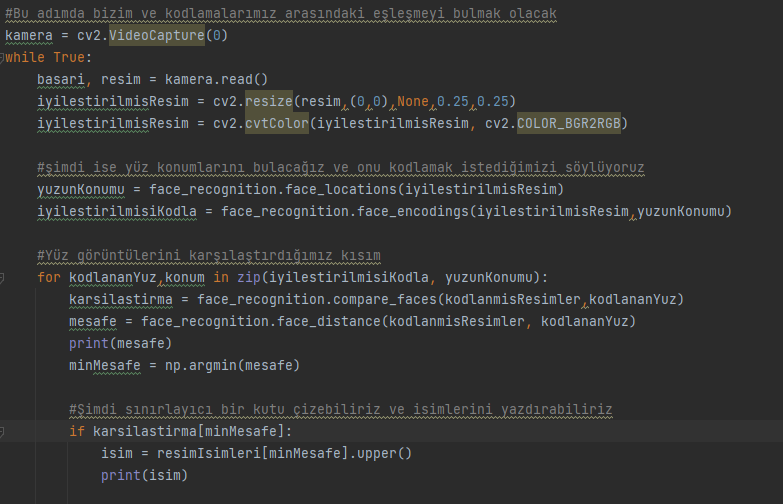
Şimdi de kodlanmasını istediğimiz tüm resimlerin sayısını konsola bastırıyoruz. Yukarıdaki görselde yazılan fonksiyonu çağırarak kodlanmış resimleri alıyoruz ve son satırda bunun uzunluğunu ekrana bastırıyoruz.

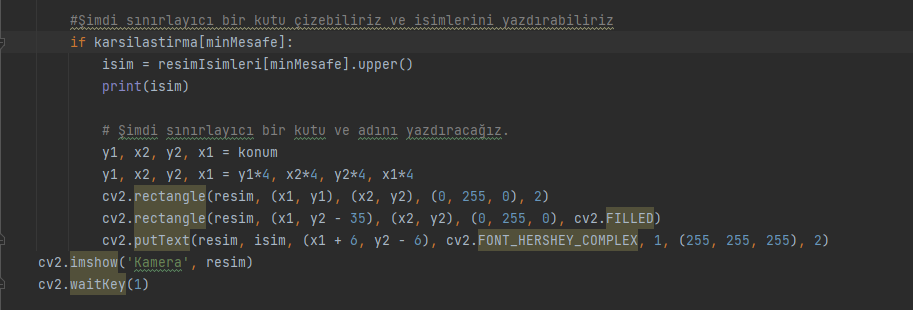


Ve son aşamaya geldik bu aşama biraz uzun olacak çünkü bu aşama birçok işlemi barındırıyor. Bu aşamanın genel amacı kameradaki görüntümüz ile resimler arasındaki eşleşmeyi bulmak olacak. İlk satırda kameramızı başlatarak başlıyoruz ve sonra her bir kareyi teker teker almak için while döngümüzü başlatıyoruz. 3.Satırda kameradaki görüntümüzü resim ve basari değişkenlerine aktarıyoruz. 4.Satırda süreci hızlandırmak için kameranın boyutunu küçülttük ve iyilestirilmisResim içerisine aktardık 5. Adımda da bu görüntüyü RGB’ye çevirdik.

Sonrasında açıklama satırında da yazıldığı gibi yüz konumlarını buluyoruz ve onu kodlamak istediğimizi söylüyoruz. For döngüsüne geldiğimizde ise burada yüz görüntülerini karşılaştıracağız. For döngüsünde zip kullanarak eşleme yapmış oluyoruz.Yani iyilestirilmisiKodla ve yuzunKonumu adlı değişkenleri KodlananYuz ve konum’a eşlemiş olduk ve ilk olarak karşılaştırma yapıyoruz. Karşılaştırma kamera karşısındaki kişi ile tüm resimler arasında geçiyor ve karşılaştırma sonucunu karsilastirma değişkenine aktarıyoruz. Bir sonraki satırda kameradaki görüntü ile resimler arasındaki mesafeyi ölçüyoruz. Sonrasında mesafeyi bastırıyoruz ve minimum mesafeyi hesaplıyoruz. İf ile beraber en küçük mesafe değerine sahip resmi buluyoruz ve bu resmin ismini büyük harflerle yazmak istediğimizi söylüyoruz. Ve ismi bastırıyoruz.

Sonraki kısım yüzü çerçeve içine almak. Önce konum değerlerimizi x ve y değişkenlerine aktarıyoruz sonra bu değerleri 4 ile çarparak genişletiyoruz. Sonraki satırda yüz çerçevesini çizdirmiş oluyoruz ve hemen altındaki satırda da çerçevenin altında isimlerin yazılı olacağı 2. Çerçeveyi çizdiriyoruz. İf içerisindeki son satırda yazıyı yazdırıyoruz ve son olarak da görüntümüzü görmek istediğimizi söylüyoruz ve bir gecikme veriyoruz.





**5. SONUÇLAR**

Bitirme ödevi kapsamında geliştirilen proje ile:

* Sisteme daha önce tanıtılan resimler sayesinde kişilerin yüz ve kimlik tanımlaması yapılmaktadır. Sisteme tanıtılacak kişilerin sayısının artırılması ile bu projenin çeşitli uygulama alanlarındaki etkinliğini daha da fazla artırabiliriz
* Ülkelerde bu ve bunun gibi yüz tanımaya dayalı projeler kullanılırsa ve her ülke için o ülkenin tüm vatandaşlarının yüz kaydını sistemimize ekleyebilirsek o ülkelerdeki suçluların tespitinde çok önemli bir yol almış oluruz ve belki de tüm dünyada suç işlenme oranını düşürebiliriz. Yine bu şekilde kullanım alanını değiştirerek farklı alanlarda da faydalı hizmetler verebiliriz.
* Bu proje için çalışırken yaşadığım en büyük problemlerden bir tanesi Dlib kütüphanesinin projeye eklenmesiydi. Dlib kütüphanesinin çoğu versiyonunu, PyCharm IDE’m ile uyuşmaması çok uzun uğraşlar vermeme sebep oldu. Ve yine diğer bir sorun da kodda yüzün konumunun bulunduğu kısımda konum değerlerinin x ve y değişkenlerine doğru bir şekilde aktarılamamasıydı.
* Yukarıda yazmış olduğum sorunların çözümünü derin öğrenme ile çalışan arkadaşlarım, çeşitli web sayfaları, youtube ve https://stackoverflow.com/ sayfalarından giderdim. Bu sayfaların linkleri son kısımdaki kaynaklar kısmında paylaşılmıştır.

**KAYNAKLAR**

[1] Anonim, "Yüz Tanıma teknolojileri Nasıl Çalışır"

http://www.olaganustukanitlar.com/yuz-tanima-teknolojisi-nasil-calisir/[Accesed: 15Kasım2021]

[2] Anonim, " Yüz Tanıma sistemi Algoritmaları nelerdir?"

http://ieee.omu.edu.tr/yuz-tanima-algoritmalari-ve-uygulamalari/[Accesed:15Kasım2021]

[3] Anonim, "Python ile Yüz Tanıma"

https://www.mekinfo.net/opencv-ve-python-ile-goruntu-isleme-kamerada-yuz-tespiti/ [Accesed: 15Ocak2022]

[4] Anonim, "OpenCV Nedeir?"

https://mekatronikmuhendisligi.com/opencv-nedir/

[Accesed: 15Ocak2022]

[5] Anonim, "Fecial Recognition With Dlib Library"

https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/ [Accesed: 8Şubat2022]

[6] Anonim, "Dlib Library Not Found"

https://stackoverflow.com/questions/41912372/dlib-installation-on-windows-10 [Accesed: 23Haziran2022]

[7] Anonim, "Face Detection using HOG and Dlib"

https://www.youtube.com/watch?v=7kjxxP2N3lU

[Accesed: 23Haziran2022]