



KARABÜK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ
LİSANS BİTİRME TEZİ

VR GÖZLÜKLE YÜZ TANIMAYA DAYALI SABİKA ANALİZİ

ALPER ASLAN

Danışman

Dr. Öğr. Üyesi Ümit ATİLA

KARABÜK 2019

Alper ASLAN tarafından hazırlanan “VR GÖZLÜKLE YÜZ TANIMAYA DAYALI SABİKA ANALİZİ” başlıklı bu projenin Bitirme Projesi Tezi olarak uygun olduğunu onaylarım.

Dr.Öğr.Üyesi Ümit ATİLA

.....

Bitime Projesi Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

“Bu projedeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.”

Alper ASLAN

ÖZET

Bitirme Projesi Tezi

VR GÖZLÜKLE YÜZ TANIMAYA DAYALI SABİKA ANALİZİ

Alper ASLAN

Karabük Üniversitesi

Bilgisayar Mühendisliği

Bilgisayar Mühendisliği Bölümü

Tez Danışmanı:

Dr.Öğr.Üyesi Ümit ATİLA

Haziran 2019, 35 sayfa

Teknoloji sürekli gelişmektedir. Teknolojinin gelişmesiyle birlikte günlük hayatımızda yaptığımız işler hem daha kolay hem daha güvenilir hale gelmiştir. Bununla birlikte kaçınılmaz bir şekilde suç oranları da doğru orantıda artmıştır. Suçlularla mücadele etmek amacıyla havaalanı gibi geçiş noktalarında güvenliğin daha üst düzeye çıkarılması amacıyla bir gözlük vasıtasıyla yüz tanımaya dayalı olarak sabıka analizi yapabilen bir gözlük geliştirilmiştir.Yüz tanıma süreci dört aşama halinde değerlendirilebilir : Yüz verilerinin veritabanında suçlular ile ilişkilendirilmesi, yüklenen fotoğraflardan yüzlerin tespit edilmesi ve gerekli işlemler ile eğitime hazır hale getirilmesi, eğitime hazır hale getirilen fotoğrafların yüz tanıma algoritmalarıyla eğitilmesi ve elde edilen eğitilmiş yüz verileri ve veritabanıyla yüzlerin tespit edilmesinin sağlanmasıdır. Bu çalışmada yüz tanıma sürecinin iyileştirilmesi için daha etkili yöntemler ve donanımlar üzerine bir takım geliştirmeler önerilecektir.

ABSTRACT

Senior Project Thesis

CRIMINAL RECORD ANALYSIS WITH VR GLASSES BASED ON FACIAL RECOGNITION

Alper ASLAN

Karabuk University

Faculty of Engineering

Department of Computer Engineering

Project Supervisor:

Assist.Prof.Dr. Ümit ATİLA

June 2019, 35 pages

Technology is constantly evolving. With the development of technology, our work in daily life has become easier and more reliable. However, inevitably crime rates have increased in the right proportion. In order to combat criminals, a spectacle can be developed by means of face recognition based on face recognition through a pair of goggles to enhance security at crossing points such as airports. The face recognition process can be evaluated in four stages: Prepared for training with the operations, training prepared for the face of the photo-recognition algorithm to be trained and the face data obtained and the database is to provide the detection of faces. In this study, a number of enhancements on more effective methods and equipment will be proposed to improve the facial recognition process.

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde, oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, Sayın hocam Dr.Öğr.Üyesi Ümit ATİLA'a teşekkürlerimi sunar ve üniversitede aldığım eğitim boyunca bilgisayar teknolojileri konusunda bizi bilgilendiren, yönlendiren ve ufkumuzu genişleten tüm hocalarımıza teşekkürü borç bilirim.

İÇİNDEKİLER

Sayfa

KABUL.....	ii
ÖZET.....	iv
ABSTRACT.....	v
TEŞEKKÜR.....	vi
İÇİNDEKİLER	vii
ŞEKİLLER DİZİNİ.....	viii
BÖLÜM 1	1
GİRİŞ	1
1.1. LİTERATÜR ÖZETİ	1
1.2. PROJENİN AMACI.....	2
BÖLÜM 2	3
GÖZLÜK DİZAYNINDA KULLANILAN MALZEMELER.....	3
2.1. Raspberry Pi NEDİR?	3
2.1.1. Raspberry Pi 3B+.....	3
2.1.2. Raspberry Pi 3B+ İle Neler Yapılabilir ?	4
2.2. 5 inç Taşınabilir Dokunmatik Ekran	5
2.3. Güç Modülü	6
2.4. Fresnel Lens.....	7
2.5. Everest VR-0023 3D SANAL GERÇEKLİK GÖZLÜĞÜ	8
2.6. Ara Uzatma Parçası	8
2.7. Raspberry Pi Model B Kamera Kiti	9
BÖLÜM 3	10
PROJENİN YAZILIM VE TASARIM KISMININ HAZIRLANMA	
AŞAMALARI	10
3.1.Masaüstü Uygulamasının Hazırlanması	10
3.1.1. Ana Pencere Özelliklerinin Belirlenmesi.....	10
3.1.2. Giriş Ekranı Tasarımı Ve Veritabanı Bağlantısının Kurulması.....	11
3.1.3. Ana Ekranın Tasarımı , Verilerin Listelenmesi ve CRUD İşlemleri .	15

3.1.4. Suçlu Fotoğraf Ekleme , Fotoğrafların Gösterilmesi, Fotoğraflar Üzerinde Yüz Tespitinin Yapılması ve Eklenen Fotoğrafların Eğitilmesi ..	19
3.2.Yüz Tanıma İşlemi	26
3.3.Gözlüğün Fiziksel Tasarımı.....	29
BÖLÜM 4	33
SONUÇ VE DEĞERLENDİRME	33
KAYNAKLAR.....	34
ÖZGEÇMİŞ.....	35

ŞEKİLLER DİZİNİ

Şekil 2. 1.Raspberry Pi 3B+	3
Şekil 2. 2.Ekran	5
Şekil 2. 3.Güç Modülü	6
Şekil 2. 4.Fresnel Lens	7
Şekil 2. 5.VR Gözlük	8
Şekil 2. 6.Ara Uzatma Parçası.....	8
Şekil 2. 7.Raspberry Pi Kamerası	9
Şekil 3. 1.Program Ekran Dizaynı.....	10
Şekil 3. 2. Admin Tablosu Oluşturma.....	11
Şekil 3. 3. Veritabanı Bağlantısı Oluşturma	12
Şekil 3. 4. Giriş Ekranı Form Oluşturma	13
Şekil 3. 5. Admin Bilgi Kontrolü Fonksiyonu	13
Şekil 3. 6. Giriş Ekranı.....	14
Şekil 3. 7. Suçlu Veritabanının Oluşturulması.....	15
Şekil 3. 8. Ana Ekran Label Ve Entry Oluşturulması	15
Şekil 3. 9. Butonların Yerleştirilmesi.....	16
Şekil 3. 10. Ekranda Listeleme	16
Şekil 3. 11. Suçlu Kaydetme	17
Şekil 3. 12. Suçlu Bilgi Giriş Bölümü.....	17
Şekil 3. 13. Suçlu Bilgileri Listeleme Fonksiyonu	17
Şekil 3. 14. Suçlu Listesi.....	17
Şekil 3. 15. Entryler Üzerinde Veri Getirilmesi.....	18

Şekil 3. 16. Suçlu Bilgileri Güncelleme Fonksiyonu.....	18
Şekil 3. 17. Suçlu Bilgileri Güncelleme.....	18
Şekil 3. 18. Silme Fonksiyonu	19
Şekil 3. 19. Fotoğraf Ekleme Fonksiyonu.....	19
Şekil 3. 20. Fotoğrafları Gösterme Fonksiyonu	20
Şekil 3. 21. Haar Cascade Yöntemi Anlatımı	20
Şekil 3. 22. Klasörden Fotoğrafların Alınması ve Okunması	21
Şekil 3. 23. Fotoğraflardan Yüz ve Gözlerin Tespit Edilmesi	21
Şekil 3. 24. Fotoğraflardan Yüz ve Gözlerin Tespit Ekranı.....	22
Şekil 3. 25. Detected Faces Klasörü.....	23
Şekil 3. 26. LBPH Çalışma Mantığı	23
Şekil 3. 27. LBP Histogram	24
Şekil 3. 28. Yüz Eğitme Fonksiyonu	24
Şekil 3. 29. Yüz Eğitme Ekranı.....	25
Şekil 3. 30. Yüz Tanıma Veritabanı ve Kamera Ayarı	26
Şekil 3. 31. Yüz Tanıma İşlemi ve Veritabanından Bilgi Çekme İşlemi.....	27
Şekil 3. 32. Yüzlerin Karşılaştırma Sonucu Bilgilerin Ekranda Gösterilmesi.....	28
Şekil 3. 33. Sonuç	28
Şekil 3. 34. Fresnel Lens	29
Şekil 3. 35. Ekran Montajı	30
Şekil 3. 36. Ara Uzatma Parçası Montajı.....	30
Şekil 3. 37. Kamera Montajı	31
Şekil 3. 38. Raspberry Pi ve Güç Modülü Montajı.....	32
Şekil 3. 39. Kullanıcının Gözlükten Gördüğü Sonuç.....	32

BÖLÜM 1

GİRİŞ

1.1. LİTERATÜR ÖZETİ

Son yıllarda Bilgisayarların gelişmesi ve teknolojik yeniliklerin artmasıyla günlük hayatta yapılabilen birçok işin insan hatasından arındırılmış bir şekilde bilgisayarlara yaptırıldığını görmekteyiz. Bilgisarlar aracılığıyla yapılan işlerdeki en büyük fayda işlemlerin hem daha hızlı yapılması hem de hata oranının minimuma çekilerek olası zararlardan kurtulunmasıdır. Bu konu ile ilgili yapılan bazı çalışmalar alttaki gibidir.

Recep HOLAT’ a ait olan “YÜZ BULMA VE TANIMA SİSTEMLERİ KULLANARAK KİMLİK TESPİTİNİN YAPILMASI” isimli makalede kimlik tespitine yönelik mevcut yaklaşımlar, yüz tanıma da ki zorluklar ve yüz tanıma teknikleri konularına değinilmiştir.

Kontrol ve Bilgisayar Müh. Erkan SÜTÇÜLER’ e ait olan “GERÇEK ZAMANLI VIDEO GÖRÜNTÜLERİNDEN YÜZ BULMA VE TANIMA SİSTEMİ” isimli makalede yüz bulmaya yönelik mevcut yöntemlerin tanıtılması, yüz tanıma da kullanılan tekniklerin uygulamasının gösterilmesi ve tasarlanan sistemin deneysel sonuçlarına değinilmiştir.

Ghulam Sakhi SHOKOUH’ a ait olan “GERÇEK ZAMANLI SAYISAL GÖRÜNTÜ İŞLEME VE ÖRÜNTÜ TANIMA TEKNİKLERİNİN ARAŞTIRILMASI VE UYGULANMASI” isimli makalede yüz bulmaya yönelik mevcut matematiksel yaklaşımların tanıtılması, bu yaklaşımların uygulanması ve çıkan sonuçların karşılaştırılması konularına değinilmiştir.

1.2. PROJENİN AMACI

Yüz tanıma sistemleri artık hayatımızın büyük bir bölümünde yer almaktadır. Gelişen teknoloji ve teknikler ile bu sistemler cep telefonlarımıza kadar inmiş bulunmaktadır. Verilerimizin ve kişisel bilgilerimizin gizliliğini korumak amacıyla eskiden şifre ve örüntü gibi bazı yöntemler kullanılmaktaydı. Bunların yeterli olmaması ile birlikte daha gelişmiş teknolojiler ihtiyaç duymaya başladık. Verilere ulaşacak kişinin doğru kişi olduğunu anlayabilmek için kullandığımız gelişmiş teknolojilerin başında kişiye özgü olan parmak izi, iris ve yüz gibi bazı başka kimsede bulunmayan fiziksel özelliklerin algoritmalar ile işlenerek doğrulama yapması amaçlanır.

Projemizde yüz tanımaya dayalı yöntemler kullanarak kişi doğrulaması yapılmaktadır. Bunun bir VR gözlük yardımıyla güvenlik güçlerinin giyebileceği ve gözlüğün içinde bulunan ekranda önceden eğitilmiş ve hazırlanmış veritabanını kullanarak yüz tanıma yapılması amaçlanmıştır.

Bir gözlük yardımı ile yapılmasının amacı ise güvenlik güçlerine hareket esnekliği sağlamak ve suçlulara anında müdahale edebilme çevikliği kazandırmaktır.

Kullanılacak gözlük havaalanı ve buna benzer birçok geçiş noktasında güvenliği sağlamak ve izinsiz girişleri önleyebilmek amacıyla dizayn edilmiştir.

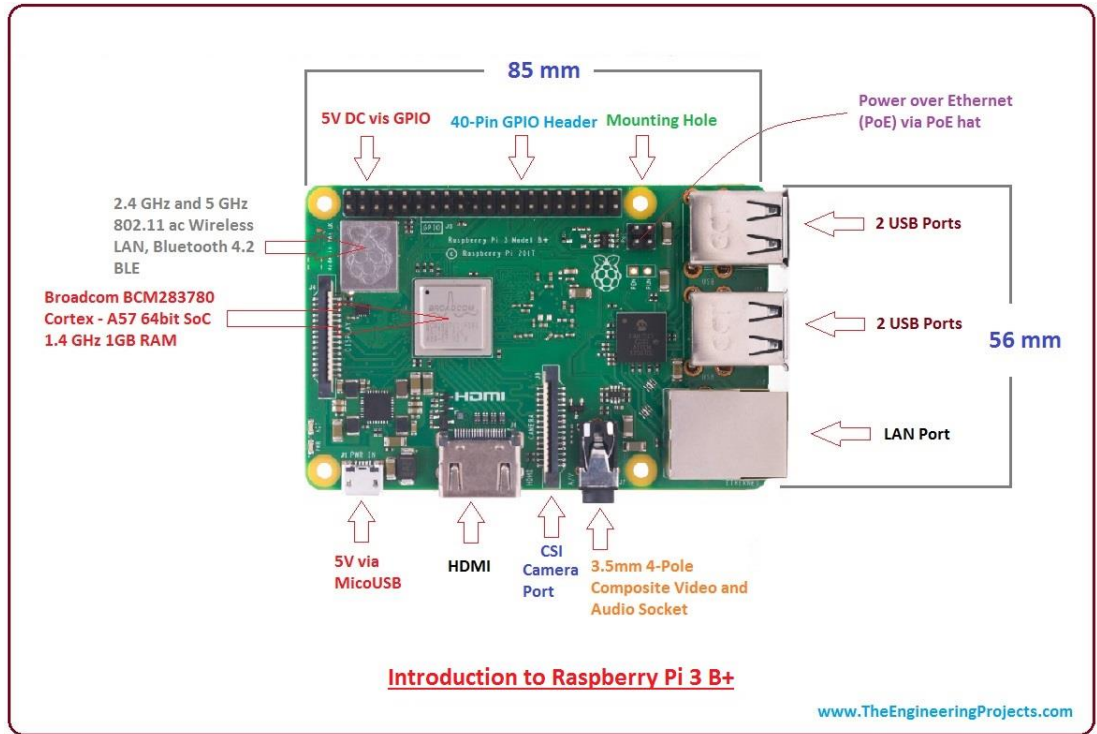
BÖLÜM 2

GÖZLÜK DİZAYNINDA KULLANILAN MALZEMELER

2.1. Raspberry Pi NEDİR?

Raspberry Pi , İngiltere’de kurulmuş olan Raspberry Pi Vakfı tarafından fonlanan; öğrenci ve amatör kullanımlar için tasarlanmış kredi kartı büyüklüğünde, tek bir board’dan oluşan mini bilgisayardır. Kendisine ait çok işlem yeteneği gerektirmeyen işler için Raspian adında linux tabanlı bir işletim sistemi vardır.

2.1.1. Raspberry Pi 3B+



Şekil 2. 1.Raspberry Pi 3B+

Kullanılan Raspberry Pi 3 B+ Modelinin Teknik Özellikleri Şunlardır :

- **İşlemci:** Dört Çekirdek 64-bit 1.4GHz A53/ARMv8
- **RAM:** 1GB LPDDR2 SDRAM'e sahiptir.
- **Wireless LAN:** Çift bant 2.4 + 5 GHz 802.11.b/g/n/ac
- **Ethernet:** 300Mbps Gigabit, PoE HAT uyumlu
- **Bluetooth:** 4.2, Düşük Enerji
- **GPIO:** 40 pinli header, güç giriş pinleri
- **Hafıza:** Micro SD Kart girişi bulunmaktadır.
- **Video:** HDMI, DSI ekran portu + CSI kamera portu vardır.
- **Ses:** 4 kutuplu 3.5mm ses + kompozit video portu
- **USB:** 4 adet USB 2.0 + MikroUSB 5V/2,5A güç girişi
- **Multimedya:** H.264, MPEG-4 1080p@30, OpenGL 2.0

2.1.2. Raspberry Pi 3B+ İle Neler Yapılabilir ?

Bu board ile bazı yapılabilecekler şunlardır. Bunlar :

- **Desktop(PC) :** Cihaza işletim sistemini yükledikten sonra kolay bir şekilde usb klavye-mouse ve de HDMI üzerinden bir ekran bağlayarak kişisel bir bilgisayar olarak kullanmak mümkündür.
- **Web Server :** Raspian distrosunu kurup kablolu internet ya da kablosuz olarak internet bağlantısını yaptıktan sonra tek yapılması gereken repository den web server indirip kurmak.
- **Ağ Kamerası/NVR :** Raspberry de bulunan CSI portu ile kendine ait kamerası veya USB'den bağlayabileceğiniz başka bir kamera ile cihazınızı bir ağ kamerası ve kayıt aracına dönüştürmek çok basittir.

Bunların dışında yapılan birçok uygulama mevcuttur.

2.2. 5 inç Taşınabilir Dokunmatik Ekran



Şekil 2. 2.Ekran

Kullanılan Ekranın Teknik Özellikleri Şunlardır :

- USB kapasitif dokunmatik ekran
- Çözünürlük 800x480
- HDMI ve HDMI ses çıkışı destekler.
- Ultra düşük güç modu.
- 5 Nokta kapasitif dokunma
- Boyutları 121 x 95 x 13 mm

Kullanılan bu ekranın en büyük artıları düşük güç modunda çalışabilmesi, boyutunun yaklaşık bir telefon boyutunda olması ve taşınabilir olması.

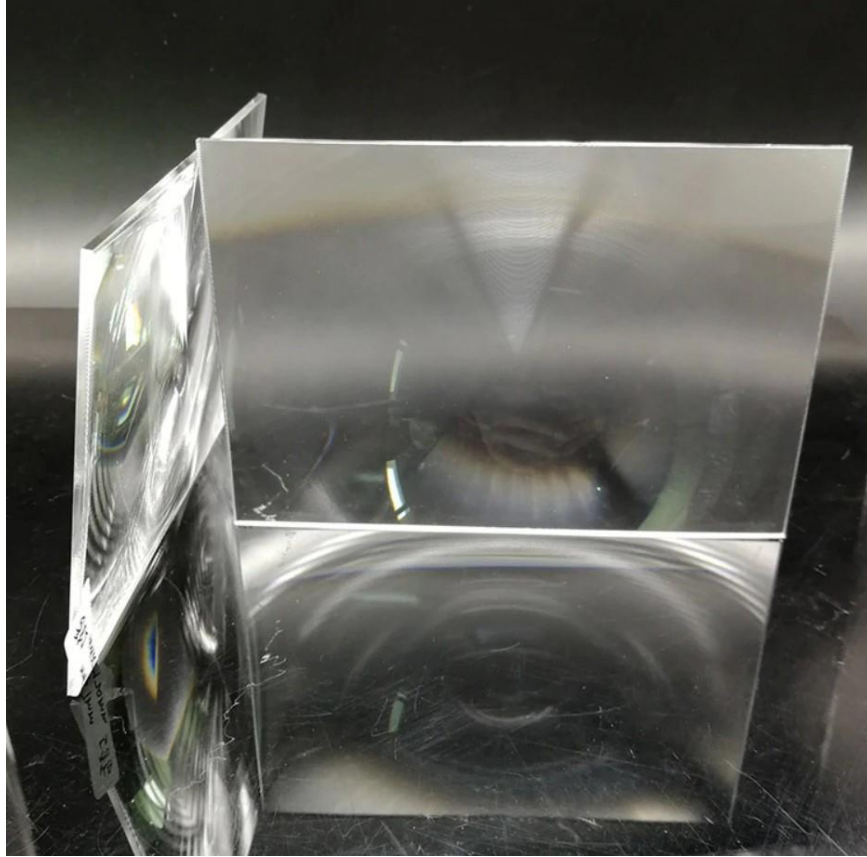
2.3. Güç Modülü



Şekil 2. 3.Güç Modülü

Bu güç modülü özellikle Raspberry Pi için tasarlanmıştır. Pilin kapasitesi 3800 mAH olmakla birlikte çıkış akımı 1.8A dir. Bu modülden çıkan gerilim ise 5.1 V dur. Çift USB çıkışı ile hem ekranı hemde Raspberry Pi yi besleyebilmektedir. Micro Usb ile şarj edilebilmektedir. Uzun pil ömrüne sahip olmakla birlikte enerji verme süresi oldukça yüksektir. En büyük avantajı ise Raspberry Pi boyutlarında olması ve herhangi bir güç kaynağına bağlı olmadan çalışabilmesidir.

2.4. Fresnel Lens



Şekil 2. 4.Fresnel Lens

Fresnel Lens in kullanım amacı ise ekran ile kullanıcı arasındaki göz mesafesini ayarlamaktır. Ekran insan gözüne yaklaştıkça odaklanması zorlaşır. Fresnel Lens bir büyüteç gibi karşısındaki nesneyi büyütür ve bakışı kolaylaştırır.

Özellikleri Şunlardır:

- **Kalınlık** : 2 mm
- **Odak Uzunluğu** : 185/120 mm
- **Malzeme** : PMMA (Akrilik)

2.5. Everest VR-0023 3D SANAL GERÇEKLİK GÖZLÜĞÜ



Şekil 2. 5.VR Gözlük

Bu gözlüğü kullanma amacımız ise ekranı kamerayı ve Raspberry Pi yi rahatça konumlandırabileceğimiz giyilebilir bir gözlük olmasıdır. 4-5.5 inç boyutunda bulunan her türlü ekranı kolayca yerleştirebiliyoruz. Ayarlanabilir kafa bandı sayesinde gözlüğün sabit durması kolaylaşıyor.

2.6. Ara Uzatma Parçası

Bu parçayı modelleyerek 3D yazıcı ile bastırdık. Parçanın yapılış amacı ise kullandığımız fresnel lensin belli bir odak mesafesi olması yani ekranı odaklaması için belli bir mesafede bulunmasıdır. Bu ara uzatma parçası ile gözlüğümüzü uzatmış olduk.



Şekil 2. 6.Ara Uzatma Parçası

2.7. Raspberry Pi Model B Kamera Kiti

for Raspberry Pi 3 Model B+/3B/2B



Şekil 2. 7.Raspberry Pi Kamerası

Raspberry Pi kamerası Raspberry Pi Board üzerinde bulunan CSI portuna direk bağlanabilmektedir. İşletim sistemi üzerinden kamera aktif edilince direk kullanılabilir.

Özellikleri Şunlardır:

- **Uygunluk** : Tüm Raspberry Pi sürümlerini destekler.
- **Gece Görüş** : 2 infrared led eklenti bağlanarak gece görüşü elde edilebilir.
- **Odak** : Ayarlanabilir odak özelliği bulunmaktadır.

BÖLÜM 3

PROJENİN YAZILIM VE TASARIM KISMININ HAZIRLANMA AŞAMALARI

Projemiz temel olarak 3 aşamada anlatılacaktır. Masaüstü uygulamasının arayüzünün ayarlanması, yüz tanıma sürecinde kullanılan tekniklerin anlatımları ve fiziksel tasarım.

3.1.Masaüstü Uygulamasının Hazırlanması

3.1.1. Ana Pencere Özelliklerinin Belirlenmesi

Uygulamayı yazarken python için özel olarak oluşturulmuş bir GUI(Grafiksel Kullanıcı Arayüzü) modülü olan Tkinter Modülü kullanılmıştır.

```
class SahinGoz(tk.Tk):  
    def __init__(self,*args,**kwargs):  
        tk.Tk.__init__(self,*args,**kwargs)  
        tk.Tk.wm_title(self,"Şahin Göz") # üst pencere program adı  
        tk.Tk.wm_geometry(self,'1024x700')  
        tk.Tk.resizable(self,width=False,height=False)  
        tk.Tk.wm_attributes(self,"-alpha",0.97)  
        tk.Tk.iconbitmap(self,"hawk.ico")  
  
        container=tk.Frame(self)  
        container.pack(side="top",fill="both",expand=True)  
  
        container.grid_rowconfigure(0,weight=1)  
        container.grid_columnconfigure(0,weight=1)  
  
        self.frames={}  
  
        for F in (Giris,Kayit):  
            frame = F(container, self)  
  
            self.frames[F]= frame  
            frame.grid(row=0,column=0,sticky="nsew")  
  
        self.show_frame(Giris)  
  
    def show_frame(self,cont):  
        frame=self.frames[cont]  
        frame.tkraise()
```

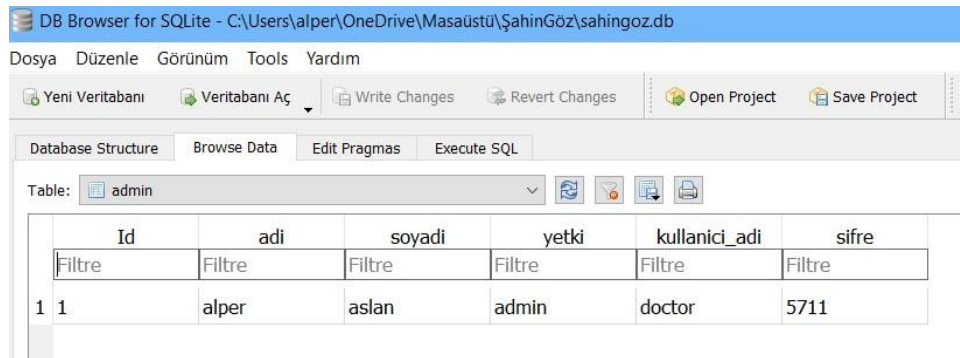
Şekil 3. 1.Program Ekran Dizaynı

Şahin Göz adlı uygulamamız bir class tanımlanarak oluşturuldu . def __init__ adlı fonksiyon program ilk çalıştırıldığı zaman çalışacak olan ilk fonksiyondur. Bu fonksiyonda tkinter kütüphanesinin bize sağladığı bazı metotlar sayesinde programın açılış ekranı ayarlandı. Metotların işlevlerini şöyle sıralayabiliriz :

- **wm_title** : Uygulama başlığını yazmak için kullanılır.
- **wm_geometry** : Pencerenin ekran boyutlarının ayarlanmasını sağlar.
- **Resizable()** : Pencerenin ayarlanabilir olup olmadığını belirler.
- **wm_attributes** : Pencerenin saydamlık ayarını sağlar.
- **Iconbitmap()** : Uygulamaya ikon eklemeye yarar.
- **Tk.Frame()** : Pencere oluşturulması için bir frame nesnesi oluşturur.
- **Pack ()** : Bu geometri yöneticisi, widget'ları üst widget'e yerleştirmeden önce bloklar halinde düzenler.
- **Grid_row&column configure** : Satır ve sütunlarının düzenlenmesini sağlar.
- **Def show_frame:** Kendisine parametre olarak gelen frameleri kodda belirtilen ölçülerde olacak şekilde otomatik ayarlar ve frameler arası geçişi sağlar.

3.1.2. Giriş Ekranı Tasarımı Ve Veritabanı Bağlantısının Kurulması

İlk önce veritabanı oluşturuldu. Veritabanı oluşturmak için SQLite kullanıldı. Belli bir kurulum gerektirmeden direk dosya üzerinden verileri aldığı için sql lite tercih edildi. Veritabanında adminin ait bilgiler yer alacak şekilde admin tablosu ve alan adları oluşturuldu.



The screenshot shows the DB Browser for SQLite interface. The title bar indicates the database file is 'C:\Users\alper\OneDrive\Masaüstü\ŞahinGöz\shahingoz.db'. The 'Database Structure' tab is active, showing a table named 'admin'. The table has six columns: Id, adi, soyadi, yetki, kullanıcı_adi, and sifre. The data row shows Id: 1, adi: alper, soyadi: aslan, yetki: admin, kullanıcı_adi: doctor, and sifre: 5711.

Id	adi	soyadi	yetki	kullanıcı_adi	sifre
1	alper	aslan	admin	doctor	5711

Şekil 3. 2. Admin Tablosu Oluşturma

```

def dict_factory(cursor, row):
    d = {}
    for idx,col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

vt=sqlite3.connect('sahingoz.db')
vt.row_factory=dict_factory
cur=vt.cursor()
sql="SELECT * FROM admin"
cur.execute(sql)
veriler=cur.fetchall()
print(veriler)

```

Şekil 3. 3. Veritabanı Bağlantısı Oluşturma

Yukarıda görüldüğü üzere dic_factory adlı fonksiyonda veritabanından verilerin dictionary şeklinde alınması sağlandı. Normalde verileri tuple olarak aldığı için üzerinde değişiklik yapılamıyordu. Bu fonksiyon yardımıyla dictionary veri tipinde veri alma imkanı elde edildi.

- Connect() fonksiyonu ile bağlanılmak istenen veritabanına bağlandı.
- Vt.row_factory ile verilerin dictionary veri tipinde alınacağı belirtildi.
- Cursor ile veritabanında gezme imkanına sahip olundu.İmleç görevi görüyor.
- Sql sorgusu ile admin tablosundan veriler alındı ve execute metodu yardımıyla programın içine bilgiler aktarıldı.
- Cur.fetchall() diyerek tüm bilgilerin alınmak istendiği belirtildi. Ve en sonunda verilerin doğru gelip gelmediğini anlamak için ekrana yazdırıldı.

```

yazi=Label(self,justify=CENTER,text="Şahin Göz",fg="black",font=("Open Sans","60","bold"))
yazi.place(x=330,y=20)

kullanici_adi=Label(self,text="Kullanıcı Adı :",font=("Open Sans","10","bold")).place(x=350,y=390)
ekullanici_adi=Entry(self,bd=2,width=30)
ekullanici_adi.place(x=460,y=390)

sifre=Label(self,text="Şifre :",font=("Open Sans","10","bold")).place(x=350,y=420)
esifre=Entry(self,bd=2,width=30,show="*")
esifre.place(x=460,y=420)

giris=Button(self,text="Giriş",command=admin_kontrol,font=("Open Sans","10","bold"),bg="cyan",
             justify=CENTER,width=10,relief=RAISED).place(x=500,y=450)

```

Şekil 3. 4. Giriş Ekranı Form Oluşturma

Kullanıcıdan giriş bilgilerini almak için bir form oluşturuldu. Label metodu ile kullanıcıya giriş yapacağı alanların adları gösterildi. Place() metodu ile bu label'in ekranda hangi konumda olacağı belirlendi. Entry metodu ile de kullanıcıdan veri alınması sağlandı. Button metodu ile de kullanıcından alınan bu bilgilerin butona tıklandığında admin_kontrol fonksiyonuna gönderilmesi sağlandı.

```
class Giris(tk.Frame):
```

```

def __init__(self,parent,controller):
    def admin_kontrol():
        # print("Clicked")

        kullanici_adi = ekullanici_adi.get()
        sifre = esifre.get()

        print(kullanici_adi, sifre)
        sql="SELECT * FROM admin WHERE Id=1"
        cur.execute(sql)
        sorgu=cur.fetchone()

        print(sorgu)

        if kullanici_adi == sorgu['kullanici_adi'] and sifre == sorgu['sifre']:
            tm.showinfo("Login info", "Hoşgeldin Alper")
            controller.show_frame(Kayit)
        else:
            tm.showerror("Login error", "Hatalı Kullanıcı adı veya Şifre")

    tk.Frame.__init__(self,parent)

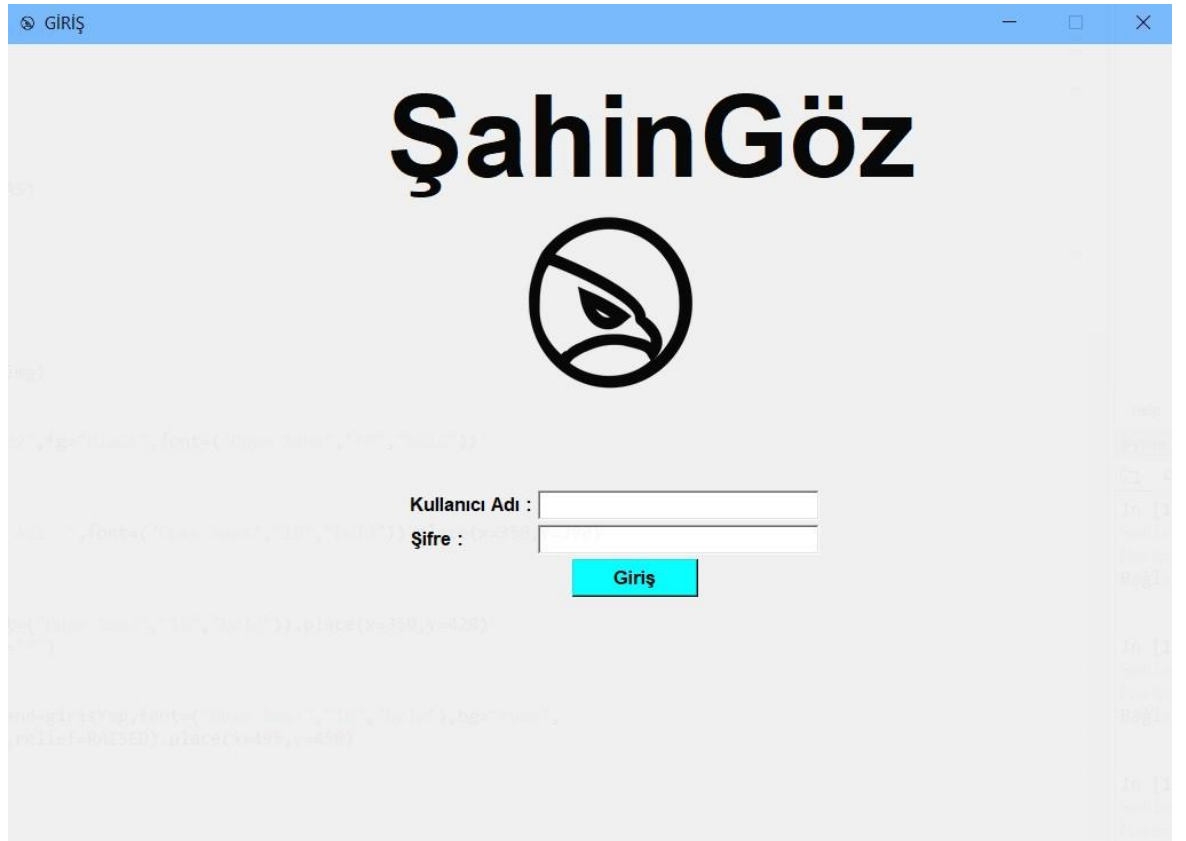
```

Şekil 3. 5. Admin Bilgi Kontrolü Fonksiyonu

Def__init__ fonksiyonunu çağırarak istenilen ekranın önceden belirtilmiş olan ekran boyutlarında olması sağlandı.

Get() fonksiyonu kullanılarak formdan gelen bilgilerin fonksiyona aktarılması sağlandı. Sql sorgusu yapılarak admin tablosundaki yetkili kişinin bilgileri getirildi. If bloğunda ise formdan gelen bilgiler ile veritabanında kayıtlı olan admin bilgilerinin uyuşup uyuşmadığı kontrol edildi. Showinfo ve showerror metotlarıyla da giriş bilgileri ekrana yazdırıldı.

Sonuç olarak kullanıcının kullanıcı adını ve şifresini girerek sisteme giriş yapabileceği bir giriş sayfası tkinter modülü yardımıyla oluşturuldu.



Şekil 3. 6. Giriş Ekranı

3.1.3. Ana Ekranın Tasarımı , Verilerin Listelenmesi ve CRUD İşlemleri

Eğer giriş yapan kişi doğru bilgileri girmişse ana ekrana yani suçlu kayıt ekranına ulaşacaktır.

Table: suçlular

	Id	adi	soyadi	durum	tcno
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Alper	ASLAN	Admin	28278
2	2	Robert	Downey Jr.	Araniyor	212
3	3	Mert	Ekinci	Araniyor	789
4	4	Suayb	Kiris	Hukumlu	75756

Şekil 3. 7. Suçlu Veritabanının Oluşturulması

İlk önce veritabanı oluşturuldu. Olmasını istenen ilk bilgi kişiyi eşsiz şekilde tanımlayabilecek ve otomatik artacak olan Id sütunudur. Daha sonra kişiye ait ad , soyad, suçluların o anki durumlarını gösteren durum verisi ve tcno bilgilerinin alanları veritabanında oluşturuldu.

```
Ladi=Label(self,text="Adı :",font=("Open Sans","10","bold")).place(x=10,y=10)
Eadi=Entry(self,bd=2,width=25)
Eadi.place(x=100,y=10)

Ltcno=Label(self,text="Tc No :",font=("Open Sans","10","bold")).place(x=280,y=10)
Etcno=Entry(self,bd=2)
Etcno.place(x=330,y=10)

Lsoyadi=Label(self,text="Soyadı :",font=("Open Sans","10","bold")).place(x=10,y=40)
Esoyadi=Entry(self,bd=2,width=25)
Esoyadi.place(x=100,y=40)

Ldurum=Label(self,text="Durum :",font=("Open Sans","10","bold")).place(x=10,y=70)
Edurum=ttk.Combobox(self,width=22,values=['Araniyor','Hukumlu','YurtDisi Yasagi'])
Edurum.place(x=100,y=70)
```

Şekil 3. 8. Ana Ekran Label Ve Entry Oluşturulması

Sisteme kaydedilmek istenen suçlunun adını, soyadını, TC numarasını ve de suçlunun aranma mı yoksa hükümlümü vb. hangi durumda olduğunu gösterecek durum bilgisinin kaydedilmesini sağlayacak label ve entry alanlarının oluşturulması sağlandı. Diğerlerinden farklı olarak sadece durum bilgisi açılabilir kutu menüsü yani combobox ile gösterildi.


```

kaydet=Button(self,text="EKLE",command=kaydet).place(x=100,y=100)
guncelle=Button(self,text="GUNCELLE",command=guncelle).place(x=160,y=100)

silme=Button(self,text="SİLME",command=sil).place(x=260,y=100)

foto_yukle=Button(self,text="Fotoğraf Yükle",command=foto_yukle,width="15",
,height="3",background="cyan").place(x=350,y=40)

foto_goster=Button(self,text="Fotoğrafları Göster",command=foto_goster,width="15",
,height="3",background="DarkOliveGreen2").place(x=500,y=40)

yuz_tespitet=Button(self,text="Yüzleri Tespit Et",command=yuz_bul,width="15",
,height="3",background="dark turquoise").place(x=650,y=40)

egitme=Button(self,text="Eğit",command=egitme,width="15",
,height="3",background="DarkOrange1").place(x=800,y=40)

```

Şekil 3. 9. Butonların Yerleştirilmesi

Yukarıda ki şekilde görüldüğü üzere pencereye bilgilerin kayıt edilmesi için ekle, güncelle ,silme butonlarını eklendi. Diğer butonlarda kayıtlı kişiye fotoğraf ekleme ve fotoğraflar üzerindeki işlemler ile alakalıdır.Daha sonra ayrıntılı olarak anlatılacaktır.

```

liste=ttk.Treeview(self) # listeleme için
liste["columns"]=("sut1", "sut2", "sut3", "sut4")
liste.place(x=0,y=140)
liste.heading("#0",text="Id")
liste.heading("sut1",text="Ad")
liste.heading("sut2",text="Soyad")
liste.heading("sut3",text="Tc No:")
liste.heading("sut4",text="Durum")

liste.bind('<ButtonRelease-1>',datagetir) # tıklanan veriyi alıp fonksiyona gönderme
liste()

```

Şekil 3. 10. Ekranda Listeleme

Tkinter modülünün sağlamış olduğu treeview özelliği kullanılarak kayıtlı olan kişiler listelenmiş oldu. Liste 4 kolona ayrıldı. Ve gerekli bilgiler bu kolonlarda gösterildi. Butona tıklandığı zamanda veriler otomatik olarak entryler üzerinde gösterilmiş oldu.

```
def kaydet():
    sql="INSERT INTO suclular (adi,soyadi,tcno,durum) VALUES ('%s','%s','%s','%s')" %(Eadi.get()
    |,Esoyadi.get(),Etcno.get(),Edurum.get())
    cur.execute(sql)
    vt.commit() # CRUD İŞLEMLERİ İÇİN COMMIT KULLANIRIZ.
    mesaj('Kaydetme')
    listele()
```

Şekil 3. 11. Suçlu Kaydetme

Oluşturulan bilgi girişi kutucuklarında girilen bilgileri get() fonksiyonuyla entrylerden alınarak sql sorgusuyla veritabanına eklendi. Ve kaydetme uyarısı verildi.

Şekil 3. 12. Suçlu Bilgi Giriş Bölümü

Aşağıdaki şekilde veritabanına kaydedilen suçluların listelenmesi gösterilmiştir. Veritabanından suçluların tüm bilgileri sql sorgusuyla çekilip listelendi.

```
def listele():
    liste.delete(*liste.get_children()) # treeview i temizle
    sql="SELECT * FROM suclular"
    cur.execute(sql)
    results=cur.fetchall()
    for rs in results:
        liste.insert("",0,text=rs['Id'],values=(rs['adi'],rs['soyadi'],rs['tcno'],rs['durum']))
```

Şekil 3. 13. Suçlu Bilgileri Listeleme Fonksiyonu

Id	Ad	Soyad	Tc No	Durum
4	Suayb	Kiris	75756	Hukumlu
3	Mert	Ekinci	789	Aranıyor
2	Robert	Downey Jr.	212	Aranıyor
1	Alper	ASLAN	28278	Admin

Şekil 3. 14. Suçlu Listesi

```
def datagetir(event):
    idno=liste.item(liste.selection()[0])['text'] # listeden id değerini alıyor
    sql="SELECT * FROM suclular WHERE Id=%s" % (idno)
    cur.execute(sql)
    results=cur.fetchone()

    Eadi.delete(0, 'end')
    Eadi.insert(0,results['adi'])

    Esoyadi.delete(0, 'end')
    Esoyadi.insert(0,results['soyadi'])

    Etcno.delete(0, 'end')
    Etcno.insert(0,results['tcno'])

    Edurum.delete(0, 'end')
    Edurum.insert(0,results['durum'])
```

Şekil 3. 15. Entryler Üzerinde Veri Getirilmesi

Yazılan bu fonksiyonun amacı ise kayıtlar arasında gezerken bir kişinin kaydını entryler üzerinde otomatik olarak göstermektir. Başka bir kişiye tıkladığında üst bölümde bilgileri veritabanından otomatik olarak çekilir ve gösterilir. Böylece güncelleme gibi işlemler kolaylaşır.

```
def guncelle():
    idno=liste.item(liste.selection()[0])['text']
    sql="UPDATE suclular SET adi='%s',soyadi='%s',tcno='%s',durum='%s' WHERE Id=%s" %(Eadi.get(),
    Esoyadi.get(),Etcno.get(),Edurum.get(),idno)

    cur.execute(sql)
    vt.commit() # CRUD İŞLEMLERİ İÇİN COMMIT KULLANIRIZ.
    mesaj('Guncelleme')
    liste()
```

Şekil 3. 16. Suçlu Bilgileri Güncelleme Fonksiyonu

Güncellemek istenilen kişinin üzerine tıkladığında veriler datagetir fonksiyonu yardımıyla otomatik olarak gösterildi. Değişiklik yapıldığında verilerin güncellenmesi istenirse güncelle butonuna tıklayıp sql sorgusuyla yeni bilgiler veritabanına gönderilir. Ve güncellendi uyarısı ekranda gösterilir.

Id	Ad	Soyad	Tc No	Durum
4	Suayb	Kiris	75756	Hukumlu
3	Mert	Ekinci	789	Araniyor
2	Robert		212	Araniyor
1	Alper		28278	Admin

Şekil 3. 17. Suçlu Bilgileri Güncelleme

```
def sil():
    idno=liste.item(liste.selection()[0])['text']
    sql="DELETE FROM suclular WHERE Id=%s" %idno
    cur.execute(sql)
    vt.commit() # CRUD İŞLEMLERİ İÇİN COMMIT KULLANIRIZ.
    mesaj('Silme')
    listele()
```

Şekil 3. 18. Silme Fonksiyonu

Treeview olarak oluşturulan listede 0.sütundan kişinin Id bilgisi alındı. İlk önce silinmek istenen kişinin bilgisine tıklandı sonra sil diyerek bu Id bilgisine ulaşıldı. Sql sorgusu ile veritabanında Id si eşlesen kayıt silindi. Ve ekranda uyarı gösterildi. Ve tekrardan listele fonksiyonunu çağırılarak liste yenilendi.

3.1.4. Suçlu Fotoğraf Ekleme , Fotoğrafların Gösterilmesi, Fotoğraflar Üzerinde Yüz Tespitinin Yapılması ve Eklenen Fotoğrafların Eğitilmesi

3.1.4.1.Suçlu Fotoğraf Ekleme

Sisteme eklenmek istenen suçlunun kişisel bilgileri girilip veritabanına kaydedildikten sonra o kişi için fotoğraf eklenir.

```
def foto_yukle():
    idno=liste.item(liste.selection()[0])['text']
    filez = filedialog.askopenfilenames(parent=self,title='Choose a file')
    lst = list(filez)
    path = "suclu_foto/%d" % idno

    pathexist=os.path.exists(path)
    if not pathexist:
        os.makedirs(path)

    for item in lst:
        filename = os.path.basename(item)
        shutil.copy(item, path)
    tm.showinfo("İşlem Tamamlandı","Fotoğraflar yüklendi.")
```

Şekil 3. 19. Fotoğraf Ekleme Fonksiyonu

Seçilen kişinin Id si listeden alındı. Tkinter da bulunan filedialog kütüphanesi kullanılarak suçluya ait fotoğrafların dosya yolları bir liste oluşturularak listeye

kopyalandı. Suçlunun Id numarası ile suçlu fotoğraflarının alt klasöründe bir klasör oluşturuldu. Eğer aynı yol önceden varsa bu işlem atlandı yoksa klasör oluşturuldu. Listede bulunan her bir dosya döngüye sokularak shutil modülü yardımıyla belirtilen path in içine kopyalandı. İşlem tamamlandı bilgisi verildi.

3.1.4.2.Suçlu Fotoğraf Gösterme

```
def foto_goster():  
    idno=liste.item(liste.selection()[0])['text']  
    path="suclu_foto/%d" % idno  
    if os.path.exists(path):  
        path=os.path.realpath(path)  
        os.startfile(path)  
    else:  
        tm.showinfo("Klasör Hatası", "Fotoğraf yok.İlk önce fotoğraf yükleyiniz.")
```

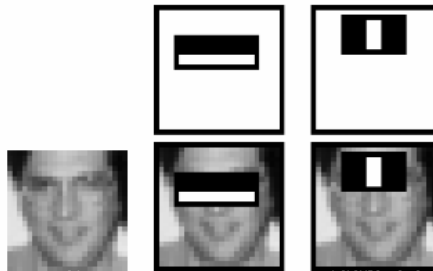
Şekil 3. 20. Fotoğrafları Gösterme Fonksiyonu

Suçluya ait fotoğraflar görmek istenirse o kişinin üzerine tıklanıp fotoğrafları göster butonuna tıklanılır.Listeden kişinin Id bilgisi alınır ve o Id numarasına ait bir klasör olup olmadığı kontrol edilir. Klasör varsa içinde dosya olup olmadığı OS kütüphanesi kullanılarak kontrol edilir. Eğer dosya varsa startfile diyilerek suçluya ait klasör açılır ve fotoğraflar gösterilir. Eğer yoksa uyarı verilir.

3.1.4.3.Fotoğraflar Üzerinde Yüz Tespitinin Yapılması

Yüklenen fotoğrafların sadece yüz kısımlarının alınması gerekmektedir. Fotoğraflar içerisindeki yüz kısmının tespit edilmesi için Haar Cascade yöntemi kullanıldı.

Haar Cascade basitçe anlatılırsa fotoğraftaki koyu ve açık renk piksellerini kullanarak kaş, burun ve ağız gibi organların tespit edilmesiyle yüzdeki organların yerlerinin ve yüzün tespit edilmesini sağlar.



Şekil 3. 21. Haar Cascade Yöntemi Anlatımı


```

def yuz_bul():
    idno=liste.item(liste.selection()[0])['text']

    yuz_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    goz_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

    DIR = r"C:\Users\alper\OneDrive\Masaüstü\ŞahinGöz\sucflu_foto\%d" % idno
    fotoSayisi = len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name))])
    foto_icindeki_yuzler = 0
    list=os.listdir(DIR)
    for pic in range(0, (fotoSayisi)):

        foto = cv2.imread("sucflu_foto/"+str(idno) + "/" + list[pic] + ".jpg")

```

Şekil 3. 22. Klasörden Fotoğrafların Alınması ve Okunması

Yüz bulma fonksiyonu parça parça anlatılacaktır. Kullanılan kütüphane OpenCv kütüphanesidir. Yüklenen fotoğraflardan yüzlerin bulunması istenilen kişiye tıklanılarak Id numarasını alındı. Yüz ve gözlerin tespiti için önceden birçok yüz verisi kullanılarak eğitilmiş olan haarcascade ön yüz tespiti ve haarcascade göz tespiti xml dosyaları alındı. Cascade sınıflandırıcısı ile cascade dosyaları yuz_cascade ve goz_cascade olmak üzere alındı. Bu değişkenler daha sonra döngüye sokularak yüz ve göz tespitinde kullanılacaktır.

Yüzlerinin tespit edilmesi istenen suçlunun klasör yoluna suçlunun Id si kullanılarak erişildi ve DIR değişkenine atandı.

OS kütüphanesi kullanılarak klasör içindeki fotoğrafların sayısı bulunarak fotoSayisi değişkenine atandı. List değişkenine belirtilen yoldaki dosyaların yolları kopyalandı.

For döngüsü yardımıyla yüzlerin bulunması için cv2 kütüphanesi içindeki imread özelliği kullanılarak klasördeki tüm dosyalar okunup foto değişkenine atandı.

```

gri = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY)
yuzler = yuz_cascade.detectMultiScale(gri, scaleFactor=1.05, minNeighbors=5
, flags = cv2.CASCADE_SCALE_IMAGE ) # minSize=(15, 15) eklenebilir.

foto_icindeki_yuzler = 0
for (x,y,w,h) in yuzler:
    yuz_ve_goz_tespitedilen = 0
    roi_gri = gri[y:y+h, x:x+w]
    gozler = goz_cascade.detectMultiScale(roi_gri, scaleFactor=1.05, minNeighbors=5
, flags = cv2.CASCADE_SCALE_IMAGE ) # minSize=(5, 5) eklenebilir..

    goz_sayisi = 0
    for (ex,ey,ew,eh) in gozler:
        goz_sayisi = goz_sayisi + 1

    if goz_sayisi >= 2:
        yuz_ve_goz_tespitedilen = 1

    if yuz_ve_goz_tespitedilen > 0:
        cv2.imwrite("sucflu_foto/detected_faces/User." + str(idno) + "." + str(pic+1) + ".jpg", gri[y:y+h, x:x+w])
        print("Foto"+str(pic)+ " " +str(foto_icindeki_yuzler)+" işlendi ve kırpıldı.")
        foto_icindeki_yuzler += 1
        tespit_edilen_yuzler += 1

```

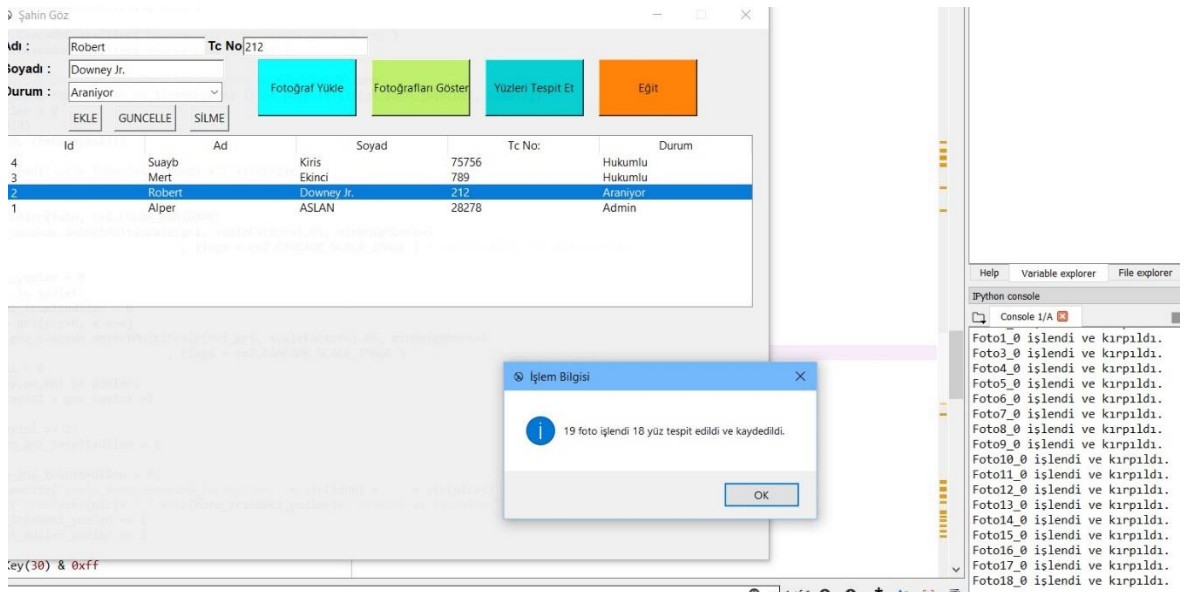
Şekil 3. 23. Fotoğraflardan Yüz ve Gözlerin Tespit Edilmesi

Okunan her bir fotoğraf cvtColor özelliğiyle griye çevrildi. Griye çevrilmesinin amacı iki kanallı fotoğraf elde etmektir. RGB yani renkli fotoğraflar üç kanallıdır. Griye çevirerek fotoğraflar siyah ve beyaz şeklinde iki kanal olacak şekilde çevirildi. Bu işlem fotoğraflardaki yüz tespit hız ve doğruluğunu arttırmaktadır. Daha önceden belirtilen yuz_cascade değişkeninde detectMultiScale metodu kullanılarak yüzlerin nasıl ve hangi ölçeklerde tespit edilmesi belirtildi.

ScaleFactor parametresi kullanılarak fotoğraf içindeki yüzün uzak ve yakın olarak bulunması sağlandı. Parametre değeri azaldıkça fotoğraf içindeki küçük yüzlerin bulunma oranı artacaktır. Takriben 1.05 ve 1.5 arası uygun değerlerdir.

MinNeighbors(minimum komşuluk) parametresi kısaca pikselleri art arda kontrol ediyor. Daha yüksek değerler daha az tespiti sebep olur.

Ölçeği ayarlandıktan sonra for döngüsüne sokularak yüzlerin ve gözlerin sayısı bulunur. Önce yüzlerin tespiti yapılır. Ön yüz bulunması istendiği için iki göz bulunması istendi. Eğer iki göz tespit edildiyse yuz_ve_goz_tespitedilen değişkeninin değeri arttırıldı. Bulunan her yüz imWrite metodu ile detected_faces klasörüne kullanıcı Id bilgisini içerecek şekilde isimlendirilerek kaydedildi. Kaydedilirken de yine bulunan yüzler griye çevrilerek kaydedildi. İşlenen fotoğrafların bilgisi ve fotoğraflardan kaç yüzün bulunduğu bilgisi kullanıcıya gösterildi. Fonksiyon sonlandırıldı.



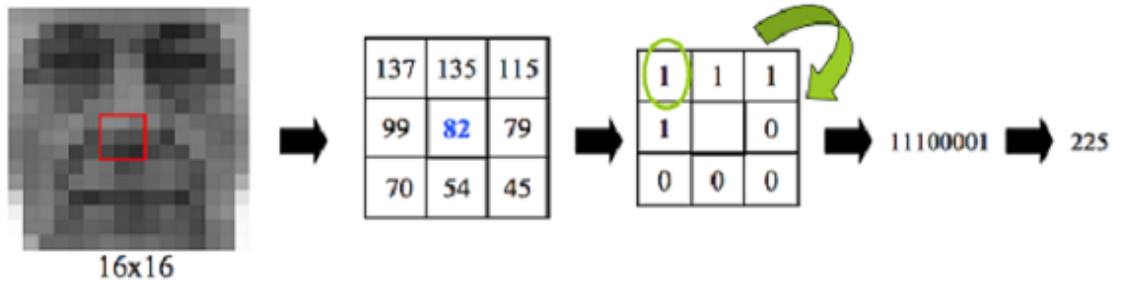
Şekil 3. 24. Fotoğraflardan Yüz ve Gözlerin Tespit Ekranı



Şekil 3. 25. Detected Faces Klasörü

3.1.4.4.Yüzlerin Eğitilmesi

Detecte Faces klasöründe tüm yüzler tespit edilerek toplandı. Sıra bu fotoğrafların eğitilme kısmına geldi. Eğitimde kullanılan algoritmanın adı LBPH (Local Binary Pattern Histogram) algoritmasıdır.



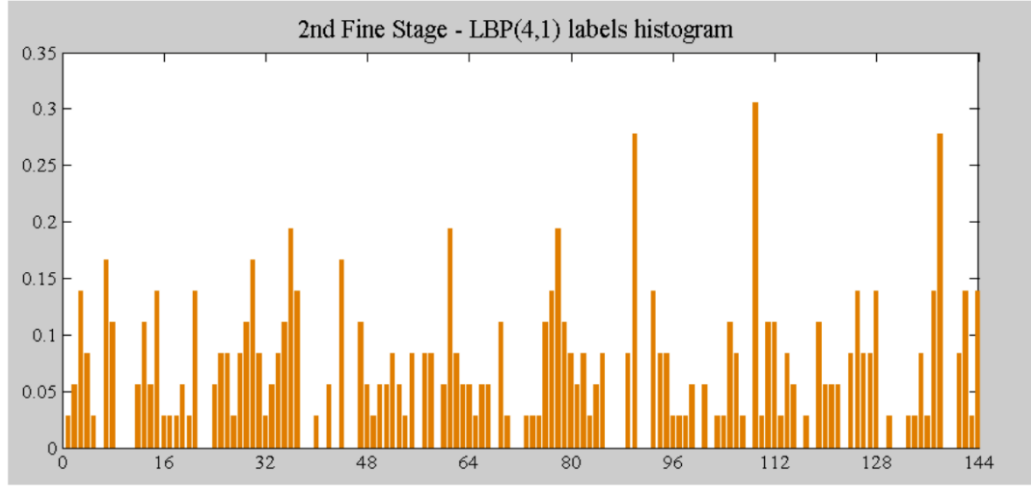
Şekil 3. 26. LBPH Çalışma Mantığı

Her blok için, LBP bir kerede 9 piksele (3 x 3 pencereye) bakar ve pencerenin merkezinde bulunan piksele özel bir ilgi gösterir.

Daha sonra, merkezi piksel değerini, 3 × 3 penceresinin altındaki her bir komşunun piksel değeri ile karşılaştırır.

Merkez piksele eşit veya daha büyük olan her bir komşu piksel için değerini 1'e, diğerleri için ise 0'a ayarlar.

Bundan sonra, güncellenen piksel değerlerini (0 veya 1 olabilir) saat yönünde okur ve ikili sayı oluşturur. Daha sonra, ikili sayıyı bir ondalık sayıya dönüştürür ve bu ondalık sayı, orta pikselin yeni değeridir. Bu bir bloktaki her piksel için yapılır. Ardından, her blok değerini bir histograma dönüştürür, böylece şimdi bir görüntüdeki her blok için bir histogram elde edilir, şunun gibi:



Şekil 3. 27. LBP Histogram

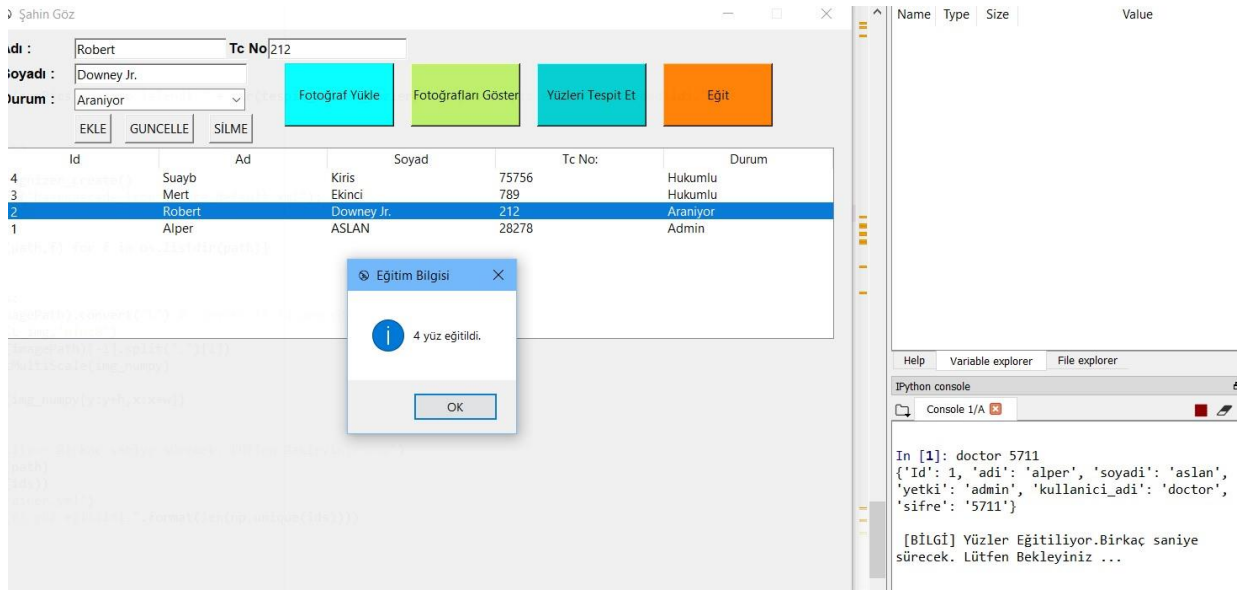
Son olarak, ilgililen tüm özellikleri içeren tek bir görüntü için tek bir özellik vektörü oluşturmak üzere bu blok histogramlarını birleştirir.

```
def egitme():
    path = 'succlu_foto/detected_faces'

    taniyici = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
    # fotoğrafları ve id yi almak için
    def getImagesAndLabels(path):
        imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
        yuzVerileri=[]
        ids = []
        for imagePath in imagePaths:
            PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
            img_numpy = np.array(PIL_img,'uint8')
            id = int(os.path.split(imagePath)[-1].split(".")[1])
            faces = detector.detectMultiScale(img_numpy)
            for (x,y,w,h) in faces:
                yuzVerileri.append(img_numpy[y:y+h,x:x+w])
                ids.append(id)
        return yuzVerileri,ids
    print ("\n [BİLGİ] Yüzler Eğitiliyor.Birkaç saniye sürecek. Lütfen Bekleyiniz ...")
    faces,ids = getImagesAndLabels(path)
    taniyici.train(faces, np.array(ids))
    taniyici.write('egitilenveri/trainer.yml')
    tm.showinfo("Eğitim Bilgisi","{0} yüz eğitildi.".format(len(np.unique(ids))))
```

Şekil 3. 28. Yüz Eğitme Fonksiyonu

Yukarıda Şekil 3.28 te gösterilen fonksiyon Eğit butonuna tıklandıktan sonra çalışır. Detected Faces klasöründe yüzler toplandıktan sonra klasör path olarak tanımlandı. Taniyici adında ki değişkene kullanılacak LBPH yöntemini OpenCV kütüphanesini kullanılarak import edildi. Yüzlerin eğitilmesi içinde yine haarcascade yöntemi kullanıldı. Path içindeki her bir fotoğraf liste şeklinde alındı. For döngüsüne sokularak PIL kütüphanesi kullanılarak fotoğraf array olacak şekilde img_numpy değişkenine atandı. Klasörün içindeki her bir fotoğraf User.(Id).fotoğraf_no olacak şekilde kaydedilmişti. Id bilgisini alıp etiketleme yapılması için yine OS kütüphanesi kullanıldı. Yüz verileri her bir yüz için for döngüsüne sokularak eğitildi. Yüz verileri ve Id ler fonksiyondan geri döndürüldü. Alınan bu veriler ile train yani eğitime işlemi başlatıldı ve yüzler eğitilip trainer.yml dosyası olarak egitilenveri klasörünün içine kaydedildi. Kullanıcıya kaç kişinin yüzünün eğitildiği bilgisi gösterildi.



Şekil 3. 29. Yüz Eğitme Ekranı

3.2.Yüz Tanıma İşlemi

Masaüstü programı ile hazırlanan eğitilmiş yüz verileri trainer.yml dosyasına kaydedildi. Veritabanında sahingoz.db olarak kaydedildi. Sistemi kurarken ki mantık bu iki dosyanın Raspberry Pi içine gönderilerek yüz tanıma kodunda kullanılması üzerineydi. Raspberry Pi üzerinde sadece yüz tanıma kodu çalıştırılarak CPU dan tasarruf edildi. Bundan sonra anlatılacak olan tüm işlemler Raspberry Pi üzerinde gerçekleşecektir.

```
1 #-*- coding: utf-8 -*-
2 import cv2
3 import os
4 import sqlite3
5
6 vt=sqlite3.connect('sahingoz.db')
7 cur=vt.cursor()
8
9
10 os.chdir("/home/pi/opencv/data/haarcascades")
11 taniyici = cv2.face.LBPHFaceRecognizer_create()
12 taniyici.read('/home/pi/Desktop/YuzTanima/trainer/trainer.yml')
13 cascadeYolu= "/home/pi/Desktop/YuzTanima/lbpcascade_frontalface.xml"
14 yuzCascade = cv2.CascadeClassifier(cascadeYolu);
15
16 font = cv2.FONT_HERSHEY_SIMPLEX
17
18
19 # Gerçek zamanlı video yakalamayı başlat
20 cam = cv2.VideoCapture(0)
21 cam.set(3, 800) # video genişliği ayarı
22 cam.set(4, 450) # video yüksekliği ayarı
23
24 # Yüz olarak tanınacak minimum pencere boyutunu tanımlama
25 minW = 0.1*cam.get(3)
26 minH = 0.1*cam.get(4)
27
```

Şekil 3. 30. Yüz Tanıma Veritabanı ve Kamera Ayarı

Gerekli kütüphaneleri import ettikten sonra veritabanına bağlanıldı.

Yüz tanıma işlemi için gerekli olan LBPH algoritmasını kullanarak taniyici adında bir değişken kullanıldı. Read fonksiyonu ile masaüstü program ile eğitilmiş olan yüz verilerinin olduğu trainer.yml dosyası okundu. Burada yüz tespitinden farklı olarak lbpcascade kullanıldı. Haarcascade ın doğruluk oranı yüksek olmasına rağmen hızı yavaş kalıyor. Kameradan sürekli frame alınacağı için daha hızlı çalışan bir cascade

olan lbp(local binary pattern) cascade kullanıldı. Birden fazla yüzde düşük bir doğruluk oranı verse de tek yüzde tanımlama yapılacağı için lbp hızlı olması açısından tercih edildi.

Gerçek zamanlı video yakalama işlemi için cv2 modülünün VideoCapture metodu kullanıldı. Yakalanacak frameelerin genişlik ve yükseklik ayarı set metoduyla düzenlendi.

```
30 while True:
31     ret, img = cam.read()
32     gri = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
33
34     faces = yuzCascade.detectMultiScale(
35         gri,
36         scaleFactor = 1.05,
37         minNeighbors = 5,
38         minSize = (int(minW), int(minH)),
39     )
40
41     for(x,y,w,h) in faces:
42         cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
43         id, confidence = taniyici.predict(gri[y:y+h,x:x+w])
44
45         cur.execute("select adi,soyadi,durum from suclular where Id=(?);" ,(id,))
46         result=cur.fetchall()
47         adi=result[0][0]
48         soyadi=result[0][1]
49         durumu=result[0][2]
50
```

Şekil 3. 31. Yüz Tanıma İşlemi ve Veritabanından Bilgi Çekme İşlemi

Yukarıda görüldüğü üzere bir döngü içinde kameradan alınan her frame gri tona çevirilip yüz tespit işlemi yapıldı. Buradaki amaç kameradan görünen kişinin yüz verileri ile önceden eğitilmiş olan yüz verilerini karşılaştırma yapmak için kameradan görünen kişinin yüz verisini almaktır. Alınan yüz verileri predict metodu ile confidence(güven) değerine atanıyor. Eğitilmiş yüz verileri ile karşılaştırıp bir güven değeri veriyor yani benzerlik oranı. Aynı zamanda eğer suçlu ile eşleşme sağlarsa veritabanından suçlunun adını, soyadını ve durumunu çekiyor.

```

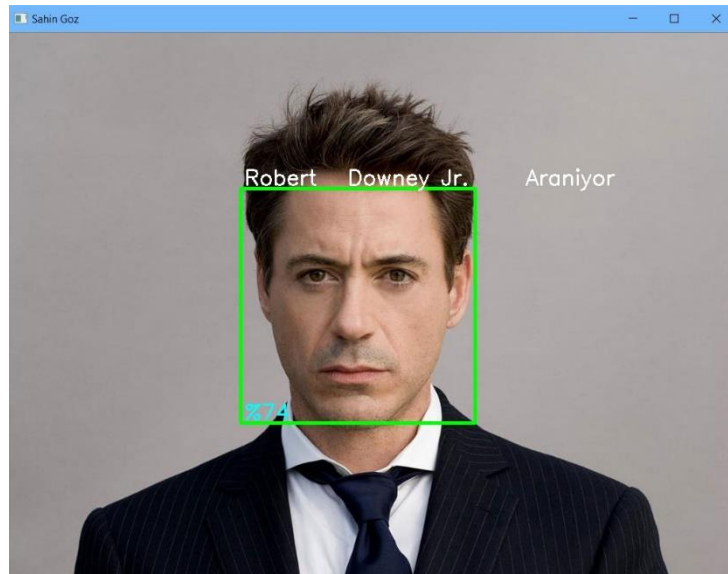
3     if (confidence > 65):
4         id = adi
5         soyad=soyadi
6         durum=durumu
7         confidence = " {0}%".format(round(confidence))
8     else:
9         id = "unknown"
10        soyad="unknown"
11        durum="unknown"
12        confidence = " {0}%".format(round(confidence))
13
14    cv2.putText(img, str(id), (x+5,y-5),font, 1, (255,255,255), 2)
15    cv2.putText(img,str(soyad),(x+150,y-5), font, 1, (255,255,255), 2)
16    cv2.putText(img, str(durum), (x+350,y-5), font, 1, (255,255,255), 2)
17    cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)
18
19    cv2.imshow('Sahin Goz',img)
20
21    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
22    if k == 27:
23        break
24
25    # Do a bit of cleanup
26    print("\n [Bilgi] Programdan çıkılıyor..")
27    cam.release()
28    cv2.destroyAllWindows()

```

Şekil 3. 32. Yüzlerin Karşılaştırma Sonucu Bilgilerin Ekranda Gösterilmesi

Eğer kameradan canlı olarak gelen görüntüde kişinin yüzü veritabanında kayıtlı olan herhangi bir suçluyla confidence değeri 65 ten fazla olacak şekilde uyuşma sağlarsa suçlunun bilgileri değişkenlere atanıyor. Uyuşma sağlanmazsa unknown(bilinmeyen) yazılıyor. Bunları yaparken kişinin yüzü bir kare içine alınıyor. Bilgiler karenin belli bölgelerine konumlandırılmış şekilde kullanıcıya gösteriliyor. Bu da putText metodu ile yapılıyor. Gösterilecek bilgileri, koordinatları, yazı fontu, çerçeve kalınlığı ve rengi bu metot ile ayarlanıyor. ImShow metodu ile kullanıcıya ekran açılıyor ve kullanıcı yüz tanıma işlemine gözlüğü takarak başlıyor.

Kullanıcının göreceği ekran ise şöyle olacaktır.



Şekil 3. 33. Sonuç

3.3.Gözlüğün Fiziksel Tasarımı

Vr olarak alınan gözlük kendi lensleri çıkartılarak ve içindeki gereksiz parçalar sökülerek yeniden modifiye edildi.



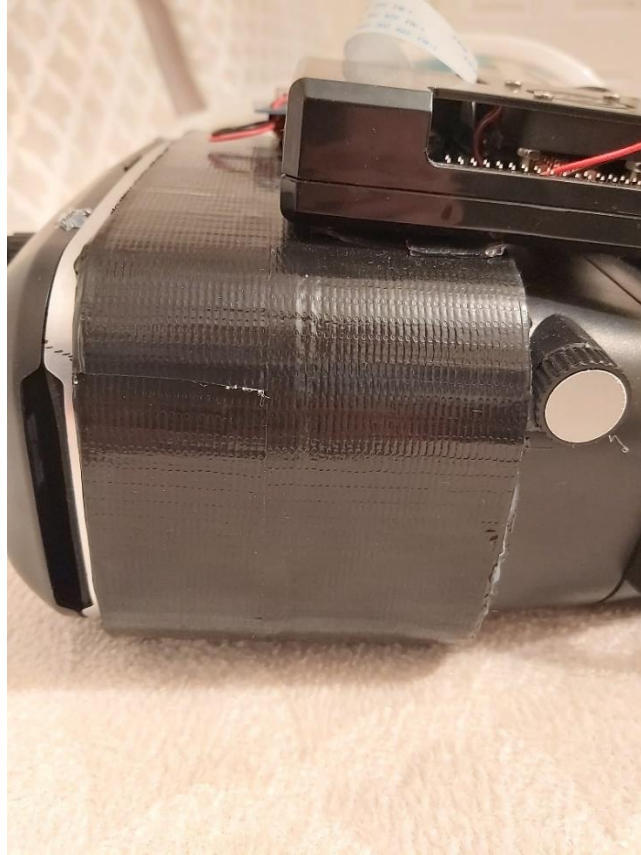
Şekil 3. 34. Fresnel Lens

Vr gözlüğün kendi içindeki lensler söküldü ve Fresnel Lens gözlüğün iç kısmına yerleştirildi. Bu lensin yerleştirilme amacı kullanıcının yakın mesafeden ekrana odaklanabilmesini sağlamaktır. Fresnel Lens bir büyüteç gibi görev görür ve ekranı olduğundan daha geniş bir şekilde gösterir.



Şekil 3. 35. Ekran Montajı

Normalde telefon için yapılmış olan ön panele sıcak silikon yardımıyla alınan 5 inçlik ekran sabitlendi. Gözlüğün yan tarafından kabloların geçeceği kadar kesikler açıldı.



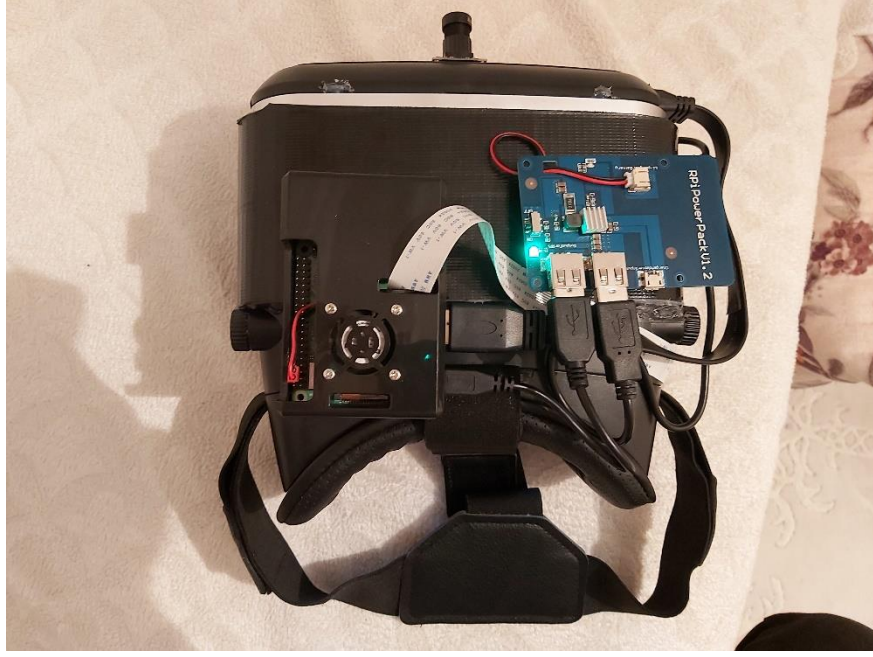
Şekil 3. 36. Ara Uzatma Parçası Montajı

Yukarıda şekil 3.36 da görüldüğü gibi tasarımını yapıp 3D yazıcı yardımıyla çıkartılan parçanın montajı görüntüleniyor. Bu parçanın çıkartılma amacı gözlüğün içine monte edilen Fresnel Lens in belli bir odak mesafesine sahip olmasıdır. İnsan gözünün rahatça görebilmesi için hem göz ile fresnel lens arasında hemde fresnel lens ile ekran arasında belli bir uzaklık olmalıdır. Yapılan hesaplamalara göre optimum uzaklık belirlenmiş ve bu uzaklığı sağlayacak olan parça eklenmiştir. Parçanın çizimi Şekil 2.5 te malzemeler tanıtılırken gösterilmiştir.



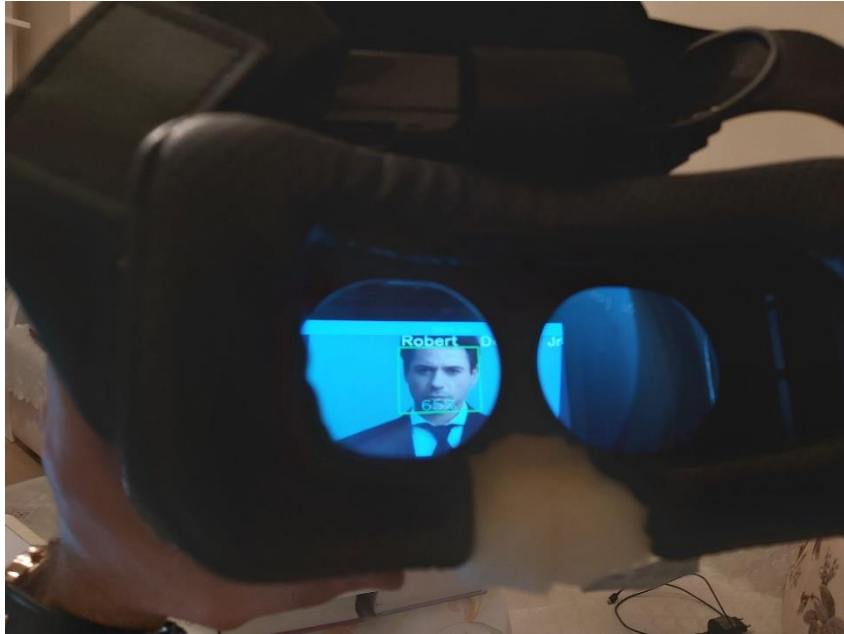
Şekil 3. 37. Kamera Montajı

Görüntüyü göz mesafesinden almak için gözlüğün tam ortasına kamera montaj edildi.



Şekil 3. 38. Raspberry Pi ve Güç Modülü Montajı

Gözlüğün ağırlık merkezine göre çok önde olmayacak şekilde solda siyah kutusunda Raspberry Pi ve sağda Raspberry Pi yi ve ekranı besleyecek olan 3800 mAH lik güç modülü monte edildi. Raspberry Pi ve ekran arasında görüntü aktarımı için HDMI kablo bağlantısı bağlandı. Güç modülünde Raspberry Pi için ayrılmış güç girişine Raspberry Pi bağlandı. Ekran için ayrılmış girişten ekran beslendi. Gözlük çalıştırıldı.



Şekil 3. 39. Kullanıcının Gözlükten Gördüğü Sonuç

BÖLÜM 4

SONUÇ VE DEĞERLENDİRME

Yüz tanıma gibi işlemlerin mini bilgisayar olan Raspberry Pi da nasıl çalıştığını incelemiş olduk. Raspberry Pi gibi içinde üst düzey GPU (Grafik İşlemci Birimi) barındırmayan mini bilgisayarlarda yüz tanıma işlemlerinin zorluğu hakkında bilgi edinilmiş olundu. Özellikle önemli geçiş noktalarında bu tür teknolojilerin güvenliği üst düzeye çıkaracağı anlaşılmış olundu. Teknolojinin şu anda computer vision (bilgisayar görüşü) alanında birçok ilerleme kaydettiği görülüyor. Geleceğin teknolojisinin derin öğrenme , bilgisayar görüşü ve yapay zeka gibi teknolojilerin bir harmanı olacağı görüşü bizce çok güçlü hale gelmiş bulunmaktadır.

“Sistemi hem fiziksel hem de yazılımsal olarak yeniden kurabilseydim daha dikkat edeceğim “ hususlar işe şöyledir. Daha güçlü bir mini bilgisayar özellikle GPU konusunda kendini kanıtlamış derin öğrenme için geliştirilmiş NVIDIA Jetson Nano gibi bir mini bilgisayar , daha hafif bir gözlük tasarımı çünkü araya eklenen eklenti ile birlikte gözlük normalden daha ağır bir hale geldi. Lens daha iyi ayarlanarak aradaki mesafe azaltılabilir ve doğru orantılı olarak ağırlık azaltılabilirdi. Ve de derin öğrenme algoritmaları kullanılarak alınacak verimin üst düzeye çıkacağını düşünüyoruz. Kullanılan LBPH algoritması düşük ışık ortamlarında ve benzer sebeplerden dolayı düşük bir doğruluk değeri sunuyor. Daha kesin sonuçlar için derin öğrenme algoritmaları kullanmak doğruluğu %99 a kadar arttıracaktır. Tabiki bunun için daha güçlü bir bilgisayar lazım olacaktır. Sonuç olarak hem fiziksel hem de yazılımsal bir sistem hazırlayarak birçok deneyim elde ettiğimi düşünüyorum.

KAYNAKLAR

1. Yüz Bulma ve Tanıma Sistemleri Kullanarak Kimlik Tespitinin Yapılması. https://www.researchgate.net/publication/309704392_YUZ_BULMA_VE_TANIMA_SISTEMLERI_KULLANARAK_KIMLIK_TESPITININ_YAPILMASI
2. Gerçek Zamanlı Video Görüntülerinden Yüz Bulma Ve Tanıma Sistemi. <https://docplayer.biz.tr/52268525-Gercek-zamanli-video-goruntulerinden-yuz-bulma-ve-tanima-sistemi.html>
3. Gerçek Zamanlı Sayısal Görüntü İşleme Ve Örüntü Tanıma Tekniklerinin Araştırılması Ve Uygulanması . <https://docplayer.biz.tr/35473892-Ankara-universitesi-fen-bilimleri-enstitusu-yuksek-lisans-tezi.html>
4. Face Recognition using OpenCV And Python: A Beginner's Guider. <https://www.superdatascience.com/blogs/opencv-face-recognition>

ÖZGEÇMİŞ

Alper ASLAN 1995 yılında ANKARA’da doğdu; ilk ve orta öğrenimini ANKARA’da tamamladı. 2013 yılında Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü’nde öğrenime başladı.

ADRES BİLGİLERİ

Adres :

Tel :

E-posta : alper0695@gmail.com