YÜZ, PARMAK İZİ, RFID VE KAREKODLU ENTEGRE GEÇİŞ KONTROL SİSTEMİ TASARIM VE UYGULAMASI

2019
YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

ALPER AKKAYA

YÜZ, PARMAK İZİ, RFID VE KAREKODLU ENTEGRE GEÇİŞ KONTROL SİSTEMİ TASARIM VE UYGULAMASI

ALPER AKKAYA

Karabük Üniversitesi

Fen Bilimleri Enstitüsü

Elektrik – Elektronik Mühendisliği Anabilim Dalında

Yüksek Lisans Tezi

Olarak Hazırlanmıştır

KARABÜK

Haziran 2019

Alper AKKAYA tarafından hazırlanan "YÜZ, PARMAK İZİ, RFID VE KAREKODLU ENTEGRE GEÇİŞ KONTROL SİSTEMİ TASARIM VE UYGULAMASI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Ahmet Hayrettin YÜZER .....
Tez Danışmanı, Elektrik – Elektronik Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Elektrik – Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 21/06/2019

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Dr. Öğr. Üyesi Ahmet Hayrettin YÜZER (KBÜ)	.....
Üye : Doç. Dr. Kemal POLAT (BAİBÜ)	.....
Üye : Dr. Öğr. Üyesi A. Reşit KAVSAOĞLU(KBÜ)	.....

...../...../2019

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Filiz ERSÖZ .....
Fen Bilimleri Enstitüsü Müdür V.

"Bu tezdeki 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 AKKAYA

ÖZET

Yüksek Lisans Tezi

YÜZ, PARMAK İZİ, RFID VE KAREKODLU ENTEGRE GEÇİŞ KONTROL SİSTEMİ TASARIM VE UYGULAMASI

Alper AKKAYA

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik – Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Ahmet Hayrettin YÜZER
Haziran 2019, 83 sayfa

Bu çalışmada dört farklı tanımlama teknolojisi kullanılarak kullanıcı geçiş kontrolü sistemi tasarlanmış ve geliştirilmiştir. Proje temel olarak Raspberry Pi (mini bilgisayar) üzerine inşa edilmiştir. Projede yüz algılama ve tanıma, parmak izi tanıma, RFID kart tanıma ve karekod çözme işlemleri Raspberry Pi üzerinde Python dili; sunucu taraflı doğrulama ve geçiş kontrol işlemleri Asp.NET yazılım teknolojisi ve Sql Server veri tabanı kullanılarak gerçekleştirilmiştir. Proje başta şirketler olmak üzere geçiş kontrol sistemi kullanılan tüm noktalarda ziyaretçi ve personel giriş çıkışlarının kontrol edilebilmesi için kabul görmüş teknolojilerle entegre çözüm sunmaktadır. Projede kullanılan parmak izi cihazının saklayabileceği maximum 200 adet sınırlaması da ortadan kaldırılmıştır. Sistemin aynı konfigürasyonlarla farklı noktalarda da kullanılabilmesi ve kolay kurulum yapılabilmesini sağlamak veya farklı nedenlerle cihazda oluşabilecek veri kayıplarından etkilenmemek adına yapılan tüm kayıtlar farklı bir sunucu üzerindeki veri tabanına aktarılmaktadır. Giriş veya

çıkışı istenmeyen kullanıcılara anında müdahale edilebilmesini sağlamak amacıyla tüm sistem internet ortamında çalışmaktadır. Ayrıca yapılacak geliştirmeler ile kullanıcıların içeride veya dışarıda olma durumları izlenebilir. Piyasada bulunan cihazların sağlamış olduğu günlük, haftalık veya aylık puantaj, eksik çalışma veya fazla mesai hesaplamalarına kadar tüm raporların alınması sağlanabilir.

Anahtar Sözcükler : Raspberry Pi, parmak izi tanıma, yüz tanıma, RFID, karekod, nesnelerin interneti.

Bilim Kodu **:** 905

ABSTRACT

M. Sc. Thesis

DESIGN AND IMPLEMENTATION OF FACE, FINGERPRINT, INTEGRATED ACCESS CONTROL SYSTEM WITH RFID AND QRCODE

Alper AKKAYA

Karabük University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electornics Engineering

Thesis Advisor:
Assoc. Prof. Dr. Ahmet Hayrettin YÜZER
June 2019, 83 pages

In this study, user access control system is designed and developed using four different identification technologies. The project is basically built on Raspberry Pi (mini computer). The project includes face detection and recognition, fingerprint recognition, RFID card recognition and qrcode decoding operations with the extra hardware supported by Raspberry pi; server-side authentication and migration control operations have been developed with Asp.NET software technology and Sql Server database. The project provides an integrated solution with accepted technologies for control of entry and exit of visitors and staff at all points to be controlled. The maximum number of fingerprints (200 finger print template) that can be stored by the fingerprint device used in the project has been eliminated. All the records are transferred to the database on a different server to ensure that the system can be easily used at different points with the same configurations or to be affected by data loss which may occur in the device for different reasons. The whole system

works online on the internet in order to provide immediate response to undesired users. In addition, improvements can be made to monitor whether users are in or out. It can be ensured that all reports can be obtained from daily, weekly or monthly payroll, missing work or overtime calculations provided by commercially available devices.

Key Word **:** Raspberry Pi, fingerprint recognition, face recognition, RFID, qrcode, internet of things.

Science Code : 905

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üy. Ahmet Hayrettin YÜZER'e sonsuz teşekkürlerimi sunarım.

Sevgili aileme hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

Projenin belirlenmesinde ve geliştirme aşamasında desteğini hiçbir zaman eksik etmeyen başta tüm Teracity Ar-Ge merkezi ve Özdilek Holding ailesine teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

RFID : Radio Frequency Identification (Radyo Frekanslı Tanımlama)
QR : Quick Response (Hızlı Tepki)
DNA : Deoxyribonucleic Acid (Deoksiribonükleik Asit)
HOG : Histogram Of Gradient (Gradyanların Histogramı)
API : Application Program Interface (Uygulama Programlama Arayüzü)
FAR : False Accept Rate (Hatalı Kabul Oranı)
FRR : False Reject Rate (Hatalı Reddetme Oranı)
PCA : Pricipal Component Analysis (Temel Bileşn Analizi)
DARPA : Defense Advanced Research Projects Agency (Savunma İleri Araştırma Projeleri Ajansı)
SVM : Support Vector Machine (Destek Vektör Makineleri)
RELU : Rectified Linear Units (Doğrultulmuş Doğrusal Birimler)
NFC : Near Field Communication (Yakın Alan İletişimi)
IOT : Internet Of Things (Nesnelerin İnterneti)
SOC : System On Chip (Çip Üzerinde Sistem)
SQL : Structered Query Language (Yapılandırılmış Sorgu Dili)
MVC : Model View Controller (Model Görüntüleme Kontrol)
CRUD : Create Read Update Delete (Ekleme Okuma Güncelleme Silme)
HTTP : Hypertext Transfer Protocol (Hiper Metin Transfer Protokolü)
CNN : Convolutional Neural Network (Konvolüstonel Sinir Ağı)
PDKS : Personel Devam Kontrol Sistemleri
KVKK : Kişisel Verilerin Korunması Kanunu

BÖLÜM 1

GİRİŞ

Sürekli artan nüfus ve iş gücü ile birlikte güvenlik problemleri tüm sektörlerdeki işletmeler için bir endişe oluşturmaktadır. Bu nedenle çalışma yaşamının en önemli parçalarından çalışanların, ekipmanların ve bilgilerin güvende kalmasını sağlamak her geçen gün daha da zorlaşmaktadır. Bununla birlikte, çalışanların ve ziyaretçilerin iş yerine girmeleri için basitlik ve kolaylık sağlamak, aynı zamanda onları sorumlu tutmak önemli bir husustur. Oluşabilecek güvenlik tehditlerinin önüne geçmek için yararlanılabilecek en önemli araçlardan bir tanesi geçiş kontrol sistemleri olarak düşünülebilir.

Geçiş kontrol sistemi kişilerin, araçların veya nesnelerin hareketlerini kontrol edebilecek mekanik ve elektronik sistemler bütünüdür. Geçiş temel olarak 4 ana bileşenden oluşur. Bunlar; giriş sınırlayıcı, kontrol girdisi, kontrolör ve çıkış olarak söylenebilir. Giriş sınırlayıcı; genelde fiziksel geçişleri engelleyecek mekanik sınırlayıcılardır. Turnike, bariyer, kapı kilidi bu sınırlayıcıların başlıca örnekleri olarak gösterilebilir. Çıkış ise, giriş sınırlayıcıyı kontrol edecek işlem veya işlemler bütünüdür. Elektrikli motorlar, röleler bu bileşenin temel örnekleridir. Geçiş kontrol sistemlerinde en önemli noktalar ise kontrol girdisi ve kontrolördür. Kontrol girdisi istenilen güvenlik derecesine uygun olarak şeçilebilecek ve giriş yapanın kimliğini tespit etmeye yarayacak bileşendir. Başlıca kontrol girdileri; şifre veya pin numaraları, proximity kartları, parmak izi, retina, damar veya yüz resimleri gibi biyometrik tanımlar olarak sıralanabilir. Kontrolör ise kullanılan girdilere uygun olarak ilgili tarama donanımlarını kontrol eden, girdiyi yorumlayan ve tepki üreten karar mekanizmasıdır.

Geçiş kontrol sistemleri izinsiz geçişlerin ve buna bağlı olarak oluşacak güvenlik problemlerinin önlenmesi için son derece etkili araçlardır. Güvenliğin yanı sıra

birçok faydası daha bulunmaktadır. Kimin veya neyin hangi noktadan ne zaman geçtiğinin bilinmesi geçiş kontrol sistemini kullanan kuruma birçok farklı raporlama ve ölçüm imkanı da sağlamaktadır. Bu raporlamalar sayesinde kaç kişinin içeride olduğu, mesai saatlerine uyulup uyulmadığı, ne kadar fazla mesai yaptığı ne kadar süre ile sistemde bulunduğu, o noktadan geçiş yapan kişi sayısı gibi bilgiler edinilebilir. Yine bu bilgiler ışığında personel performans değerlendirmesi, eksik veya fazla çalışmaları da hesaba katan otomatik maaş hesaplaması veya bir noktaya kurulacak olan işletmeye uğrayabilecek kişi sayısına kadar birçok istatistiksel veri elde edilebilir.

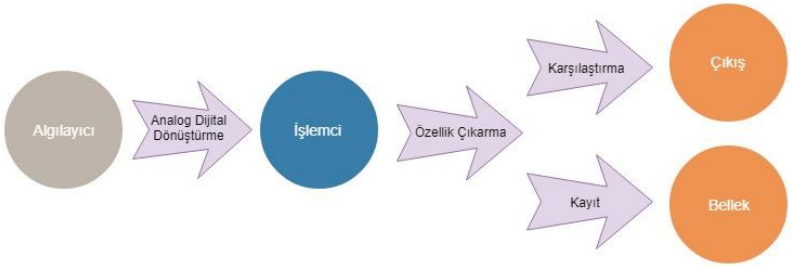
BÖLÜM 2

BİYOMETRİK TANIMA

Bio (yaşam) ve metron (metrik) kelimelerinin birleşiminden meydana gelen biyometrik tanımı fiziksel ve davranışsal ölçüleri istatistiksel olarak incelemek anlamına gelmektedir [1]. Biyometrik kimlik doğrulamanın temel dayanağı, her insanın kendi fiziksel veya davranışsal özellikleri ile doğru bir şekilde tanımlanabilmesidir [2].

İnsanlar biyometrik tanıma işlemini doğduğundan itibaren gerçekleştirmeye başlar. Yeni doğan bebekler annelerin sesini, kokusunu ve çevresini duyu organları ile algılayarak kaydetmeye başlarlar. Daha sonra kaydettikleri verileri önceki veriler ile karşılaştırarak tanıma işlemlerini gerçekleştirirler. Biyometrik tanıma cihazları da bir bebeğin tanıma işlevini yerine getirme yöntemleri ile benzer çalışmaktadır.

Birçok teknolojinin gelişmesine sebep olan güvenlik problemi biyometrik tanıma cihazlarının da geliştirmesinde en önemli etken olmuştur. Gelişen teknoloji ile birlikte biyometrik özellikler analog dünyadan dijital ortama aktarılabilmektedir. Bu aktarımla birlikte biyometrik verilerin çeşitli algoritmalarla işlenmesi ve değerlendirilmesi de söz konusudur. Şekil.2.1'de biyometrik tanımanın gerçekleştirilmesine ilişkin akış şeması gösterilmiştir.



Şekil 2.1. Biyometrik tanıma sistemi akış şeması.

Şekil 2.1'de gösterilen biyometrik tanıma akış şeması tüm biyometrik tanıma işlemleri için ortak olsa da algılayıcılardan karşılaştırma algoritmasına kadar tüm adımlar biyometrik tanıma türleri için farklılık göstermektedir. Biyometrik tanıma sistemlerinin tüm değişkenleri sistemin doğruluğu ile direk olarak ilişkilidir ve kullanılacak sistemin net belirlenebilmesi için sistem doğruluğunu ölçmek gerekmektedir. Bu noktada hem sistemin kendi doğruluğu hemde diğer sistemlerle ilişkisini tanımlamak için iki adet ölçüm parametresi bulunmaktadır. Bunlar;

- False Accept Rate (FAR) : Türkçe karşılığı hatalı kabul etme oranıdır. Bu oran sistemde bulunmayan kişilerin sistemde varolan bir kişi ile eşleştirilmesi oranını vermektedir.
- False Reject Rate (FRR) : Tükçe karşılığı hatalı reddetme oranıdır. Bu oran sistemde bulunan fakat tanınamayan kişilerin oranıdır.

Yukarıda bahsedilen doğruluk ölçüm oranlarından "FAR" en çok dikkat edilmesi gereken hususdur çünkü kayıtlı olmayan personelin tanınması tam olarak güvenlik problemi oluşturacak noktadır. Çizelge 2.1 "International Biometric Group" tarafından paylaşılan ve en çok kullanılan ticari biyometrik sistemlerden bazıları arasındaki örnek "FAR" ve "FRR" değerlerini göstermektedir.

Çizelge 2.1. Bazı ticari biyometrik sistemlerin "FAR" ve "FRR" değerleri [3].

	Parmak Damarı	**Avuç İçi Damarı**	**İris**	**Parmak İzi**
Üretici Firma	Hitachi - Omron	Fujitsu	Iris Guard	BioScrypt
Cihaz	UBReader	PalmSecure	H100	LifeView
Hatalı Kayıt	0.55 %	1.63 %	7.01 %	0 %
FRR	1.26 %	4.23 %	1.76 %	1.67 %
FAR	0.01 %	0.01%	0.01 %	1.46 %

Biyometrik sistemlerin yanlış tanıma, hatalı kayıt gibi dezavantajları bulunmasına rağmen kartlı veya şifreli kontrol sistemlerine karşı unutma, çalınma, tahmin edilebilirlik gibi problemlerinin bulunmaması ve biyometrik özelliklerini başka birisine devredemiyor olması gibi avantajları mevcuttur.

2.1. BAŞLICA BİYOMETRİK TANIMA YÖNTEMLERİ

2.1.1. Parmak İzi Tanıma

Parmak izinin tarihçesine bakıldığında ilk parmak izi örnekleri antik Babil Kentinde, kil tabletlerde ticari sözleşmeler için kullanılmıştır. 14.yyda Persler devlet sözleşmelerinde parmak izi kullanmıştır. Devlet tarafından görevlendirilmiş bir doktor; iki parmağın izlerinin birbiri ile tamamen farklı olduğunu farketmesiyle kişiye özel bir işaret olarak daha yaygın kullanılmaya başlanmıştır.

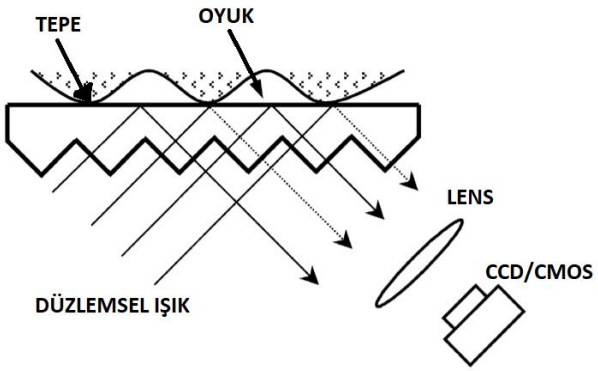
1686 yılında Italya/Bologna Üniversitesi Anatomi Profesörü Macello Malpighi parmak izinin oyuklardan, sarmallardan ve iç içe geçmiş döngülerden oluştuğunu tezinde göstermiştir fakat parmak izinin eşsiz oluşuyla ilgili bir bilgiye yer vermemiştir [4].

Sir William Hershel parmak izinin kişisel bir veri olarak kullanılabileceğini keşfeden ilk Avrupalıdır. 1858 Hint İşgalinin ardından, Herschel Hint Sivil Hizmetinin bir üyesi oldu ve Jungipoor'a gönderildi. 1858 yılının Temmuz ayında, yol yapım malzemelerinin temini için yerli Konai ile bir sözleşme hazırladı. Konai'nin daha sonraki bir tarihte imzasını reddetmesini önlemek için, Herschel belgeye bir parmak izi koydu. Herschel, parmak izleriyle deney yapmaya devam etti ve kısa zamanda sadece parmakların kullanılmasının gerekli olduğunu fark etti. Arkadaşlarından ve ailesinden baskılar topladı ve bir kişilerin parmak izlerinin zamanla değişmediği sonucuna vardı. Herschel ayrıca suçluların yakalanması ve takibi için hapishanelerde parmak izi kullanılması önerdi ve sağladı. Şekil 2.2'de Herschel tarafından alınan el ve parmak izine ait örnek bir resim gösterilmiştir [5].

Şekil 2.2. Herschel tarafından alınan el ve parmak izine ait örnek [5].

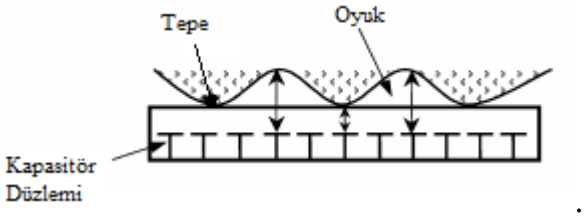
Parmak izi algılama işlemlerinin tamamen dijitalleşmesinden önce parmak izi kayıtları genel olarak suçla mücadele için mürekkep tekniği kullanılarak yapılmıştır. Bu teknikte parmak mürekkebe batırılır, kağıda bastırılır ve yuvarlanır. Mürekkep ile kağıt üzerinde oluşan parmak izi tarayıcılar ile bilgisayara alınır ve bu şekilde dijital parmak izi bilgileri oluşturulur. Bu şekilde elde edilen parmak izi taramasına offline algılama denir.

Offline algılama sistemleri pahalı, zaman alıcı ve dağınık bir işlemdir. Anlık tarama cihazlarının geliştirilmesi offline algılama sistemlerinin kullanımını ortadan kaldırmıştır. Dijital algılama sistemlerinde parmak izi resminin algılanabilmesi için çeşitli algılayıcı türleri bulunmaktadır. Bunlar optik algılayıcılar ve katı yüzey algılayıcılardır. Optik algılayıcılarda temel iki yüzey arasındaki yansıma ve geçiş kanunlarına dayanmaktadır. Gönderilen ışık parmak izi ile sensör yüzeyi arasında kırılır ve(ya) yansır. Fiziksel kurallar çerçevesinde parmak izinin doğasında bulunan tepeler ve oyuklardan belirli bir gecikme ve kırılma ile yansıyan ışık optik bir CCD/CMOS algılayıcı veya elektro-optik sensörler ile algılanarak siyah beyaz resme dönüştürülürler. Şekil 2.3'te geleneksel optik algılayıcılara örnek bir şema verilmiştir.



Şekil 2.3. Geleneksel optik algılayıcı şematik gösterimi [12].

Optik algılayıcılara ek iki tip katı yüzey algılayıcı da parmak izi algılamasında kullanılır. Bu tip algılayıcılardan en bilineni ve en çok kullanılanı kapasitif algılayıcılardır. Kapasitif algılayıcılar da optik algılayıcılara benzer şekilde çalışırlar. Bu algılama türünde parmak izi tepe ve oyuklarının mesafesi yansıma ve kırılma prensiplerine göre değil elektrik alan şiddetine göre belirlenerek parmak izi görüntüsü oluşturulur. Şekil 2.4'te kapasitif algılama prensibine dayanan bir algılayıcının şematik gösterimi sunulmuştur



Şekil 2.4. Kapasitif parmak algılayıcı şematik gösterimi [12].

Parmak izi tanıma ile ilgili daha ayrıntılı bilgiler bölüm 3'te verilmiştir.

2.1.2. Yüz Tanıma

Yüz tanıma yıllarca yalnızca bilim kurgu filmlerinde gerçeklenebilecek bir olgu gibi görünse de son birkaç yıldır bu olgu bir gerçekliğe dönüşmekle kalmadı, gerçek hayatta da yaygın bir şekilde kullanılmaya başlandı. Yüz tanıma teknolojisi son birkaç yıldır kullanım olarak yaygınlaşmaya başlamış olsa da geçmişi 1960'lı yıllara dayanmaktadır. Çoğu uzman bu teknolojinin babasının Woodrow Wilson Bledsoe

olduğunu söyler. Bu söylemin nedeni Bledsoe tarafından geliştirilen ve elektromanyetik bir kalem aracılığı ile bir resmin özelliklerinin yatay ve dikey düzlemde koordinatlarının belirlenmesini sağlayan "RAND" tabletidir. Bu tablet sayesinde göz, saç çizgisi dudaklar gibi yüz özelliklerinin koordinatları kaydedilebilir ve bir veri tabanında tekrar doğrulama yapmak üzere kullanılabilir. Bu o zamanın teknolojisindeki işlem gücü ile sınırlı olduğundan çok efektif olmamasına rağmen yüz tanıma ile ilgili atılan ilk adımdı. Şekil 2.5'te RAND tablete ait bir resim gösterilmektedir.

Şekil 2.5. Bledsoe tarafından geliştirilen RAND tablet [16].

1970'lerde Goldstein, Harmon ve Lesk tanıma işlemini otomatikleştirmek için saç rengi ve dudak kalınlığı gibi 21 spesifik subjektif belirteç kullandılar. El ile hesaplanması gereken ölçümler ve konumlar, programın çok fazla çalışma süresi gerektirmesine neden olsa da RAND Tablet teknolojisi ile karşılaştırıldığında daha doğru sonuçlar verdiği gözlemlenmiştir [6].

1988'de Kirby ve Sirovich, yüz tanıma problemine, standart bir lineer cebir tekniği olan temel bileşen analizini (PCA – Principle Component Analysis) uyguladı. Bu

analiz uygun şekilde hizalanmış ve normalize edilmiş bir yüz görüntüsünde yüz değerden daha az nokta ile tanımanın gerçekleştirilebileceğini kanıtladı [11].

1991'de, Turk ve Pentland, görüntülerdeki yüzleri nasıl tespit edeceklerini keşfederek Özyüz (Eigenface) yaklaşımını geliştirdiler. Yaklaşım çevresel faktörlerle kısıtlanmasına rağmen yine de otomatik yüz tanıma teknolojilerinin gelişiminde önemli bir ilgi yarattı.

Eigenface algoritması ile daha çok ilgi odağı olan yüz tanıma teknolojisinde pazar teşviki ve daha başarılı yüz tanıma sistemlerinin geliştirilmesini sağlamak amacıyla FERET programı devreye alındı. 1990'lı yıllarda başlayan ve en son 2003 yılında güncellenen, Savunma İleri Araştırma Projeleri Ajansı (DARPA) ve Ulusal Standartlar ve Teknoloji Enstitüsü tarafından desteklenen bu program sayesinde 856 kişiye ait 2413 adet yüz resmi veri tabanına alınarak paylaşıma açıldı.

Yüz tanıma teknolojisinde yaşanan gelişmeler üzerine teknolojinin geniş anlamda ilk kullanımı 2002 yılında güvenlik görevlileri tarafından Super Bowl projesi ile kullanıldı. Kullanım sonucu yapılan olumsuz geri dönüşler nedeniyle yüz tanıma sisteminin bu gelişmeler ile kullanıma tam hazır olmadığı görüldü.

Super Bowl projesinden sonra geniş çaplı bir yüz tanıma teknolojisi 2010 yıllarının başında Facebook sosyal medya platformu üzerinden sunuldu. Facebook yüklenen fotoğraflarda belirlenebilen kullanıcıların fotoğraf üzerinden etiketlenebilmesini sağlayarak başarılı bir yüz tanıma sisteminin kullanımdaki ilk örneği oldu. Bu gelişmelere paralel 2011 yılında Panama havaalanında suçla mücadele için kullanılmaya başlayan yüz tanıma sistemi Interpol tarafından aranan birçok suçlunun bulunmasına yardımda bulunmuştur.

Bilgisayar teknolojilerinde gelişen yapay sinir ağı ve derin öğrenme algoritmaları ile yüz tanıma sistemlerinin de başarısı artarak devam etmekte ve kullanımı da aynı oranda artmaktadır. Günümüzde yüz tanıma sistemleri havaalanlarından bankalara, stadyumlardan hastanelere, polis merkezlerinden kişisel mobil cihaz uygulamalarına kadar geniş bir alanda kullanılmaktadır.

Yüz tanıma sistemi genel olarak kameralardan alınan bir kare içerisindeki yüzlerin bulunarak ayrılması, bulunan bu yüzlerden özellik noktalarının çıkarılarak bir veri tabanına kaydedilmesi ve karşılaştırma için gelen farklı bir yüz fotoğrafı ile belirli algoritmalara uygun olarak karşılaştırılmasına dayanmaktadır. Tanıma algoritmaları ile ilgili ayrıntılı bilgi bölüm 4'te anlatılmaktadır.

2.1.3. İris Tanıma

İris göz merceği ile saydam tabaka arasında bulunan, göze rengini veren ve göze gelen ışığın yoğunluğuna uygun olarak büzülüp genişleyen ve gözün renkli kısmını oluturan kas tabakasıdır. Şekil 2.6'da gözün bölümlerine ilişkin resim gösterilmektedir.

Şekil 2.6. Gözün bölümleri.

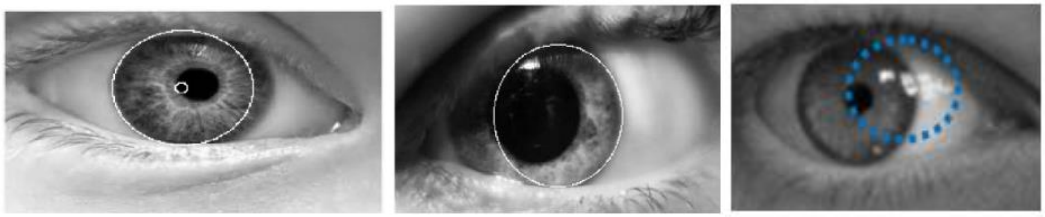
Göz rengi genetik olarak bağımlı olmasına rağmen iris içerisinde oluşan desenler genetik olarak bağımsız oluşmaktadır. İris, doku zarının sıkı bir şekilde biçimlendirilmesi ve katlanması yoluyla doğum öncesi büyüme sırasında gelişir. Doğumdan önce gerçekleşen dejenerasyonla göz bebeği açılır ve iriste rastgele desenler meydana gelir. Bu nedenle tek yumurta ikizlerinin dahi irisleri farklıdır ve bu durum iris tanımanın eşsiz olmasını mümkün kılar.

İris tanıma için ilk patent John Daughman tarafından 1994 yılında alınmış olsa da tarihi daha eskiye dayanmaktadır. İris tanıma ile ilgili 1936'da göz doktoru Frank Burch, bireyi tanımak için bir yöntem olarak iris desenlerini kullanma konseptini

önerdi. 1953 yılında ise göz fizyolojisti F.H. Adler kitabında iris tanımadan "Aslında, irisin işaretleri o kadar belirgindi ki, fotoğraflar parmak izi yerine bir tanımlama aracı olarak kullanılabilir." diye bahsetmiştir. 1980'lerde iki Amerikalı göz doktoru, L. Flom ve A. Safir Adler irisin insan tanımlayıcısı olarak hizmet edebileceği tezini savundular, ancak bunu gerçekleştirmek için gerçek bir algoritmaları ya da uygulamaları yoktu. Bu nedenle Dr. Flom irisin otomatik olarak tanımlanabilmesini sağlayacak algoritmanın geliştirilmesi için Dr. John Daugman'a başvurdu.

1993 yılında Flom, Safir ve Daugman'ın çabalarıyla nükleer savunma ajansı tarafından başlatılan araştırma ve geliştirme ile Daugman 1994 yılında iris tanıma algoritmaları için patent aldı ve 1995 yılında başarılı bir prototip üretimi gerçekleştirilerek ilk ticari iris tanıma cihazı piyasaya sunuldu [7].

İris tanıma temel olarak 4 aşamadan oluşmaktadır. Bu aşamalardan ilki göz resminin bir kamera aracığılığı ile alınmasıdır. Alınan görüntü gauss filtresi uygulanarak yumuşatılır. Bu sayede göz içerisinde oluşan ışık yansımaları ve gürültüler ortadan kaldırılır. Sonraki aşama iris tanımanın sonucunu büyük derecede etkileyecek en önemli nokta olan irisin resimden ayrılması işlemidir. İnsan gözünün yapısından ve yakalanan görüntüdeki bozukluklardan (kirpikler, göz kapakları irisin göz içerisindeki konumu vs.) dolayı elde edilecek iris kodu hatalı olacak ve bu durum sonucun hatalı olmasında en etkili rolü oynayacaktır. Şekil 2.7'de hatalı tanımaya neden olabilecek resimler gösterilmektedir.

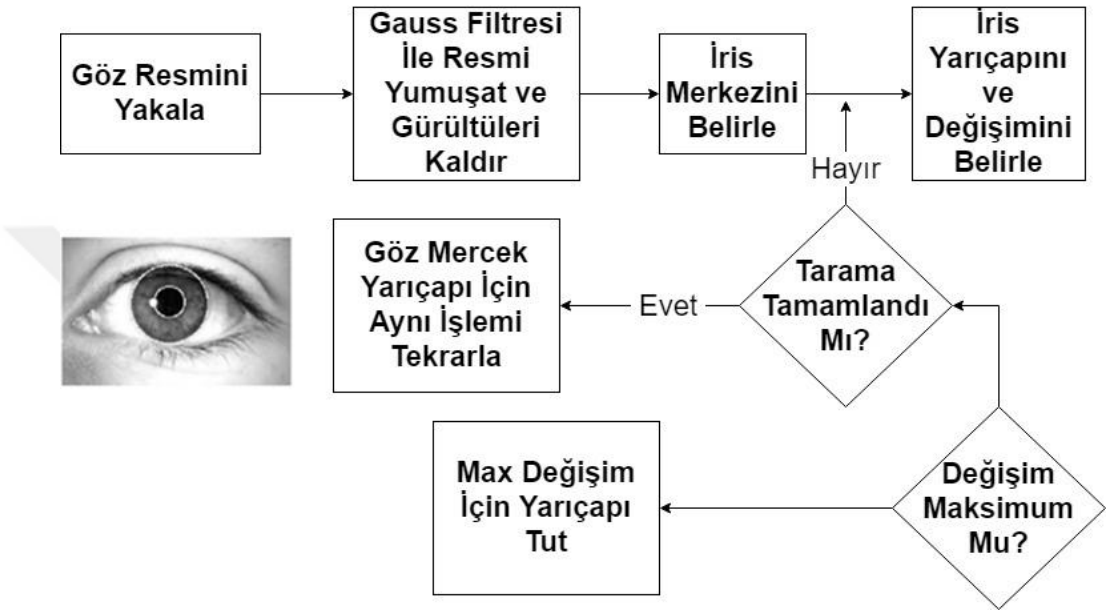


a b c

Şekil 2.7. a) Göz kapağı hatası, b) hatalı göz konumu, c) yansıma problemi.

İrisin göz merceği ve saydam bölgeden ayrılması sonrasında elde edilen iris radyal düzlemden yatay düzleme dönüştürülerek iris kodu elde edilir ve bu kod belirlenen

bir veri tabanında eşleştirilerek saklanır. Daha sonrasında tanınmak istenen yeni iris kodu ve kaydedilmiş referans kodlar çeşitli algoritmalar ile (örn. hamming mesafesi algoritması ile) karşılaştırılarak tanıma sonucu üretilir. İrisin göz resminden ayrılması için yarıçapların belirlenmesine ilişkin algoritma şeması ve algoritma sonucu Şekil 2.8'de gösterilmiştir.



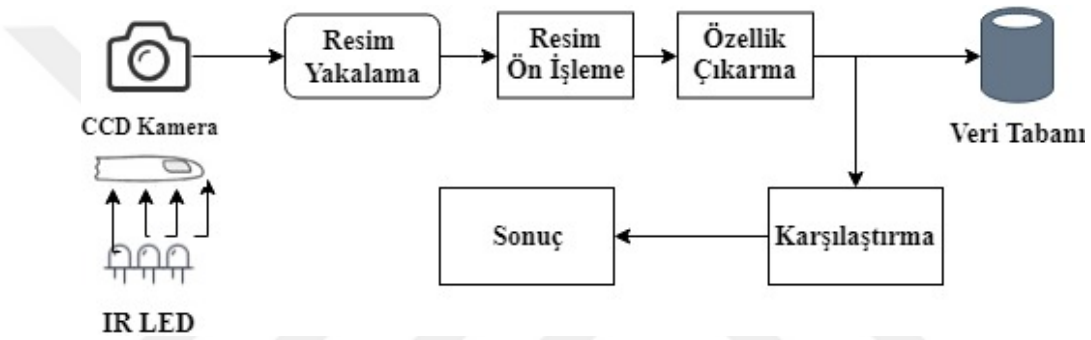
Şekil 2.8. İris ve göz merceği yarıçaplarının belirlenmesi için uygulanacak algoritma.

2.1.4. Damar Tanıma

Damar tanıma diğer tanımalar ile karşılaştırıldığında bulunuşu ve gelişimi açısından çok daha yeni bir konumdadır. Bu tanıma türü ilk olarak 1992 yılında Dr. K. Shimizu tarafından BT görüntüleme tekniklerini kaleme aldığı bir makaleye dayanmaktadır [8]. 1996 yılında, yazar Yamamoto K, K. Shimizu ile birlikte, önceki araştırmayı tartıştıkları bir başka makale sundu [9]. Biyometrik tanıma için damar desenlerinin kullanımıyla ilgili ilk araştırma makalesi 2000'li yıllarda yayınlandı [10].

Damar tanıma işlemi herhangi bir cerrahi işleme gereksinim duymadan deri altında bulunan damarların tanınması işlemidir. Bu tanımayı diğer tanımalardan (örn: parmak izi tanıma) ayıran en önemli özelliği canlılık özelliği gerektirmesinden dolayı

hiçbir şekilde taklit edilemez olmasıdır. Yine diğer tanıma özelliklerinden farklı olarak damar tanıma işlemi farklı bir bileşene ihtiyaç duymaktadır. Bu bileşen kızıl ötesi ışık kaynadığır. Kızıl ötesi ışık normal ışıktan farklı dalga boyunda yayım yaptığından derinin altına nüfuz edebilmektedir. Deri altına nüfuz eden kızıl ötesi ışık kan damarlarındaki hemoglobinin tarafından (deriye oranla) daha fazla absorbe edilmektedir. Deriden karşıya geçen ışık miktarı damarlara denk gelen ışıktan daha yoğun olacağından ccd kameralardan elde edilen görüntüde damarlar daha koyu görünmekte ve damar yolları belirlenebilmektedir. Şekil 2.9'da damar tanıma için gerekli algoritmanın akış şeması gösterilmiştir.



Şekil 2.9. Damar tanıma işlem ve uygulama akış şeması.

2.1.5. DNA Tanıma

DNA (deoksiribo nükleik asit) dünya üzerindeki bütün canlı organizamalarında ortak olan, canlıya ait tüm genetik bilgileri tutan moleküldür ve tüm genetik bilgiler canlıya ait özellikleri belirttiğinden eşsizdir. DNA, Francis Crick tarafından keşfedilmiş ve bu keşif ona 1962 yılında Nobel ödülü kazandırmıştır.

DNA tanımadan önce kimliklerin belirlenmesi çokta kolay değildi. DNA testlerinden önce yapılan testler (kan tiplemesi, serolojik testler HLA testleri) faydalı olmasına rağmen kimlik tanıma ve kişiler arası ilişkileri tanımlamak için yeterli değildi. 1970-1980 yılları arasında DNA testlerinin başlatılmasıyla, DNA tanıma ve ilişkilendirme geçerli hale geldi.

1920’li yılların başında bilim adamları kanların içerisinde A, B, AB ve 0 gruplarını oluşturan antijenler olduğunu belirlediler. Bu gruplamalar doktorlara donör ile alıcı arasındaki kan transferlerini güvenli bir şekilde yapma imkanı sağladı. Bilim adamları bu kan gruplarının ebeveynlerden miras olarak alındığını ve çocuk ile ebeveyn arasında bu sayede bağlantı kurulabileceğini anladı. Fakat kan gruplarından elde edilen bilgiler sınırlı olduğundan kesin bir biyolojik ilişki kurmak mümkün değildi.

1930’lu yıllarda bilim adamları AB0 kan grupları haricinde kan yüzeyinde başka proteinlerin bulunduğunu farketti. AB0 kan sistemi gibi Rh, Kell ve Duffy kan grubu sistemleri AB0 kan grubu sistemlerine ek katkı sağlasa da bu tanımlamalarda biyolojik tanıma için yeterli değildi.

1970'lerin ortalarında, bilim insanları doku tiplerine odaklandılar ve kırmızı hücrelerin dışında vücutta bulunan lökosit antijenini (HLA) keşfettiler. Kanda bulunan beyaz hücrelerinde yüksek yoğunlukta ve çok farklı tiplerde HLA bulunduğu ve bu HLA tiplerinin ilişkili olmayan insanlar arasında farklılık gösterdiği anlaşıldı. HLA tiplerinin ilişkili olmayan insanlar arasında gösterdiği bu yüksek çeşitlilik ve farklılık biyolojik ilişkilerin cevaplanmasında önemli bir nokta oldu. HLA testinin AB0 ile birleştiğinde %80, serolojik testte dahil olduğunda %90 oranında tanıma sağlayabildiği görülmüştür. HLA testini zorlaştıran en önemli faktör çok yüksek miktarda kan toplama ihtiyacının bulunması oldu.

1980'lerin başında, DNA kullanarak ilk genetik test haline gelen ve RFLP (Restriction Fragment Length Polymorphism) analizi olarak bilinen bir teknik geliştirildi. DNA bilgileri AB0, HLA ve serolojik bilgilerde olduğu gibi ebeveynlerden kalıtsal olarak aktarılır. Bilim adamları DNA içerisinde HLA ve kan gruplarından daha fazla değişkenlik gösteren alanlar olduğunu keşfettiler. DNA kırmızı kan hücreleri hariç tüm hücrelerde bulunduğundan biyolojik ilişkilerin çözümü için ideal bir çözümdür. RLFP tekniğinde DNA da tekrar eden bölgelerin kesilmesi için enzimler kullanılır. Bu enzimler DNA üzerinde bulunan belirli noktaları tanıyarak kesim yapar ve DNA’yı ikiye ayırır. Çocuk ve ebeveynlerin test edildiği babalık testinde DNA’nın yarısı anne diğer yarısı ise baba ile eşleşmelidir.

Nadiren de olsa bazı mutasyonlar veya dışlamalar sonucu çocukta bulunan bir konum ebeveynlerle eşleşmeyebilir. Fakat RLFP DNA testinin sonuçları %99.99 civarındadır. Fakat gerekli DNA miktarı ve uzun hesaplama süreleri nedeniyle bu test düzenli olarak uygulanmamaktadır.

1990'lı yıllarda sadece yanaktan alınan birkaç doku hücresinden elde edilen PRC (Polymerase Chain Reaction) çok hızlı bir şekilde sonuç üretebildiğinden RLFP tekniğinin yerini almıştır. PCR, DNA'da STR'ler (Short Tandem Repeats) adı verilen oldukça değişken bölgeleri hedefler. Annenin, çocuğun ve iddia edilen babanın test edildiği babalık testinde, çocuğun DNA'sı, mutasyon olmadıkça her iki biyolojik ebeveyne uymalıdır. Bu testte de RLFP tekniğinde olduğu gibi %99.99 oranında doğru sonuçlar üretilmektedir.

2.2. BİYOMETRİK TANIMA KULLANIM ALANLARI VE FAYDALARI

Biyometrik tanıma kişiye özgü tanımanın gerekli olduğu tüm işlemler için kanunların izin verdiği ölçüde kullanılabilmektedir. Tanımanın sahada sıklıkla kullanıldığı alanlar kısaca şu şekilde özetlenebilir.

- Yasaların Uygulanması: Tarihsel olarak polis teşkilatları biyometrik çözümlerin en çok kullanıldığı alan olmuştur. Yüzyıldan uzun süredir parmak izi tanıma, son birkaç on yıldır DNA tanıma ve son birkaç yıldır ise yüz tanıma sistemleri suçluların teşhisinde ve yakalanmasında önemli rol oynamıştır.
- Taşımacılık: Hava, deniz, kara, demiryolu vb taşımacılıklarda yolculuğun güvenli bir şekilde tamamlamasını sağlamak taşımacılık işletmelerinin ortak paydasıdır. Göçmenlik, korsanlık, terörizm gibi yasa dışı girişimlerin önlenmesi, tanımlama için gereken ve uzun süren incelemeler (pasaport, kimlik vb. kontrolleri) gibi sorunların önüne geçmek için biyometrik doğrulama sistemleri kullanılmaktadır. Özellikle son yıllarda yüz tanıma teknolojisi ile gelişen kimlik ve davranış tespitleri doğrultusunda bu tür sorunlar daha da azaltılmıştır.

- Personel Devam Takip Sistemleri: İş gücünün devamlı ve düzenli şekilde yönetimi tüm işletmelerin yaşam döngüsü için bir zorunluluktur. İşler ve çalışanlar için bu düzenin sağlanabilmesi için çok çeşitli yöntemler bulunmaktadır. Gelişen dünyaya ayak uydurmak, zaman ve maliyet dengesini sağlayabilmek için biyometrik tanıma sistemleri gerekli hale gelmiştir. İşyerlerinin iş kollarına ve işin önem derecesine uygun olarak çeşitli biyometrik tanıma sistemleri mevcuttur. Yaygın olarak kullanılan biyometrik tanıma parmak izi tanımadır. Yüz, iris, damar vb. tanıma türleri de ayrıca kullanım alanları bulmaktadır. Biyometrik tanıma cihazları ile çalışanın o an orada bulunma zorunluluğu yerine getirilmiş olacağı gibi, alınan verilerin gün saat bilgileri ile aylık çalışma raporunun oluşturulması, maaş hesaplanması, izinsiz geçişlerin engellenmesi gibi birçok fayda da bu cihazlar sayesinde sağlanmaktadır.
- Bankacılık: Bankalar kişilerin kurumların maddi varlıklarını değerlendirme ve koruma amacıyla hizmet veren kuruluşlardır. Bu nedenle finansal varlıklar dijital olarak daha fazla temel aldıkça, bankalar sahtekarlıkla mücadelede, işlem güvenliğini arttırmada ve müşteri rahatlığını arttırma çabasında müşteri ve çalışan kimlik yönetimini geliştirmek için biyometrik teknolojiyi kullanıyorlar. Kullanımı gün geçtikçe artan mobil bankacılık işlemlerine erişim için yüz ve parmak izi tanıma teknolojileri kişisel olarak kullanılmaktadır. Ayrıca banka ATMlerinde yapılacak fiziksel işlemlerde (para giriş-çıkışı, bakım işlemleri vb.) güvenliğin kesin olarak sağlanabilmesi için birden fazla kişi ve birden fazla tanıma teknolojisi ile erişim günümüz bankacılık işlemlerinde kullanılmaktadır. Bu teknolojilerin kullanılması ile planlanmamış veya yetkisi olmayan kullanıcıların ATM içerisine erişimi engellenmiş olur.
- Gözetim: Büyük insan gruplarının bulunduğu stadyumlar, konserler, kutlamalar vb. gibi noktalarda anormal durumların ve davranışların belirlenmesi için de biyometrik tanıma sistemleri kullanılmaktadır. Grup içerisinden belirgin bir kişinin tanınması ve ölçülmesi için yüz tanıma teknolojisi kullanılmaktadır.

BÖLÜM 3

PARMAK İZİ TANIMA

Parmak izi tanıma, iki parmak izinin karşılaştırmasına dayanarak, bir bireyin kimliğini belirlemek veya onaylamak için kullanılan otomatik yöntem anlamına gelir. Parmak izi tanıma en iyi bilinen biyometrilerden biridir ve bugüne kadar bilgisayarlı sistemlerde kimlik doğrulama için en çok kullanılan biyometrik çözümdür. Parmak izi tanımanın bu kadar popüler olmasının nedenleri; edinme kolaylığı, diğer biyometrilere göre kabul edilebilirliğinin tartışmasız oluşu ve her bireyde çok sayıda kaynağının (on parmak) oluşudur.

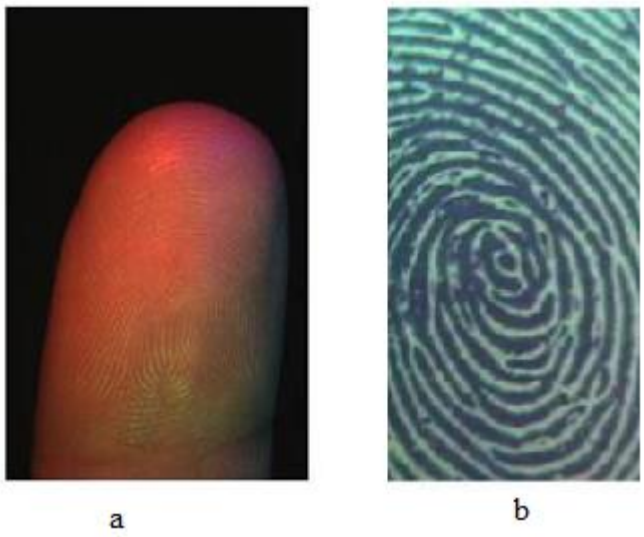
Parmak izi parmak iç yüzey derisinde bulunan tepe (çıkıntı) yüzeylerinin görüntüsüdür. Bu çıkıntılar sonlanma, çatallanma ve kısa sonlanma olarak üç temel özellik (minutiae) olarak karşımıza çıkar. Sonlanma adından da anlaşılacağı üzere bir çıkıntının sonlandığı noktadır. Çatallanma bir çıkıntının iki ayrı çıkıntı olarak ayrıldığı noktadır. Kısa sonlanma ise diğer sonlanmalardan belirli bir şekilde kısa uzunlukla sonlanan noktalardır. Şekil 3.1'de parmak izi özellik noktalarına ilişkin görsel gösterilmektedir.

Şekil 3.1. a) Parmak izi sonlanma özellik noktası, b) parmak izi çatallanma özellik noktası.

Parmak izi tanıma sistemleri ise bir bireye ait özellik noktalarının diğer bireylere ait özellik noktaları ile çeşitli benzerlik ve eşleştirme algoritmları yardımıyla karşılaştırılmasını sağlayan sistemler bütününü ifade etmektedir.

3.1. PARMAK İZİ TANIMA ALGORİTMASI AKIŞ ŞEMASI

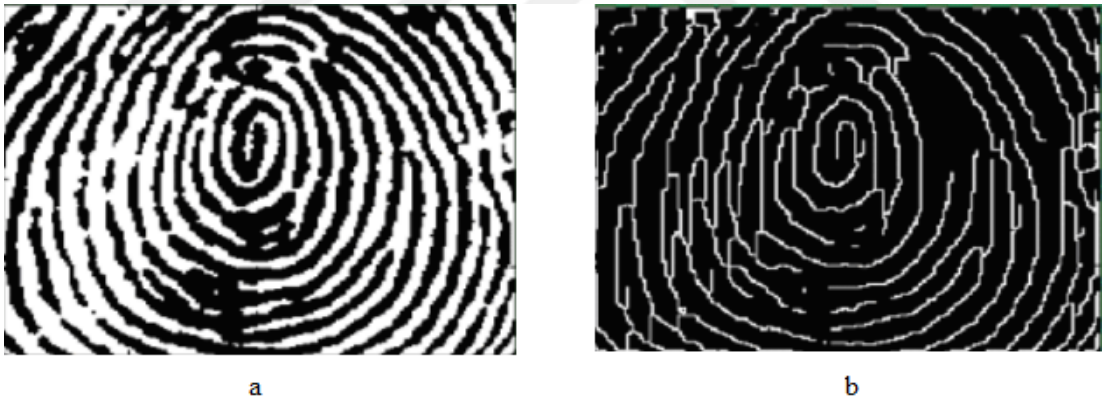
Parmak izi tanıma sistemleri parmak izi görüntüsünün kameralar aracılığı ile dijital ortama alınması ile başlar. Alınan bu görüntüler resmin daha sağlıklı olarak işlenebilmesi için 8 bitlik gri renk uzayına dönüştürülür.



Şekil 3.1. a) RGB renk uzayı parmak izi görüntüsü, b)8 bit gri renk uzayı parmak izi.

Elde edilen resim üzerine histogram eşitleme tekniği uygulanır ardından arkaplan ile parmak izi görüntüsü birbirinden ayrılarak (segmentation) parmak izi hatlarının net olarak ortaya çıkarılması sağlanır. Şekil 3.2'de histogram eşitleme işlemi gerçekleştirilmiş parmak izi görüntüsü gösterilmiştir. Arkaplan gürültüsünden de arındırılan parmak izi görüntüsü tekrardan 8 bitlik gri renk uzayından 2 bitlik siyah ve beyaz renk uzayına dönüştürülür ve özellik noktalarının net olarak belirlenebilmesi için inceltme (thinning, skeleton) işlemine tabi tutulur. İnceltme işlemi, parmak izi işaretlerinin bir piksel genişliğe düşürülerek özellik noktalarının kesin olarak belirlenebilmesi için en önemli aşamalardan birisidir. Şekil 3.3'de 2 bitlik renk uzayına dönüştürülmüş (binarized) ve inceltilmiş (thinned) parmak izi görüntüleri gösterilmiştir.

Şekil 3.2. a) 8 bit gri renk uzayında parmak izi görüntüsü, b) histogram eşitleme işleminden geçirilmiş parmak izi görüntüsü.

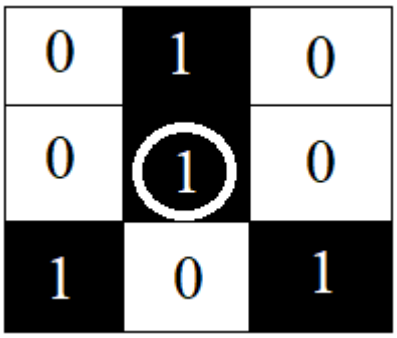


Şekil 3.3. a) Binarized parmak izi görüntüsü, b) thinned parmak izi görüntüsü.

Çoğu otomatik parmak izi eşleştirme sistemleri özellik noktalarının eşleştirilmesi esasına dayanır. Özellik noktaları parmak izlerinde meydana gelen sonlanma ve çatallanmaları ifade etmektedir.

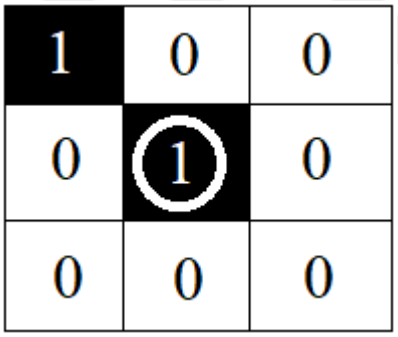
Parmak izi inceltmesinden sonra parmak izi özellik noktalarının belirlenmesi nispeten kolay bir işlemdir. Özellik noktalarının belirlenmesinde tüm resmi tarayan 3x3 boyutunda bir özellik çıkarım penceresi kullanılır. Şekil 3.4'te gösterildiği gibi

3x3 tarama penceresinde merkez pixel değeri 1 ve 1 değerinde 3 komşu pikseli bulunuyorsa bu durumda merkez noktası bir çatallanma özellik noktasıdır.



Şekil 3.4. Parmak izi çatallanma noktası örnek pencere görüntüsü.

Şekil 3.5'te gösterildiği gibi 3x3 tarama penceresinde merkez piksel değeri 1 ve 1 değerinde yalnızca 1 komşusu bulunuyorsa bu durumda merkez nokta bir sonlanma özellik noktasıdır.



Şekil 3.5. Parmak izi sonlanma noktası örnek pencere görüntüsü.

Parmak izi resimlerinden elde edilen sonlanma ve çatallanma noktaları daha sonra eşleştirme yapmak üzere belirlenen formatta veri tabanına kaydedilirler.

3.2. PARMAK İZİ EŞLEŞTİRME

Parmak izi eşleştirme işlemi özellik noktaları belirlenmiş ve veri tabanına kaydı gerçekleştirilmiş bir izin diğer parmak izi ile karşılaştırılması işlemidir. Eşleştirme işlemi için parmak izi noktalarının piksel cinsinden yerlerinin ve türlerinin

belirlenmesi yeterli değildir. Çünkü parmak izi görüntülerini her zaman aynı pozisyonda almak mümkün değildir bu nedenle bir referans özellik noktası gerekmektedir. Referans noktasının belirlenmesi işlemi için çıkarılan özellik noktalarının komşu özellik noktaları ile uzaklığı ve açısı hesaplanır. Bu sayede karşılaştırılacak parmak izinin pozisyonunun önemi olmayacaktır. Şekil 3.6 aynı kişiye ait ayrı zamanlarda alınmış iki parmak izinin karşılaştırılması için kullanılacak algoritma yapısını göstermektedir.

Şekil 3.6. Parmak izi görüntülerinin özellik noktalarının eşleştirilmesi [13].

Şekil 3.6'da mavi nokta ile işaretlenen özellik noktası belirlenen referans noktasını, yeşil çizgiler eşleşen komşu özellik noktası kenarlarını ve sarı çizgiler ise kenar eşleşmesini destekleyen kenarları göstermektedir.

Eşleşen özellik noktalarının sayısına ve destekleyen kenarların sayısına uygun olarak belirlenen bir eşik değerini geçen karşılaştırma işlemlerinde parmak izlerinin aynı kişiye ait olduğu söylenebilir. Araştırmalara göre eşleştirilen 12 adet özellik noktası parmak izlerinin aynı kişiye ait olduğunun söylenebileceği eşik olarak belirtilmektedir [14].

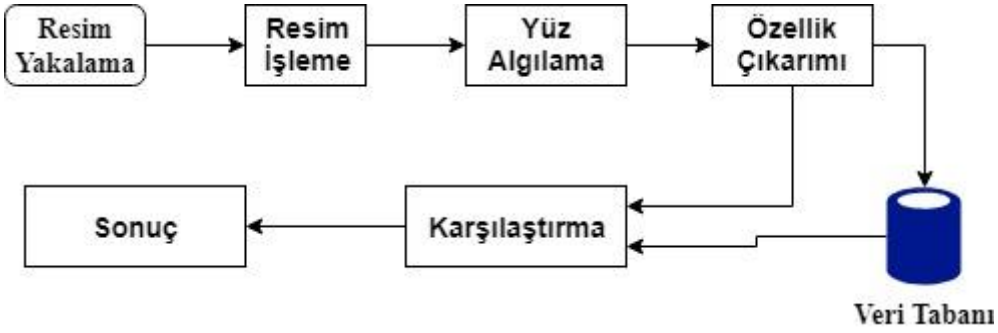
BÖLÜM 4

YÜZ ALGILAMA VE TANIMA

Yüz tanıma bir biyometrik tanıma türüdür. Teknolojinin en temel amaçlarından birisi makinaların da insanlar gibi öğrenen varlıklar haline gelmesidir. İnsanların belirli bir yaştan sonra otomatik olarak gerçekleştirmiş olduğu yüz tanıma işlemi son birkaç yıldır güvenlik endişelerinin artması ve teknolojinin de gelişmesi ile bilgisayarlar tarafından yüksek doğrulukla gerçekleştirilebilmektedir.

Yüz tanıma sistemleri yüz algılama (face detection) ve yüz tanıma (face recognition) algoritmalarından oluşmaktadır. Yüz tanıma ve yüz algılama birbiri ile karıştırılan konulardır. Yüz algılama adından da anlaşılacağı üzere bir karede bir yüz olup olmadığının, kare içerisindeki konumunun, pozisyonunun vb. gibi özelliklerin belirlenmesini ifade etmektedir. Yüz tanıma ise algılanan yüzü referans alarak kime ait olduğunun belirlenmesi olarak ifade edilebilir. Bu durumda yüz algılama yüz tanımanın ilk basamağıdır.

Yüz tanıma sistemleri de parmak izi tanımaya benzer olarak analog dünyadan kameralar ile alınan görüntülerin dijital ortama aktarılması ile başlar. Dijital ortama alınan resimler bir dizi resim düzenleme işlemlerinden geçirilir. Daha sonra düzeltilmiş resim üzerinde bulunan yüzler tespit edilerek resimden çıkarılır ve bu yüz resimleri veri tabanında bulunan diğer yüz resimleri ile karşılaştırılarak uygun sonuçlar üretilir.



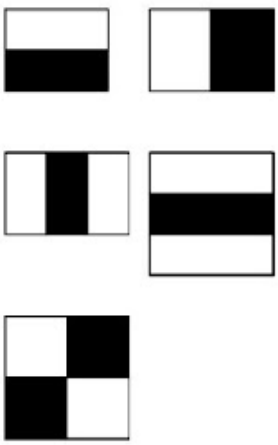
Şekil 4.1. Yüz tanıma sistemi algoritma şeması.

4.1. YÜZ ALGILAMA

4.1.1. Haar – Cascade Sınıflandırıcı

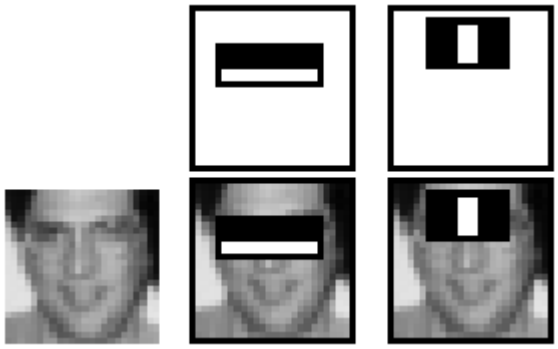
Haar tabanlı kaskad sınıflandırıcılar Paul Viola ve Michael Jones tarafından önerilen etkili bir nesne algılama yöntemidir [15]. Sınıflandırıcı pozitif ve negatif resimlerle eğitildikten sonra daha önce karşılaşmadığı resimlerden pozitif olarak eğitildiği nesneleri tespit etmek üzere kullanılır.

Haar sınıflandırıcı nesneleri tanımak için Şekil 4.2'de gösterilen özellik pencereleri ile pozitif ve negatif resimleri tarayarak aydınlık ve karanlık piksel değerleri toplanmından bir sonuç vektörü elde eder.



Şekil 4.2. Sınıflandırıcı özellik pencereleri [17].

Şekil 4.2'de gösterilen özellik pencereleri zayıf sınıflandırıcılar olarak adlandırılırlar. Zayıf sınıflandırıcılar tek başlarına iyi bir sınıflandırıcı olamazlar. Bu nedenle nesnelerin belirlenmesinde birden fazla zayıf sınıflandırıcı kullanılır ve zayıf sınıflandırıcılardan birkaçının yüksek sonuçlar ürettiği çerçevelerin içerisinde aranılan nesnenin bulunduğu sonucuna varılır. Yüzün doğası gereği pozlanması esnasında (ışığın geliş açısına da bağlı olarak) çeşitli karanlık ve aydınlık noktalar bulunmaktadır. Örneğin yüz üzerinde kaşlar karanlık nokta olarak görünürken kaşların başladığı alın noktası aydınlıktır veya pozlama sonrası burun kemiği aydınlık iken burun kenarları veya göz çukuru daha aşağıda bulunduğundan daha karanlık noktalar olarak gözükecektir. Şekil 4.3'te verilen örneklere ait iki adet zayıf sınıflandırıcı gösterilmiştir.



Şekil 4.3. Yüz tanıma için örnek olarak seçilen zayıf sınıflandırıcı örnekleri [15].

Resim boyutlarının çeşitlilik göstermesi nedeniyle tarama çerçeveleri ve zayıf sınıflandırıcılar küçükten başlayarak sürekli büyütülür ve genelde sol üst köşeden başlayarak orijinal resim taratılır. Çerçeve içerisinde yapılan tarama sonuçlarından eşik değerini aşan ve en yüksek değere sahip olan çerçeveler konumsal olarak işaretlenerek nesne tanıma işlemi gerçekleştirilir.

Şekil 4.3'te üst satırın sağında bulunan zayıf sınıflandırıcı göz çevresini belirlemeye yardımcı olduğundan yararlı sınıflandırıcıdır fakat aynı sınıflandırıcı ağız bölgesinin belirlenmesinde yararsızdır. Bu nedenle yararlı sınıflandırıcılar ve yararsız sınıflandırıcılar nesnenin daha yüksek doğrulukla tanımlanabilmesi için Adaboost algoritması ile ağırlıklandırılırlar.

Her bir çerçeve taraması için tüm olası sınıflandırıcıları kullanarak konvolüsyon işlemini gerçekleştirmek çok ağır bir işlem yükü gerektirir (24x24 boyutunda bir tarama penceresinde 160.000'den fazla özelliğe neden olur [18]). Bu konvolüsyon işlemlerini daha az işlem yükü ile gerçekleştirmek için integral resim algoritması kullanılır. İntegral resim algoritmasında taranacak olan resmin her bir piksel değeri sol üstten başlayarak ilgili piksel noktasını kapsayacak şekilde bir dikdörtgen çizildiğinde tüm piksel değerlerinin toplamının hesaplanmak istenen piksel değerine yazılması ile elde edilir. Örneğin belirli bir pencere içerisindeki değerlerin ortalamasını hesaplamak için klasik yöntem ve integral resim algoritmasını aşağıdaki şekilde karşılaştırabilir.

5	4	3	8	3
3	9	1	2	6
9	6	0	5	7
7	3	6	5	9
1	2	2	8	3

a

5	9	12	20	23
8	21	25	35	44
17	36	40	55	71
24	46	56	76	101
25	49	61	89	117

b

Şekil 4.4. a) Orjinal piksel değerleri, b) integral resim piksel değerleri.

5	4	3	8	3
3	9	1	2	6
9	6	0	5	7
7	3	6	5	9
1	2	2	8	3

Şekil 4.5. Klasik hesaplama ile ortalaması hesaplanacak alan çerçevesi.

Şekil 4.5'te işaretli alanın ortalama değerini hesaplamak için klasik hesaplamada 9 adet işlem kullanılmaktadır.

(9+1+2+6+0+5+3+6+5) / 9 = 4.11

Klasik hesaplama kullanılarak aynı işlem 100 kez tekrarlandığında işlem adedi 900 olacaktır. Aynı işlemleri integral resim yöntemi ile hesaplarsak.

5	9	12	20	23
8	21	25	35	44
17	36	40	55	71
24	46	56	76	101
25	49	61	89	117

Şekil 4.6. Integral resim hesaplama ile ortalaması hesaplanacak alan çerçevesi.

Şekil 4.5'te işaretli alanın ortalama değerini hesaplamak için klasik hesaplamada 4 adet işlem kullanılmaktadır.

((76 – 20) – (24 – 5)) / 9 =4.11

İntegral hesaplama kullanılarak aynı işlem 100 kez tekrarlandığında işlem adedi 400 olacaktır. İntegral resmin hesaplanması içinde gerekli olan 56 adım ile birlikte işlem yükü yarı yarıya azaltılacaktır.

4.1.2. HOG Sınıflandırıcı

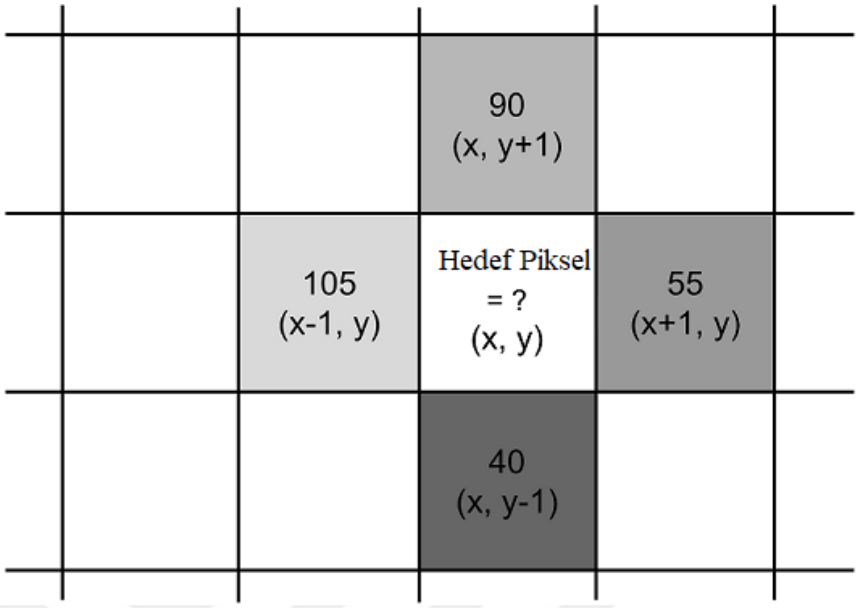
HOG (Histogram of Gradient); bilgisayarlı görü ve resim işlemede nesne algılama amacıyla kullanılan bir özellik tanımlayıcısıdır. HOG kişi detektörü 2005 yılında CVPR konferansında Dalal ve Triggs tarafından tanıtılmıştır [19].

HOG sınıflandırıcıda temel amaç resmin bir ucundan diğer ucuna resim içerisindeki piksel değerlerinin değişimini elde etmektir. HOG sınıflandırıcılar resmin bloklara bölünmesi ile başlar. Her blok, kendi içerisindeki tüm piksellerin komşu piksellerle olan ilişkisinin büyüklük ve yön cinsinden tanımlanması ile elde edilir.

Görüntü içerisindeki her bir pikselin x ve y yönündeki renk değişimi gradyan olarak tanımlanır. Resim gradyan hesabı büyüklük ve yön olmak üzere iki önemli özellik ile belirlenir.

Büyüklük : $g = \sqrt{{g_x}^2 + {g_y}^2}$ (4.1)

Açı : $\theta = \arctan(g_y \,/\, g_x)$ (4.2)



Şekil 4.7. Örnek imge piksel değerleri.

Şekil 4.7 bir resim üzerinden alınmış piksel kesitlerini ve hesaplanacak piksel değerlerinin komşu piksel değerlerini göstermektedir. Örnek resim üzerinde hedef pikselin değişimi ve gradyan hesabı Eşitlik 4.3 ve Eşitlik 4.4 ile verilmiştir.

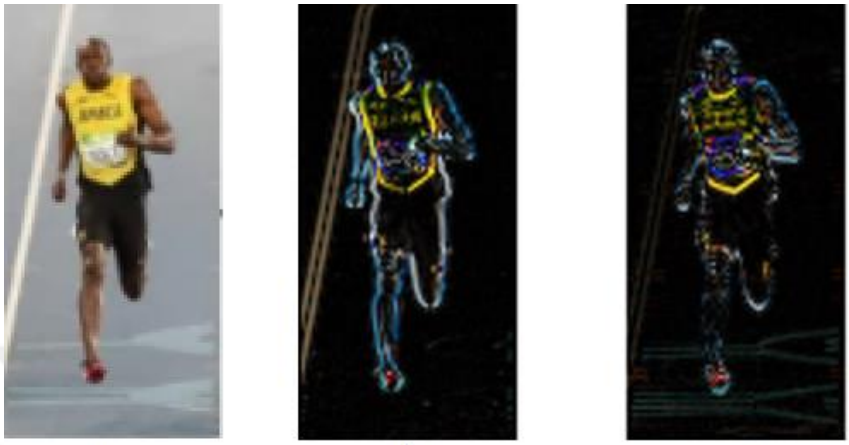
$$\nabla f = \begin{bmatrix} f(x+1,y) - f(x-1,y) - & f(x-1,y) \\ f(x,y+1) - f(x,y-1) - & f(x,y-1) \end{bmatrix} = \begin{bmatrix} 50-105 \\ 90-40 \end{bmatrix} = \begin{bmatrix} -50 \\ 50 \end{bmatrix} \quad (4.3)$$

$$g = \sqrt{50^2 + -50^2} = 70.71 \text{ ve } \theta = \arctan\left(-\frac{50}{50}\right) = -45^\circ \quad (4.4)$$

Fakat hesabın bu şekilde her piksel için iteratif yolla uygulanması çok yavaş çözümler elde edilmesine sebep olmaktadır. Bu nedenle gradyan hesabında gerekli hızı elde edebilmek için sobel çekirdek matrisleri ([-1,0,1] , [+1,0,-1]) ile konvülasyon işlemleri gerçekleştirilir. Sobel konvülasyonunun hesaplama işlemi Eşitlik 4.5 ve 4.6'da verilmiştir. Sobel konvülasyonu uygulanmış görsel ise Şekil 4.8'de gösterilmiştir.

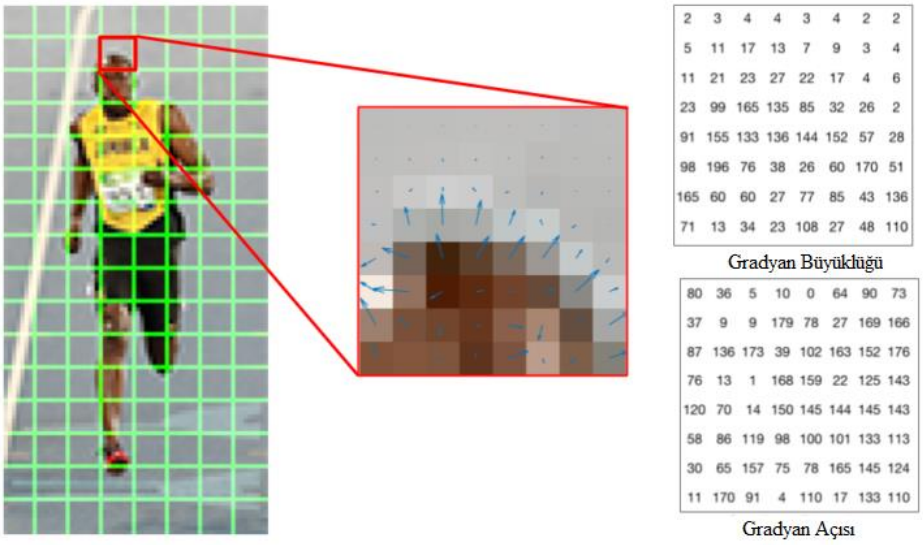
$$G_x = [-1,0,1] * [105,255,55] = -105 + 0 + 55 = -50 \quad (4.5)$$

$$G_y = [1,0,-1]^T * \begin{bmatrix} 90 \\ 255 \\ 40 \end{bmatrix} = 90 + 0 - 40 = 50 \tag{4.6}$$



Şekil 4.8. a) Orijinal resim, b) x yönünde sobel filtresi uygulanmış resim, c) y yönünde sobel filtresi uygulanmış resim [20].

Her piksel için gradyanlar hesaplandıktan sonra bulunmak istenen objeye ait resim 8x8 piksel boyutunda pencerelere bölünür. Her piksel büyüklük ve açı cinsinden 2 değer daha içereceğinden 128 adet özellik elde edilir. Resmin 8x8 pencerelere bölünmesi ve gradyan büyüklükleri Şekil 4.9'da gösterilmiştir.



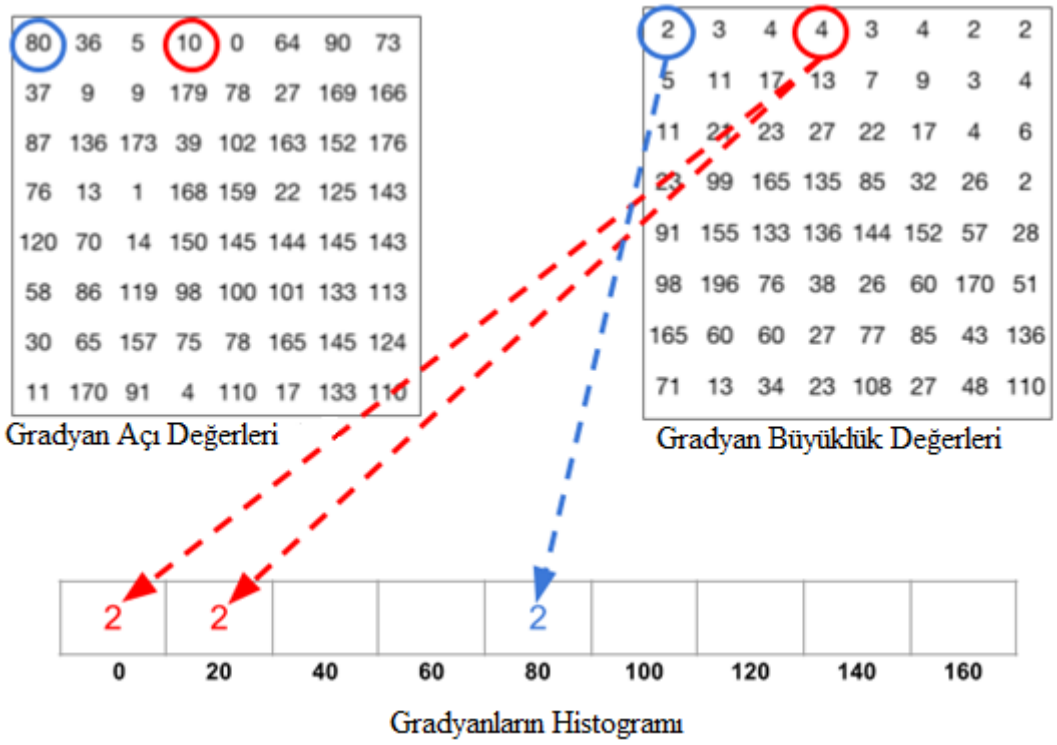
Şekil 4.9. 8x8 boyutunda pencerelere bölünerek gradyan büyüklüklerinin hesaplanması [20].

128 adet özellik 9 karakterlik bir sayı dizisi olarak saklanacak ve bu dizi 9 bölmeli histogram olarak gösterilebilecektir. Bu sayede tanınmak istenen objeye ait özellikler kompakt ve SVM (support vector machine) gibi öğrenme algoritmaları için de uygun bir yapıya kavuşmuş olacaktır. Histogramların 9 bölmeli histogramlara bölünmesi işlemi hesaplanan piksel gradyan açı değerlerinin 0 - 180 derece arasında 20'şer derecelik bölmelere, gradyan büyüklüklerine uygun olacak şekilde atanması işlemidir. Histogram gradyanlarının 9 bölmeli histogram dizisine dönüştürülmesine ilişkin görsel Şekil 4.10'da gösterilmiştir. Büyüklüklerin bölmelere dağıtımı için kullanılabilecek denklem;

$$\theta_1 < \theta < \theta_2 \;\; ve \;\; \theta_2 = \theta_1 + 20^\circ \tag{4.7}$$

$$g_{\theta_1} = \frac{\theta - \theta_1}{20} * g \tag{4.8}$$

$$g_{\theta_2} = \frac{\theta_2 - \theta}{20} * g \tag{4.9}$$



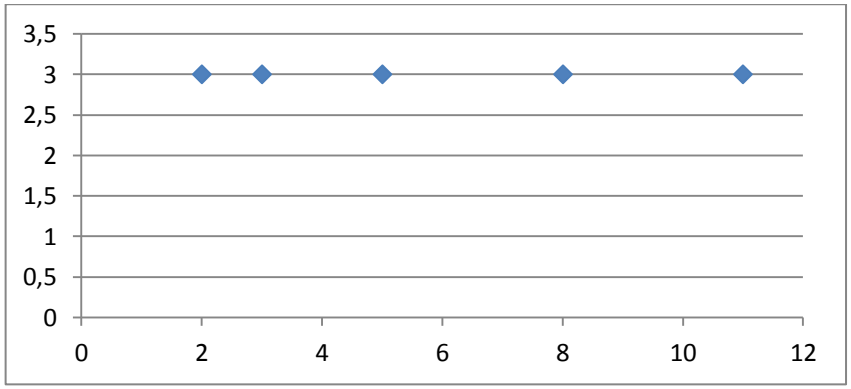
Şekil 4.10. Gradyan büyüklüklerin histogram bölmelerine dağılımı [20].

Gradyan büyüklüklerinden ve açılarından histogram oluşturulduktan sonra sınıflandırıcının ışık değişimlerinden etkilenmemesi için resim tekrar 16x16 büyüklüğünde pencerelere bölünür. Bölümlenen her pencere için histgoramlar yeniden normalize edilir ve SVM gibi öğrenme algoritmaları ile benzer nesneleri tanıyabilmek için kullanılırlar.

4.2. YÜZ TANIMA

4.2.1. Temel Bileşen Analizi ve Özyüzler Algoritması

Temel bileşen analizi (Principal Component Analysis - PCA); bir veri setinde bulunan en verimli temel bileşenlerin boyutlarının bulunmasını sağlayan ve bu sayede boyutların azaltılmasını sağlayan analiz türüdür. Varyans ise bir veri setinde verilerin dağılımını ölçmede kullanılan bir tanımdır. Bu nedenle varyans, temel bileşen analizinde temel bileşenlerin belirlenmesinin başlangıç noktasıdır. Varyans, veride yer alan bilgileri kodlar. İki boyutla tanımlanan veri setinde her noktanın tanımlanması için iki veri (x,y) gereklidir fakat veri kümesindeki tüm elemanların Şekil 4.11'de gösterildiği gibi sabit y ekseninde dizildiği düşünüldüğünde veri setinin tanımlanması için x yönündeki varyanslar yeterlidir.



Şekil 4.11. Değişken x ve sabit y düzleminde dizilmiş veri seti grafiği.

Temel bileşen analizinin tam olarak gerçekleştirdiği işlem de veri setinden anlamsız olan bilgileri atmak ve anlamlı bilgilerle veri setini tanımlamaktır.

Özyüz terimi ise yüz resimlerinden öz vektör ve öz değerlerin PCA ile analizinden ortaya çıkmış bir kavramdır. Örnek bir nxn boyutlu x matrisinde Ax = λx eşitliği sağlanıyorsa bu denklemde A matris özvektörünü λ ise matrisin öz değeri olarak tanımlanır. Örnek bir PCA hesaplaması aşağıda gösterilmiştir.

- Veri matrisinin oluşturulması: Veri setinde bulunan tüm değerlerin birleştirilmesi işlemidir.

$$D = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \end{bmatrix} \tag{4.10}$$

- Boyut ortalamalarının hesaplanması: Veri setindeki tüm verilerin ortalamasının alınması işlemidir. Veri seti boyutu kadar ortalama değer elde edilecektir.

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.11}$$

$$\mu_y = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.12}$$

$$\mu_z = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4.13}$$

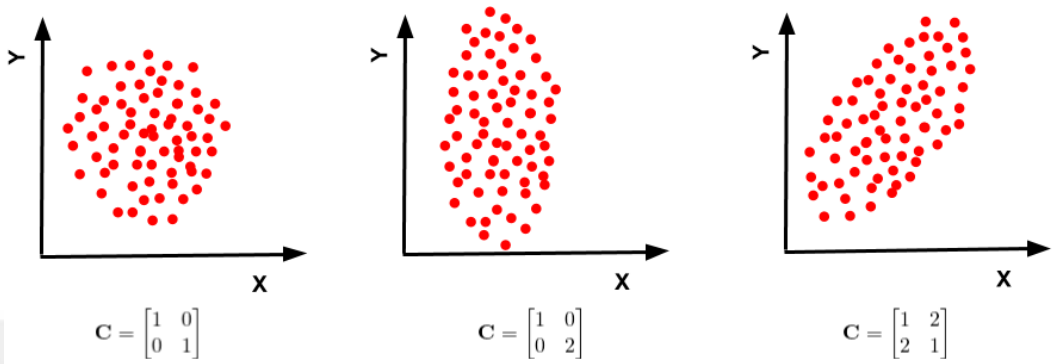
- Genel ortalamanın hesaplanması: Tüm boyut ortalamalarının ilgili veri noktalarından çıkarılması ile elde edilir.

$$M = \begin{bmatrix} (x_1 - \mu_x) & (x_2 - \mu_x) & \cdots & (x_n - \mu_x) \\ (y_1 - \mu_y) & (y_2 - \mu_y) & \cdots & (y_n - \mu_y) \\ (z_1 - \mu_z) & (z_2 - \mu_z) & \cdots & (z_n - \mu_z) \end{bmatrix} \tag{4.14}$$

- Kovaryans matrisinin hesaplanması: Kovaryans matris ortalamasının transpozu ile konvülasyonundan elde edilir ve maksimum varyans değişimleri kovaryans matristen elde edilir.

$$C = MM^T \tag{4.15}$$

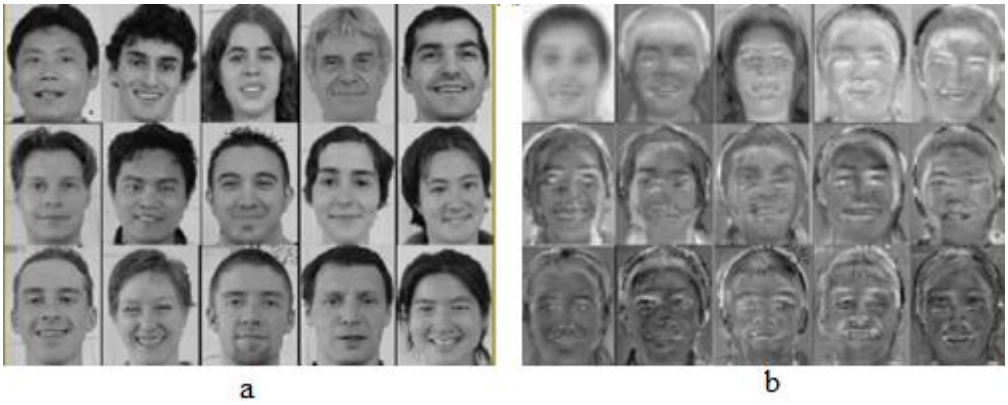
Veri seti varyans değişimlerinin kovaryans matrisi ile ilişkisi Şekil 4.12'de gösterilmiştir.



Şekil 4.12. Kovaryans matrisinin veri setine göre değişimi.

- Özvektörlerin ve öz değerlerin bulunması: Temel bileşenler kovaryans matrisinin öz vektörleridir. En büyük öz değere sahip olan en temel bileşen olarak belirlenirken öz değer ile doğru orantılı olarak temel bileşenler sıralanacaktır.

Yüz tanıma işlemi için tanınmak istenen tüm yüzler temel bileşen analizinde bir boyutu ifade etmektedir. Tanınmak istenen tüm yüzler bir diziye atandıktan sonra PCA analizi ile öz vektör ve özdeğer ağırlık matrisleri elde edilir ve kaydedilir. Tanınacak olan yeni yüz içinde aynı işlemler uygulanarak ağırlık matrisi elde edilir ve iki ağırlık matrisi arasındaki öklit uzaklığı resmin veri tabanı resimlerinden hangisine ne kadar benzediği hakkında bir sonuç üretecektir. Tanınacak yüz fotoğrafları veri seti ve bu fotoğraflardan oluşturulan özyüz resmi Şekil 4.13'te gösterilmiştir.



Şekil 4.13. a) Orijinal yüz fotoğrafları veri seti, b) özyüz vektör resimleri.

4.2.2. Konvolüsyonel Sinir Ağları İle Yüz Tanıma

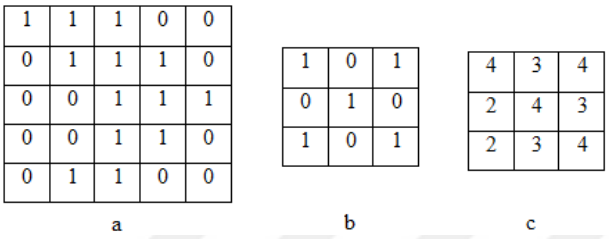
Konvolüsyonel sinir ağları insan sinir sisteminin algılama ve iletim yapısından ilham alınarak türetilmiştir. Bilgisayar görüsü ile sınıflandırma ve derin öğrenme konularının türlerinden biri olan konvolüsyonel sinir ağları, son zamanlarda sınıflandırma alanında en sık kullanılan ve en olumlu sonuçların alındığı işlemler bütünüdür.

Konvolüsyonel sinir ağları; sınıflandırma konvolüsyon katmanı, aktivasyon katmanı, havuzlama katmanı ve nöral ağ katmanlarından oluşur. Bu katmanlardan nöral ağ katmanı, eğitilebilmesi ve bu eğitim sonucu öğrenebilmesi nedeniyle algoritmanın en önemli aşamasıdır fakat nöral sinir ağları resimleri direk sınıflandırabilmek için kullanmak istersek resim üzerindeki tüm pikselleri ağa giriş olarak vermeniz gerekir. Bu tür bir süreç çok uzun matematiksel işlemler yanı sıra ağın eğitilebilmesini de imkansız hale getirecektir. Bu nedenle konvolüsyonel sinir ağlarında giriş resminin ağa uygun hale getirilmesi gerekmektedir. Resimden ağa uygun nöronların elde edilmesi için izlenecek adımlar sırasıyla aşağıda verilmiştir.

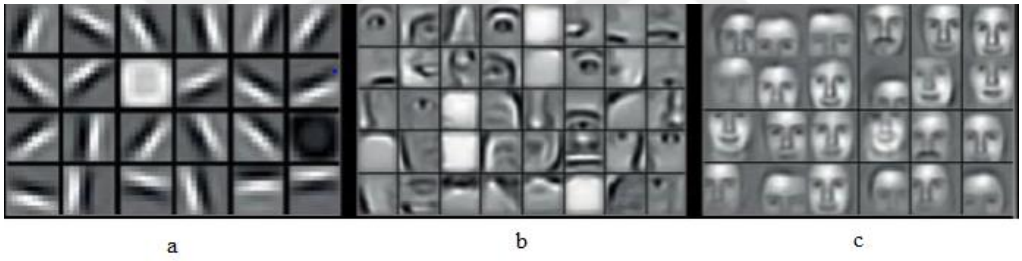
1. **Konvolüsyon Katmanı**

   Konvolüsyon katmanı algoritmanın ilk katmanıdır. İlgili resim üzerine uygulanan filtreler (çerçeveler veya çekirdekler) aracılığı ile hem resmin boyutunu azaltmaya hemde resim içerisinde bulunan anlamlı değerleri ön plana çıkarmaya yarar. Yüz tanıma bölümünde de belirtildiği gibi belirli

özellik penceleri (kenar bulma, köşe tespiti gibi filtre pencereleri) ilgili resmin üzerinde sol üst köşeden başlayarak 1 birim sağa hareket ettirilir ve iki resim arasındaki konvolüsyon sonucu çıkarılır. Şekil 4.14'te 5x5 boyutunda bir matris ile 3x3 boyutunda bir matrisin konvolüsyon sonuç matrisi; şekil 4.15'te ise yüz tanıma için kullanılan özellik berlirleme filtreleri gösterilmiştir.



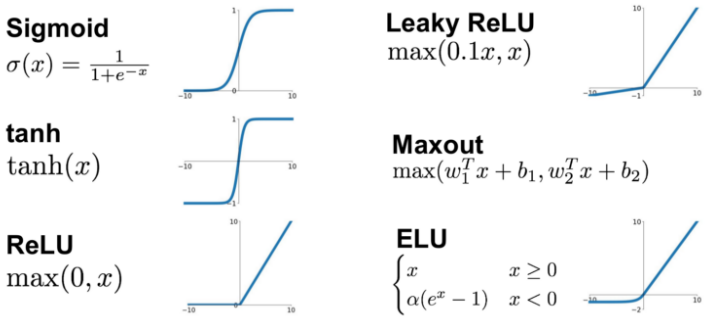
Şekil 4.14. a) İlgili resim matrisi, b) filtre matrisi, c) filtre konvolüsyon sonuç matrisi.



Şekil 4.15. a) Düşük seviye, b) orta seviye, c) yüksek seviye filtre çerçeveleri.

2. Aktivasyon Katmanı

Aktivasyon katmanı sinir ağlarının sonuna veya arasına eklenen bir düğümdür. Bu düğüm ağırlıklandırılmış nöronun bir sonraki çıkışta tetiklenip tetiklenmeyeceğini belirtir. Yapay sinir ağlarının eğitiminde kullanılan çeşitli aktivasyon fonksiyonları mevcuttur. Aktivasyon fonksyionlarının çeşitleri Şekil 4.16'da gösterilmiştir.



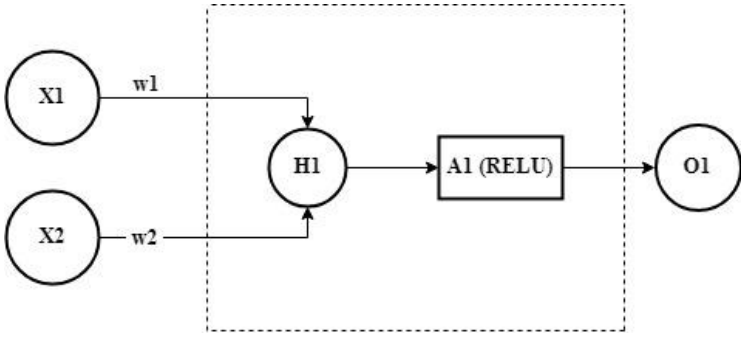
Şekil 4.16. Yapay sinir ağlarında kullanılan örnek aktivasyon fonksiyonları [21].

Kullanılacak aktivasyon fonksiyonu sinir ağının uygulanacağı sisteme göre faklılık gösterse de yapay sinir ağlarında uygun sonuçların hızlı bir şekilde üretilmesi için en çok kullanılan aktivasyon ReLU fonksiyonudur. Aşağıda ReLU fonksiyonunun yapay sinir ağlarında kullanımına ilişkin bir örnek gösterilmiştir. Örneğimizde doğruluk tablosu Çizelge 4.1'deki gibi olan yapay sinir ağının oluşturulmasını inceleyelim.

Çizelge 4.1 Örnek sinir ağı doğruluk tablosu.

X_1	X_2	Y
0	0	0
0	1	0
1	0	1
1	1	0

Şekil 4.17 de doğruluk tablosuna uygun olarak oluşturulacak yapay sinir ağı blok şeması gösterilmiştir. Burada "w1" ve "w2" nöronların ağırlıklarını "H1" gizli çıkış katmanını "A1" uygulanacak aktivasyon fonksiyonunu "O1" ise sonucu temsil etmektedir.



Şekil 4.17. Örnek yapay sinir ağı gösterimi.

Örnek sinir ağında w1 ve w2 nöron ağırlıklarını sırasıyla 1 ve -1 olarak belirlediğimizde her giriş için oluşacak ara katman sonuçları aşağıda hesaplanmıştır.

$$H_{1,1} = w_1 * X_{1,1} + w_2 * X_{2,1} = 1 * 0 + -1 * 0 = 0 \quad (4.16)$$

$$H_{1,2} = w_1 * X_{1,2} + w_2 * X_{2,2} = 1 * 0 + -1 * 1 = -1 \quad (4.17)$$

$$H_{1,3} = w_1 * X_{1,3} + w_2 * X_{2,3} = 1 * 1 + -1 * 0 = 1 \quad (4.18)$$

$$H_{1,4} = w_1 * X_{1,4} + w_2 * X_{2,4} = 1 * 1 + -1 * 1 = 0 \quad (4.19)$$

Hesaplanan tüm gizli katmanlar ReLU (negatif değerler için sıfır diğer değerler için girişe eşit çıkış) fonksiyonundan geçirildiğinde girişler için hesaplanan çıkış değerleri oluşturulan doğruluk tablosuna uygun olarak üretilmiş olacaktır.

3. Havuzlama Katmanı

Havuzlama katmanı elde edilen matrislerin özelliklerinin kaybedilmeden daha düşük boyutlu matrislere indirgenmesi ve bu sayede hız kazanılması için kullanılan bir katmandır. Havuzlama katmanı için genel olarak maksimum havuz katmanları ve ortalama havuz katmanları kullanılmaktadır. Maksimum havuzlama katmanında düşük boyutlu bir kare çerçeve resim içerisinde gezdirilerek çerçeve içerisindeki en yüksek değer alınırken, ortalama

havuzlama katmanında çerçeve içerisinde bulunan değerlerin ortalama değeri alınır. 26x26 boyutunda giriş matrisine 2x2 boyutunda maksimum havuzlama yöntemi uygulandıktan sonra elde edilen 13x13 boyutunda çıkış matrisi Şekil 4.18'de gösterilmiştir.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.4	0.6	0.7	0.5	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.3	0.6	1.2	1.4	1.6	1.6	1.6	1.6	1.9	1.9	2.2	2.3	2.1	2.0	1.7	0.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.5	1.2	1.8	2.6	2.7	3.0	3.0	3.0	3.0	3.4	3.5	3.8	4.0	3.7	3.6	3.2	2.3	1.5	0.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0
1.1	2.1	3.2	4.2	4.4	4.7	4.7	4.5	4.2	4.0	3.8	3.9	3.9	4.1	4.5	4.7	4.1	3.1	1.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
1.1	2.0	3.1	3.6	3.3	3.2	3.2	3.1	2.9	2.7	2.5	2.5	2.5	2.7	3.0	3.9	4.4	4.1	2.9	1.4	0.3	0.0	0.0	0.0	0.0	0.0
0.9	1.4	2.1	2.2	1.8	1.7	1.7	1.5	1.1	0.8	0.5	0.5	0.5	0.8	1.3	2.4	3.7	4.5	4.0	2.4	1.0	0.0	0.0	0.0	0.0	0.0
0.1	0.3	0.3	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.3	2.8	4.2	4.7	2.8	1.6	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	1.2	2.9	3.9	5.1	3.1	2.2	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.4	1.0	1.3	1.6	1.9	2.4	3.7	4.4	5.2	3.8	2.5	0.7	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.5	1.1	1.7	2.3	2.7	3.0	3.4	3.7	4.6	4.9	5.2	4.1	2.5	1.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.1	0.7	1.3	1.9	2.6	3.2	4.0	4.4	4.8	4.4	4.2	4.5	4.8	5.2	4.5	2.7	1.6	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.4	1.0	1.8	2.6	3.3	3.8	3.9	3.8	3.6	3.4	3.0	2.9	3.6	4.1	5.0	3.8	2.5	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.8	1.7	3.0	3.5	3.7	3.3	3.0	2.5	2.2	1.9	1.3	1.3	2.4	3.3	4.8	3.4	2.3	0.6	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.9	2.0	2.7	3.2	2.6	1.8	1.3	0.7	0.4	0.1	0.0	0.4	2.2	3.3	4.6	3.0	2.0	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.7	1.4	1.6	1.7	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.8	2.5	3.7	4.2	2.6	1.5	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.5	0.2	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.7	3.3	4.0	3.6	2.2	0.8	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	2.3	4.0	3.9	2.8	1.6	0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	2.3	3.1	4.5	3.4	2.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	2.6	3.4	3.8	2.5	1.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	2.0	2.8	2.4	1.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.3	2.0	1.3	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

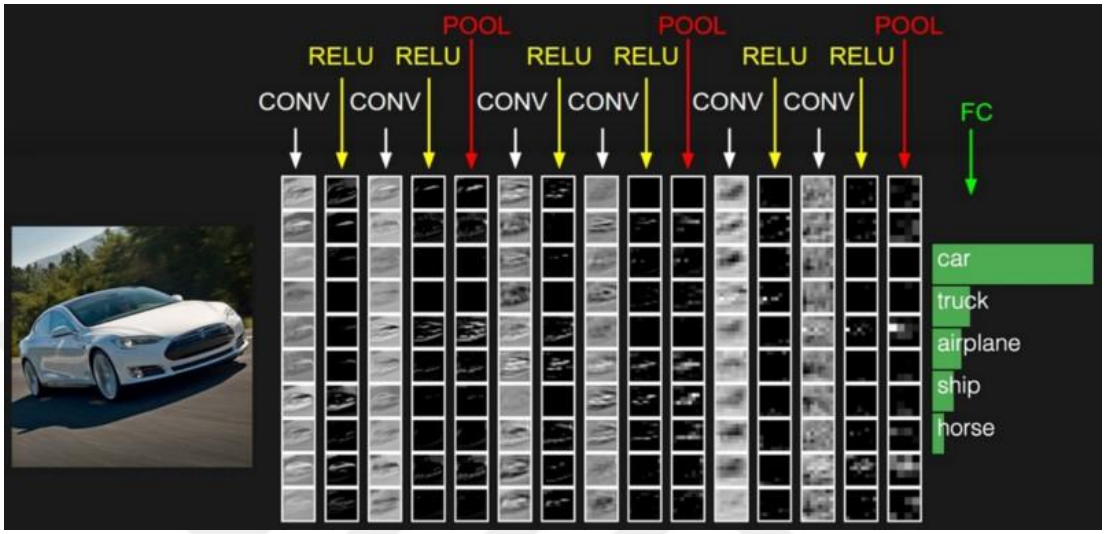
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.3	0.6	0.7	0.4	0.0	0.0	0.0	0.0	0.0
1.2	2.6	3.0	3.0	3.4	3.8	4.0	3.6	2.3	0.5	0.0	0.0	0.0
2.1	4.2	4.7	4.7	4.2	3.9	4.1	4.7	4.4	2.9	0.3	0.0	0.0
1.4	2.2	1.8	1.7	1.1	0.5	0.8	2.4	4.5	4.7	1.6	0.0	0.0
0.0	0.0	0.0	0.0	0.1	1.0	1.6	2.4	4.4	5.2	2.5	0.0	0.0
0.0	0.0	0.1	1.3	2.6	4.0	4.8	4.4	4.9	5.2	2.7	0.0	0.0
0.0	0.0	1.7	3.5	3.8	3.9	3.6	3.0	4.1	5.0	2.5	0.0	0.0
0.0	0.0	2.0	3.2	2.6	1.3	0.4	0.8	3.7	4.6	2.0	0.0	0.0
0.0	0.0	0.5	0.5	0.0	0.0	0.0	2.3	4.0	3.6	0.8	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.9	3.4	4.5	2.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.2	2.8	2.4	0.3	0.0	0.0	0.0

Şekil 4.18. Maksimum havuzlama kullanılarak filtrelenmiş örnek matris [22].

4. Yapay Sinir Ağı Oluşturulması ve Çıkış Katmanı

Konvolüsyonel sinir ağlarında amaç belirli özellikleri taşıyan matris değerlerinden sınıflandırılmış sonuçlar üretmektir. Yapay sinir ağı sınıflandırmanın eğitilerek çıkış katmanının oluşturulmasından sorumludur. Yapay sinir ağı ilk üç adımda belirtilen adımların birleşimidir. Sınıflandırılacak objeye uygun olarak ilk üç katman çoğaltılabilir. Sınıflandırılmış çıkışların elde edilmesi için sinir ağının nöron ağırlıkları geri

yayılım algoritması ile eğitilerek ağ oluşturulur. Eğitim sonucunda daha önceden eğitilmemiş bir matris ağa giriş olarak uygulandığında belirlenen ağırlıklara uygun olarak sınıflandırma çıkışı üretecektir. Örnek bir yapay sinir ağı akışı Şekil 4.19'da gösterilmiştir.



Şekil 4.19. Örnek konvolüsyonel sinir şeması [21].

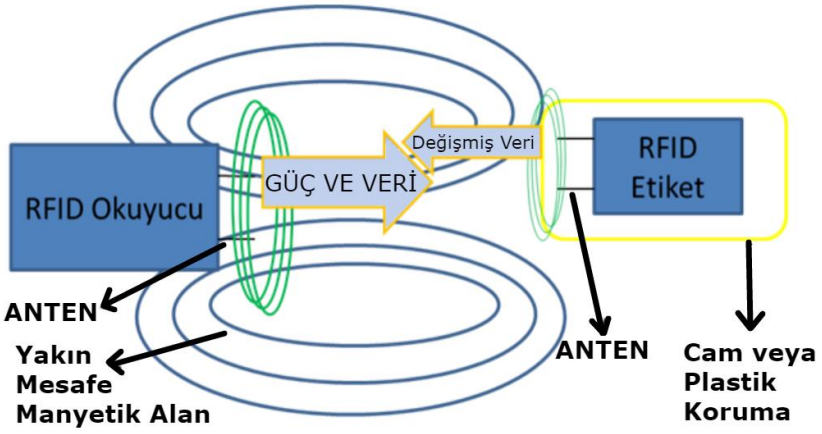
BÖLÜM 5

GEÇİŞ KONTROLÜNDE SIK KULLANILAN DİĞER YÖNTEMLER

Geçiş kontrol sistemleri başta işletmeler olmak üzere kullanıcı yetkilendirmesi ve izinsiz girişlerin engellenmesi için kullanılmaktadır. Günümüzde geçiş kontrolünde biyometrik sistemler yaygın olarak kullanılsa da geçmişten gelen kullanım alışkanlıkları da devam etmektedir. KVKK (Kişisel Verilerin Korunması Kanunu) gereğince kişilerden kimlik benzeri kartların talep edilmesi yasaklanmıştır. Özellikle işletmelere ziyarette bulunan kişilerden biyometrik bilgilerin talep edilmesi de mümkün olmayacağından geçiş kontrol sistemlerinde biyometrik tanımaya oranla daha az güvenli fakat hızlı çözümler kullanılmaktadır.

5.1. RFID TANIMA

RFID (Radio Frequency Identification – radyo frekanslı kimliklendirme) radyo frekansları ile canlıları veya nesneleri tanımak için kullanılan kablosuz haberleşme teknolojisidir. RFID sistemler basit anlamda kimlik bilgisini bulunduran bir etiket ve etiket üzerindeki bilgiyi okuyacak bir alıcıdan oluşur. RFID sisteminin çalışmasına ilişkin görsel Şekil 5.1'de gösterilmiştir.



Şekil 5.1. RFID sisteminin çalışma yapısı [23].

RFID teknolojisi günümüzde büyük alışveriş merkezlerinde, stok sayımlarında, hayvan takibinde, havayolu ve kargo hizmetlerinde ve personel takip sistemlerinde kullanılmakta ve kullanım sıklığı giderek artmaktadır. Bu teknoloji sayesinde insan gücü kaybı zamanla azaldığı gibi stok kontrolü, ürün takibi ve raporlar gibi ihtiyaç duyulan bilgiler de anlık ve hatasız olarak izlenebilmektedir.

RFID sistemler kullandıkları frekanslara göre 3'e ayrılmaktadırlar.

1. **Düşük Frekans**

   Düşük frekans (LF – Low Frequency) bandı 30 KHz ila 300 KHz arasındaki frekansları kapsar. 134 KHz'de çalışan RFID sistemler olsa da genellikle 125 KHz'de çalışan sistemler tercih edilmektedir. Bu band aralığı 10 cm'ye kadar kısa bir okuma kapsamı verir ve yüksek frekanslı modellerden daha yavaş okuma hızına sahiptir. Bu tür RFID sistemler genellikle erişim kontrolü ve evcil hayvan takip sistemlerinde kullanılırlar.
2. **Yüksek Frekans**

   Yüksek frekans (HF – High Frequency) bandı 3 MHz ile 30MHz arasındaki frekansları kapsar. Çoğu HF sistemler 13,56 MHz frekansında çalışır ve 10cm ile 1m okuma aralığında çalışırlar. NFC (near field communication – yakın alan iletişim) protokolü yüksek frekans bandına örnek protokoldür ve günümüzde cihazlar arası iletişimde ve ödeme sistemlerinde kullanımı yaygınlaşmaktadır.
3. **Ultra Yüksek Frekans**

   Ultra yüksek frekans (UHF – Ultra High Frequency) bandı 300 MHz ile 3GHz arasındaki frekansları kapsar. Ülkeler arasında değişiklik göstersede UHF Gen2 standardı genel 900 ve 915 MHz aralığında çalışırlar. Okuma aralığı 12m mesafesinde olabilir.

RFID teknolojisinin kategorizasyonu frekans haricinde etiket özelliğine göre de yapılmaktadır. RFID sistemler, kullanılan etiketlerin enerji elde etme özelliklerine göre 3'e ayrılmaktadır.

1. **Pasif Etiket**

   Pasif etiketler kendilerine ait güç kaynağı olmayan etiketlerdir. Bu nedenle etiketler haberleşebilmek için gerekli enerjiyi okuyucu anteninden yayınlanan enerjinin etiket antenlerinde endüklemesinden elde ederler. Elde edilen enerji etiket içerisindeki çipi harekete geçirir ve çip içerisindeki bilgi tekrardan antenler ile okuyucuya gönderilir.

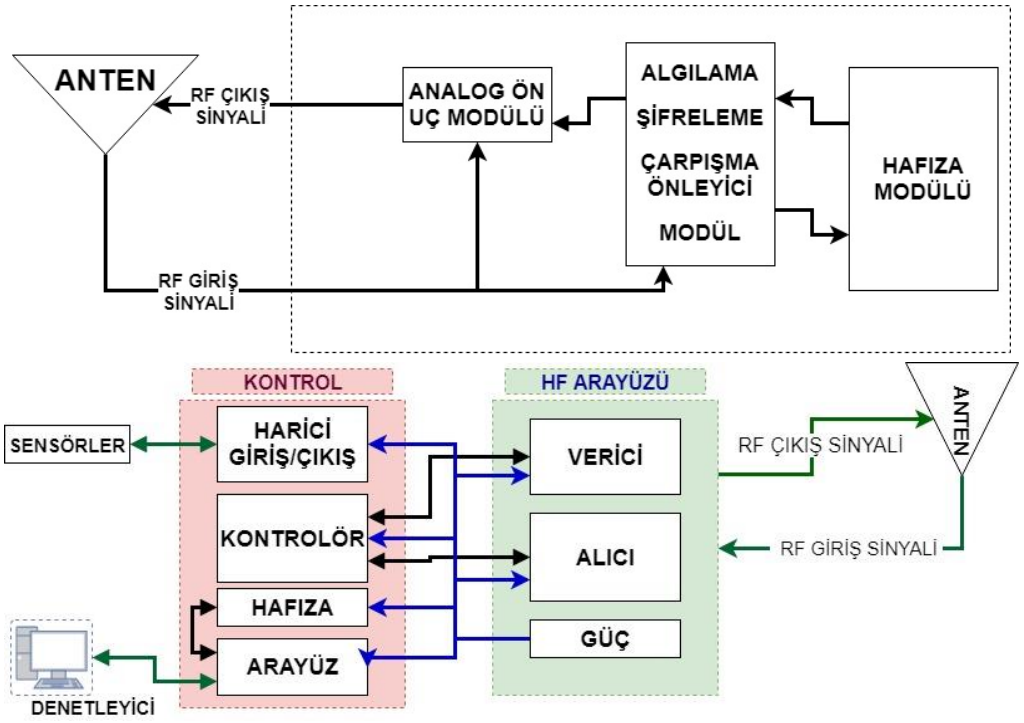
2. **Yarı Pasif Etiket**

   Yarı pasif etiketler kendilerine ait güç kaynağı olan etiketlerdir. İletişime geçme konusunda pasif etiket gibi davranarak okuyucudan sorgulama beklerler. Veri gönderimi için kendilerine ait güç kaynaklarını kullandıklarından pasif etiketlere göre daha uzak mesafelerde iletişim sağlarlar.

3. **Aktif Etiket**

   Aktif etiketler kendilerine ait güç kaynakları olan etiketlerdir. Okuyucu ile iletişime geçmek için okuyucu sorgulamasını beklemeden okuyucuya sinyal gönderirler. Okuma mesafeleri uzundur fakat maliyetleri yüksektir.

RFID etiket ve okuyucuların iç yapısı Şekil 5.2'de gösterilmiştir.



Şekil 5.2. RFID etiket ve okuyucu iç yapısı [23].

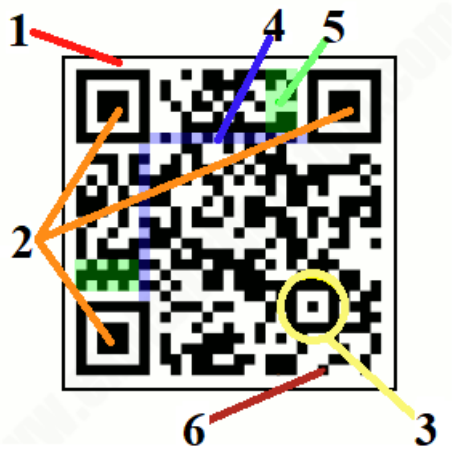
5.2. KAREKOD TANIMA

Karekod teknolojisi 1994 yılında Toyota'nın alt firması Denso Wave tarafından monte edilen araçların ve araçlara monte edilen parçaların hızlıca taranarak takip edilebilmesi için geliştirilmiştir.
Karekodlar barkod teknolojisinden farklı olarak iki boyutlu veri mimarisine sahiptirler ve bu sayede çok fazla veriyi karekod içerisinde saklamak mümkündür. Karekodlar çok fazla veriyi saklayabilmesi sebebiyle hata giderme ve şifreleme gibi teknikler ile veri aktarımına imkan sağlamaktadır.

Karekodlar QRKod, AztecKod, MaxiKod gibi birçok alt kategoriye ayrılmış olsa da günümüzde ençok kullanılan karekod teknolojisi qrkod (quick response code) teknolojisidir. Şekil 5.3'te karekod çeşitlerine ait örnekler gösterilmiştir.

Şekil 5.3. Karekod çeşitlerine ait örnek görseller [24].

Karekod; adını içerisinde bulunan ve karekodu oluşturan şekillerden almaktadır. Karekodu oluşturan tüm şekiller karekod hakkında bilgiler içermektedir. Karekod teknolojisi ile üretilmiş bir qrkod örneği ve özellik sınıflandırması Şekil 5.4'te gösterilmiştir.



Şekil 5.4. Karekod teknolojisi üretilmiş qrkod örneği [25].

Şekil 5.4'te numaralandırılmış her bir şekil qrkodun içeriği, yönü ve hata gidermeleri hakkında bilgiler içermektedir. Yapılan numaralandırmaların özellikleri sırasıyla aşağıda açıklanmıştır.

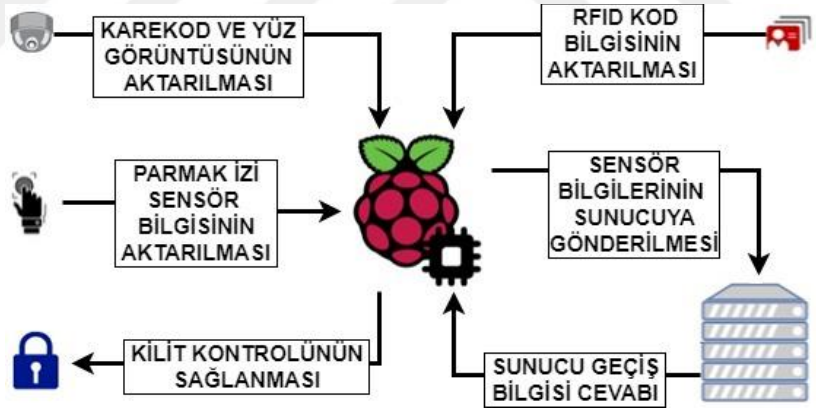
1. Karekodun çevresinde bulunan ve içeriğinin hatalı okunmasına neden olabilecek hataları gidermek için kodu çevreleyen beyaz çerçevedir.
2. Üç köşede bulunan büyük kareler kodun bir qrkod olduğunun belirlenmesini ve kodun hangi açıda olduğunun anlaşılmasını sağlar.
3. Kod basım yüzeyi hatalarından dolayı belirli bir miktar bozulmuş olsa da kodun yeniden düzeltilebilmesini sağlar.
4. Zamanlama kalıbı üç köşe deseni arasında yer alır ve sırayla siyah beyaz karelerden oluşur. Bu özellik karekodun her bir hücresini tanımlamayı kolaylaştırır ve kod bozulduğunda veya deforme olduğunda kullanışlıdır.
5. Qrkodun versiyon bilgisini tutan bölümdür.
6. Qrkodun içerik bilgilerini oluşturan hücrelerdir.

Qrkod yüzeysel hataları giderebilmesi, yüksek miktarda veri taşıyabilmesi ve hızlıca taranabilmesi sayesinde günümüzde sıkça kullanılmaktadır. Türkiye'de ilk olarak 27607 nolu Resmi Gazete'de yayınlanan yönetmelikle eczacılık alanında ilaçların takibi için kullanılmaya başlanmıştır. Son zamanlarda bankacılık alanında (para çekme, ödeme ve transfer gibi işlemlerde) ve her türlü biletleme (sinema, otobüs, uçak vb gibi) işlemlerinde sıklıkla kullanılmaktadır.

BÖLÜM 6

AMACI VE GERÇEKLENMESİ

Bu çalışma ile işletmelerde personel ve ziyaretçi takibini ve kontrolünü online olarak sağlayabilecek tümleşik bir çözüm elde edebilmek üzere planlamış ve geliştirilmiştir. Piyasada bulunan PDKS sistemler RFID, yüz tanıma, parmak izi tanıma ve karekod okuma işlemlerini gerçekleştirebilecek entegre çözüm sunmamakla birlikte bu özelliklerden birkaçını bünyesinde bulundurabilen yerli üretim bir PDKS ürünü de mevcut değildir. Günümüz teknolojisinin önemli konularından IoT (internet of things) ve görüntü işleme projelerine de altyapı oluşturması da çalışmanın bir diğer hedefidir. Çalışma sistemine ait akış şeması Şekil 6.1'de gösterilmiştir.



Şekil 6.1. Proje çalışma sistemi akış şeması.

Şekil 6.1'de gösterildiği gibi parmak izi sensöründen alınan parmak izi bilgisi, RFID etiket okuyucudan alınan etiket bilgisi ve kameradan alınan yüz görüntüleri ile karekod görüntüleri SoC (system on chip) kartına aktarılmaktadır. Kameradan alınan karekod ve yüz görüntüleri SoC tarafından çözülerek web servisler aracılığı ile sunucuya gönderilmektedir. Parmak izi ve RFID sensörlerden gelen kimlik ve kod bilgileri de veri tabanı işlemleri ile sunucu üzerinde işlendikten sonra gerekli çıkış bilgisi tekrardan SoC karta döndürülmekte ve gerekli çıkış sinyali üretilerek ilgili kilit mekanizması kontrol edilmektedir.

6.1. KULLANILAN DONANIMLAR

6.1.1. Raspberry Pi 3 Model B+

Raspberry Pi çeşitli özellikte giriş çıkış pinleri barındıran ve çevre birimleri ile donatılmış, tak çalıştır şeklinde üretilmiş, üzerinde çeşitli işletim sistemlerini barındırabilecek altyapı ile tasarlanmış maliyeti ucuz bir mini bilgisayardır. Raspberry Pi 3 Model B+ modeli 1.4GHz hızında Arm Cortex-A53 64 bit işlemci, 1GB DDR2 ram, 2.4GHz ve 5GHz destekli kablosuz ağ bağlantısı, bluetooth 4.2 ve BLE desteği, 40 pin GPIO, HDMI çıkışı, 4 adet USB2.0 portu, CSI kamera bağlantısı, SD kart yuvası ve ethernet bağlantı portu bulunmaktadır. Altyapısı sayesinde Linux, Android ve IoT cihazlar için özel olarak geliştirilmiş Win10 IoT Core işletim sistemlerini çalıştırabilmektedir. Mikrokontrolcülerin hızının yetmediği multithreading çalışma ve işletim sistemi üzerinde çalışan programlama dilleri (python, java, c#, c++ vb gibi) ile çalışabilmesi sayesinde IoT projelerinde ve küçük çaplı resim işleme projelerinde gün geçtikçe artan bir şekilde kullanılmaktadır. Şekil 6.1 çalışma akış şemasında da görüldüğü üzere Raspberry Pi sensörlerin internete açılmasını sağlayan internet nesnesi olarak kullanılmıştır.

6.1.2. CSI Kamera Modülü

CSI bağlantı noktası sayesinde Raspberry Pi ile tam uyumlu kamera modülüdür. IMX219PQ CMOS görüntü algılayıcısı, 8 megapiksel çözünürlüğü, 30fps 1080p ve 60fps'de 720p görüntü kalitesi sunabilmektedir. Kamera modülü proje içerisinde karekod ve yüz karelerinin alınarak Raspberry Pi'ye aktarılmasını sağlamaktadır.

6.1.3. GT511C3 TTL Parmak İzi Sensör

Parmak izlerinin üzerinde bulunan görüntü sensörü ve işlemcisi sayesinde yüksek doğruluk ve hızla tespitini sağlayan modüldür. Modül üzerinde Arm Cortex M3 işlemci bulundurmaktadır. 450dpi kalitesinde ve 202x258 piksel boyutunda görüntü alabilmekte, birebir parmak izi doğrulama ve çoklu parmak izi karşılaştırma yapabilmekte, UART seri haberleşme protokolü ile 11520 baud hızında

çalışabilmekte ve 200 adet parmak izini kendi rom belleğinde saklayabilmektedir. UART haberleşme standardı ve bilgi kartında [26] belirlenen protokolleri ile yeni parmak izi kaydetme, kaydedilen parmak izlerinin şablonlarını dışa ve içeri aktarma, parmak izi görüntüsünü elde etme gibi birçok fonksiyonu da bulunmaktadır. Parmak izi sensörü parmak izlerinin kaydedilmesi ve eşleşen kişilerin kimlik bilgilerinin Raspberry Pi'e aktarılması işlemlerini gerçekleştirmektedir. Ayrıca kullanılan yüz tanıma özelliği sayesinde cihazın saklayabileceği maksimum 200 parmak izinin adedinin şablon veri değişimleri ile sınırsız olarak kullanılabilmesi de sağlanmıştır.

6.1.4. RC522 RFID Kart Okuyucu

Modül okuma ve yazma için 13.56MHz frekansında NFC standardında kablosuz haberleşme yapmaktadır. Diğer CPU cihazlarla SPI haberleşme protokolünü kullanarak 424 kbit/s haberleşme hızı ile doğrudan iletişim kurabilir. 3.3V besleme gerilimine ihtiyaç duymaktadır. Maximum 6cm'ye kadar okuma yazma işlemi yapabilir. RC522 projede personeller ile eşleştirilen etiket eşsiz kimlik değerleri erişim yetki kontrolü için kullanılmıştır.

6.1.5. Kullanılan Diğer Malzemeler

Tanıma işlemleri sonrası geçiş izni verilen kişilerin geçişlerinin sağlanabilmesi için sürücü kartı malzemelerine ve mekanik kilit sistemine de ihtiyaç duyulmaktadır. Röle sürücü kartı için bc528 transistör, koruma diyodu, dirençler, röle, röle kartının kontrol edeceği selenoid kilit ve kilit için12V harici besleme kaynağı kullanılmıştır.

6.2. GELİŞTİRİLEN YAZILIMLAR

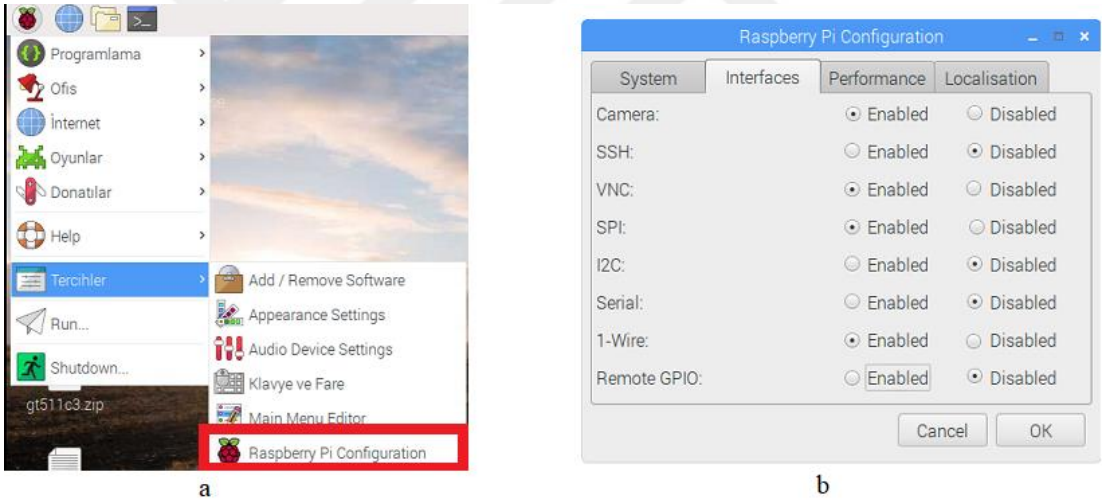
6.2.1. Gerekli Kurulumlar ve Yapılandırmalar

Sistemin geliştirilmesi için kullanılacak veritabanı Sql Server 2017 Express sürümü olarak seçilmiştir. Sql Server 2017 Express sürümü Microsoft firması tarafından geliştirilen geniş kullanım alanına sahip (özellikle modelleme ve prototip

oluşturmada) ücretsiz veri tabanıdır. Veri tabanı Microsoft resmi sitesinden kullanılacak bilgisayar özelliklerine uygun olacak şekilde indirilip yüklenebilir.

Bulut sunucu üzerinde çalışacak yazılımın geliştirilmesi için yine Microsoft tarafından geliştirilen Visual Studio 2017 Express sürümü kullanılmıştır. Sql Server Express gibi Visual Studio Express sürümü de ücretsiz olarak Microsoft resmi sitesinden indirilerek kullanılabilir.

Raspberry Pi SoC kartı için Raspberry Pi firması tarafından üretilen Raspbian işletim sistemi indirilerek SD kart içerisine kurulmuş ve Raspberry Pi SD kart girişine takılmıştır. Raspberry Pi üzerinde bulunan HDMI portu ile monitöre bağlanılarak gerekli ayarları yapılmıştır. Çalışmada gerekli olan özellik yapılandırması Şekil 6.2'de gösterilmiştir.



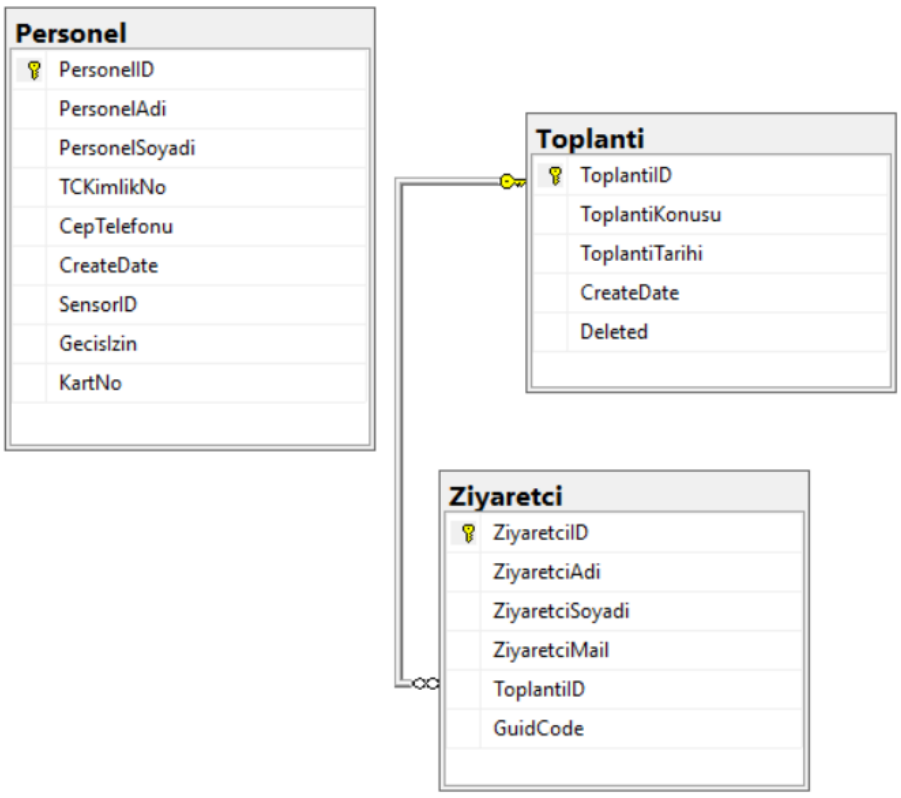
Şekil 6.2. a) Raspberry Pi ayar giriş ekranı, b) ayar yapılandırması.

Şekil 6.2 b'de görüleceği üzere Raspberry Pi bağlantısı için "VNC" aktif hale getirilmiştir. "VNC" uygulaması sayesinde Raspberry Pi cihazına uzak bağlantı yapılabilmektedir. Ayrıca çalışmada kamera ve haberleşmeyi SPI üzerinden yapan RFID modül kullanıldığından bu konfigürasyonlar da aktif duruma getirilmiştir. Raspberry'e uzak masaüstü yapılacak bilgisayarda "VNC Viewer" kurulması ve Raspberry'nin bağlı olduğu modemden almış olduğu IP adresinin bilinmesi

gereklidir. IP adresi Raspberry içerisindeki "VNC Server" uygulaması çalıştırılarak veya terminal ekranına "ifconfig" komutu yazılarak öğrenilebilir.

6.2.2. Veri Tabanı Data Model Oluşturulması

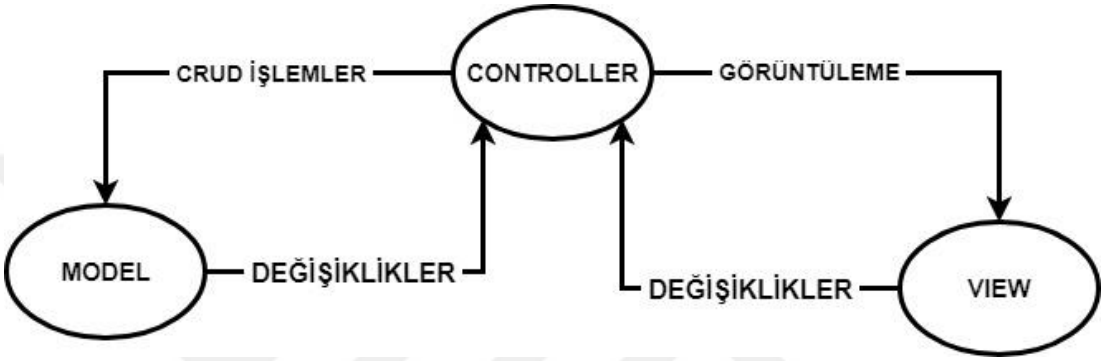
Gerçeklenen sistemde personel ve ziyaretçi bilgilerinin tutulması için Şekil 6.3'te 3 tablodan oluşan data model gösterilmiştir. Personel modeli ad, soyad, TC kimlik numarası, cep telefonu numarası, geçiş izninin olup olmadığı, ve yetkili personellerin erişim yetkilendirmesi için RFID kart numarasının saklandığı alanlardan oluşmaktadır. Toplantı tablosunda gerçekleştirilecek toplantının konusu ve ne zaman gerçekleşeceği bilgisi saklanmaktadır. Giriş yapmak isteyen ziyaretçinin o gün içerisinde toplantısı olup olmadığı bu tablo üzerinden kontrol edilecektir. Ziyaretçi tablosu ziyarete gelecek kişilerin adı, soyadı, karekodun gönderileceği mail adresi ve karekodun içerik bilgisinin bulunduğu alanlardan oluşmaktadır. Ayrıca tüm modellerde diğer modellerle ilişkisinin tanımlanabilmesi için otomatik artan benzersiz değer alanları da bulunmaktadır.



Şekil 6.3. Geçiş kontrol sistemi veri tabanı data modeli.

6.2.3. Web Sunucu Yazılımı Geliştirilmesi

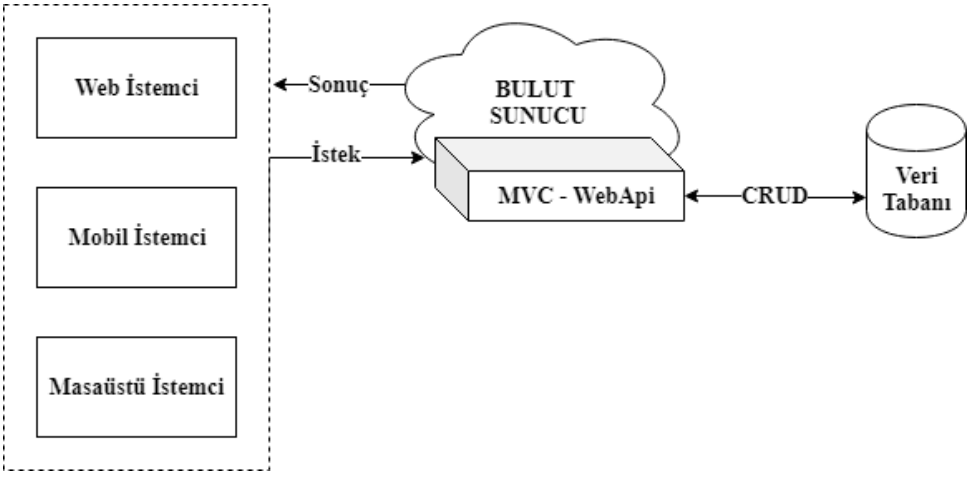
Gerçeklenen geçiş sistemine entegre edilen web sunucu yazılım teknolojileri sayesinde sistemin internet nesnesi olarak çalışması sağlanmıştır. Web sunucu yazılımı, Asp.Net frameworkü, MVC (model-view-controller) ve WebApi teknolojileri kullanılarak geliştirilmiştir. Şekil 6.4'te MVC mimarisine ait akış şeması gösterilmiştir.



Şekil 6.4. MVC mimarisi ve çalışma yapısı.

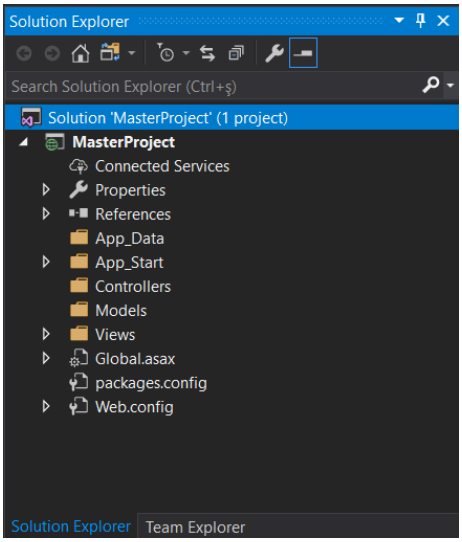
MVC teknolojisi görsel arayüzlerin hazırlanmasında ve veri modellerinin veri tabanı ile ilişkilendirilmesinde, CRUD (ekleme, okuma, güncelleme, silme) işlemlerinin gerçekleştirilerek arayüzlerin değiştirilmesi işlemleri için tanımlanmış bir yazılım standardıdır. Proje içerisinde görsel arayüzün hazırlanması ve veri tabanı işlemlerinin gerçekleştirilmesi MVC deseni üzerine inşa edilen .NET MVC 5.0 teknolojisi ile geliştirilmiştir.

Web servis; tüm programlama dillerinde ortak görülen veri yapılarının (json, xml, csv vb gibi) http protokolü üzerinden taşınarak platformlar arası bilgi alışverişi sağlayan yapılara verilen genel isimdir. Haberleşme protokolü olarak json veri türünü kullanan yapılar WebApi olarak adlandırılır. Şekil 6.5'te WebApi çalışma yapısı gösterilmiştir.



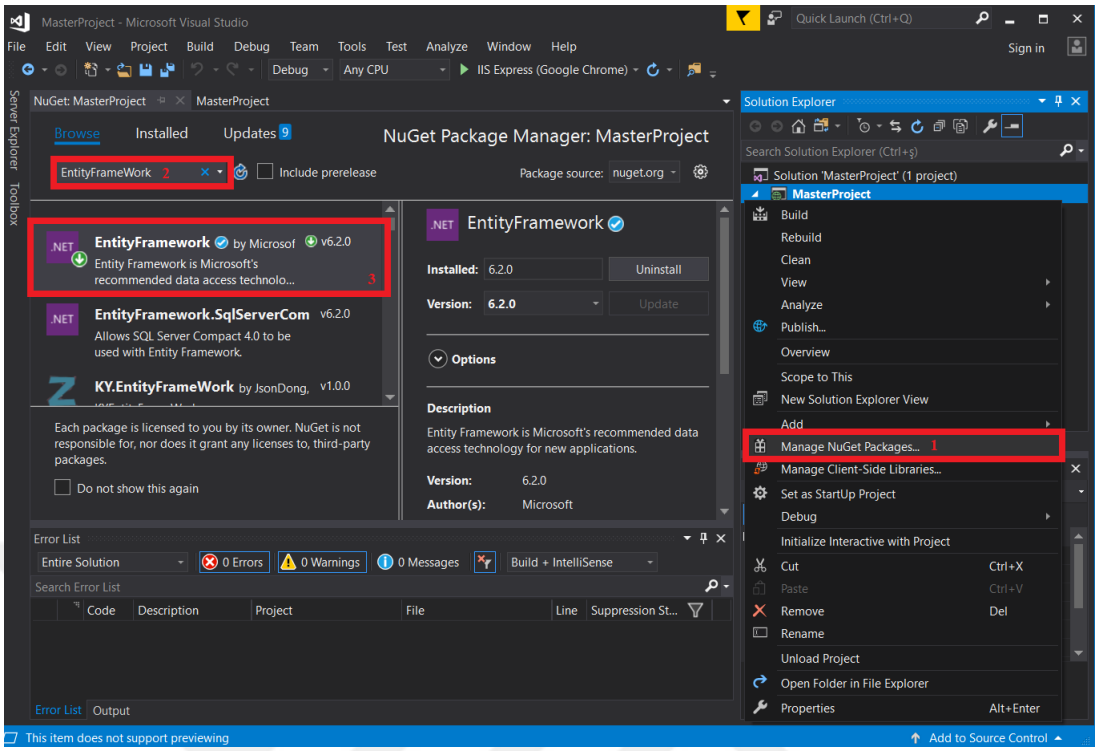
Şekil 6.5. WebApi çalışma yapısı.

Veri tabanının tasarlanan data modeller üzerinden oluşturulması, veri tabanı tablo ve kolon eşleşmelerinin sağlanması için "EntityFramework" ORM kütüphanesi (object related mapping) ve "CodeFirst" metodolojisi kullanılmıştır. Visual Studio ortamında MVC – WebApi uygulaması oluşturmak "File > New Project" admları izlenir, ekranda "Asp.Net Web Application" seçildikten sonra çıkan ekranda MVC ve WebApi kutucukları işaretlenerek işlem tamamlanır. Şekil 6.6'da oluşturulan MVC projesi dosya yapısı gösterilmiştir.



Şekil 6.6. Visual Studio ortamında oluşturulan MVC projesi.

EntityFramework kütüphanesinin "Nudget" üzerinden projeye dahil edilmesi Şekil 6.7'de gösterilmiştir.

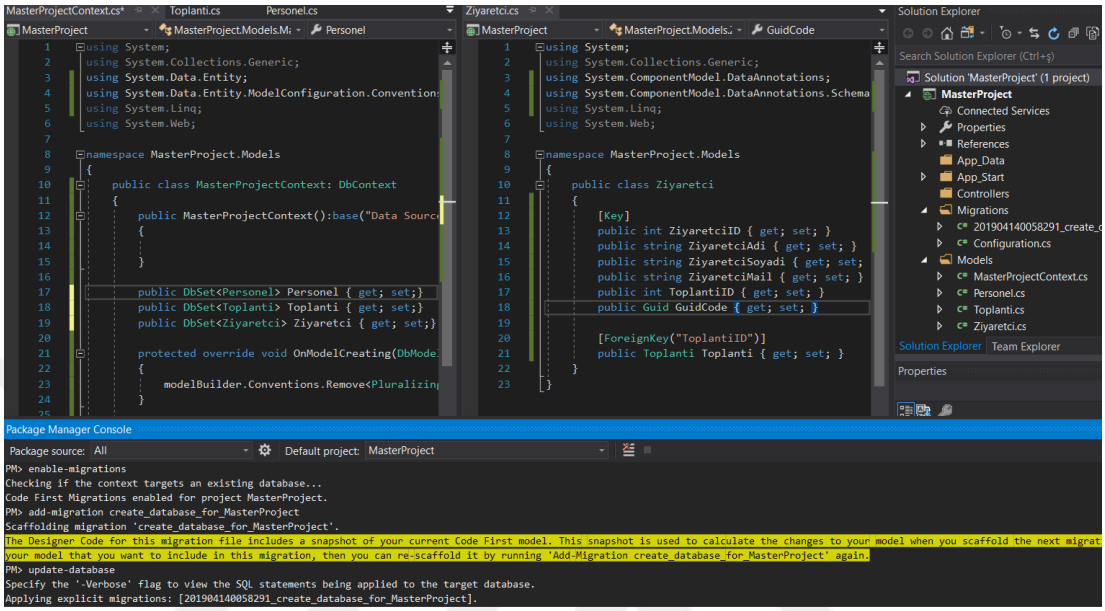


Şekil 6.7. Nudget üzerinden EntityFramework kütüphanesi yüklenmesi.

Veri tabanı data model tanımlamasının EntityFramework ile veri tabanına yansıtılması ve eşleştirilmesi işlemleri için model sınıflarının oluşturulması gerekmektedir. Data model sınıfları oluşturulduktan sonra veri tabanı bağlantısını sağlayacak olan "DbContext" sınıfından türeyen obje oluşturulur ve bu obje üzerinde veri tabanında oluşturulacak sınıflar tanımlanır. Belirlenen sınıfların veri tabanına yansıtılması için Visual Studio "Package Manager Console" (PM) kullanılır. Console üzerinde sırasıyla aşağıda verilen komutlar çalıştırılarak veri tabanı oluşturma ve eşleştirme işlemleri tamamlanır.

- Enable-migrations : Bu komut mvc projesinde migration özelliğinin açılmasını sağlar. Komut çalıştırıldığında proje klasör yapısına "Migrations" isimli yeni bir klasörün ve gerekli dosyalarının oluşturulduğu görülecetir.
- Add-migration [migration adı] : Belirlenen isimde yeni bir migration oluşturur.
- Update-database : Bu komut ile oluşturulan migration veri tabanına yansıtılarak modellerden veri tabanı oluşturma işlemi tamamlanır.

Şekil 6.8'de "Ziyaretci" data modeline uygun olarak oluşturulan sınıf içeriği, veri tabanına uygulanacak değişikliklerin tutulduğu DbContext objesi içeriği, son oluşan proje klasör yapısı ve PM üzerinde çalıştırılan komut görüntüleri gösterilmiştir.



Şekil 6.8. Veri tabanı sınıf yapılarının ve context objesinin oluşturularak PM aracılığı ile değişikliklerin veri tabanına aktarılması.

Veri tabanı oluşturulduktan sonra ziyaretçi ve personel giriş izinlerini, yetkili personel kontrolünü ve yeni kullanıcı eşleştirmesi işlemlerini Raspberry'den gelen parametrelere uygun olarak CRUD işlemlerini gerçekleştirecek WebApi uç nokta metodları hazırlanmıştır. Oluşturulan uç nokta için tanımlanan metodlar parametreleri ve dönüş değerleri Çizelge 6.1'de verilmiştir.

Çizelge 6.1 WebApi uç nokta metodları ve dönüş değerleri.

Metod Adı	Parametreler	Örnek Dönüş Değeri (json)	İstek Türü
PersonelGecisKontrol	PersonelID	{"PersonelAdi":"Alper" ,"PersonelSoyadi" : "Akkaya" ,"GecisIzin" : true}	GET
YetkiKontrol	KartNo	{"isOk":true,"Message":null,"IslemID":0}	GET
PersonelKayit	bs64FaceImage, bs64FPData, TCKN	{"isOk":true,"Message":null,"IslemID":0}	POST
ZiyaretciGirisKontrol	GuidCode	"true"	GET

WebApi metodları ile personel geçiş kontrolü, personel eşleştirmesi ve ziyaretçi geçiş kontrolü veri tabanı sorgulamalarıyla gerçekleştirilebilmektedir. Fakat bu kontrollerin gerçekleştirilebilmesi için veri tabanı personel ve ziyaretçi kayıtlarının web arayüzünden oluşturulması gerekmektedir. Bu nedenle Html, CSS, jQuery ve .Net MVC kullanılarak listeleme ve kayıt form web arayüzleri de oluşturulmuştur.

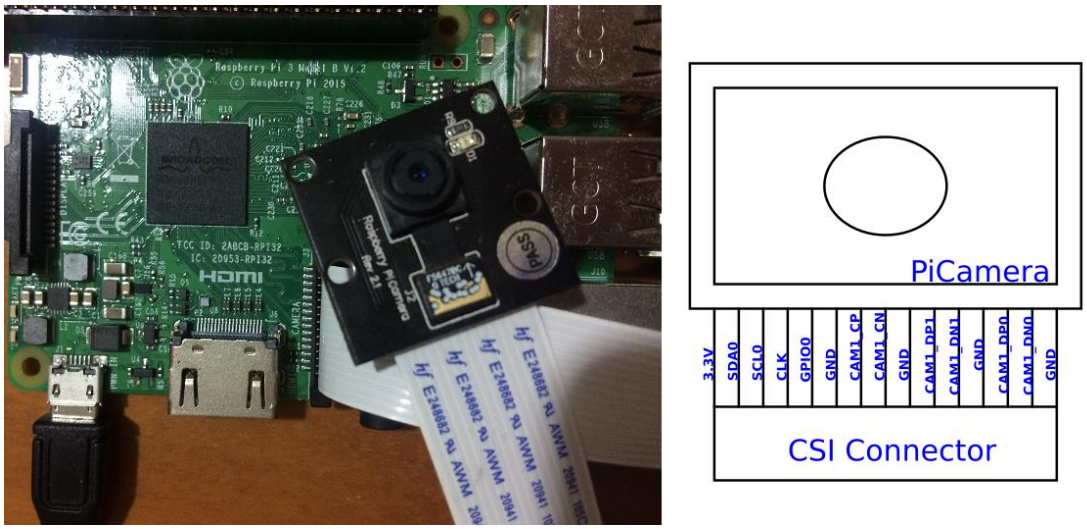
6.2.4. İstemci Yazılımının Raspberry Pi Üzerinde Geliştirilmesi

Şekil 6.1'de görüldüğü üzere Raspberry Pi gerçek dünyadan sensörler aracılığı ile aldığı verileri Şekil 6.5'te gösterilen servise göndererek dönen cevaba uygun olarak çıkış üretecektir. Bu işlemleri yerine getirecek program Raspbian işletim sistemi üzerinde Python dili ile geliştirilmiştir. Kamera, parmak izi sensörü, rfid okuma ve karekod çözme işlemleri için uygun kütüphanelerinin hazır bulunması Python dilinin seçilmesinde en önemli etken olmuştur. Tez çalışmasında Raspberry Pi ile kullanılacak modüller ve bulut sunucu ile haberleşmede kullanılacak kütüphaneler aşağıda listelenmiştir.

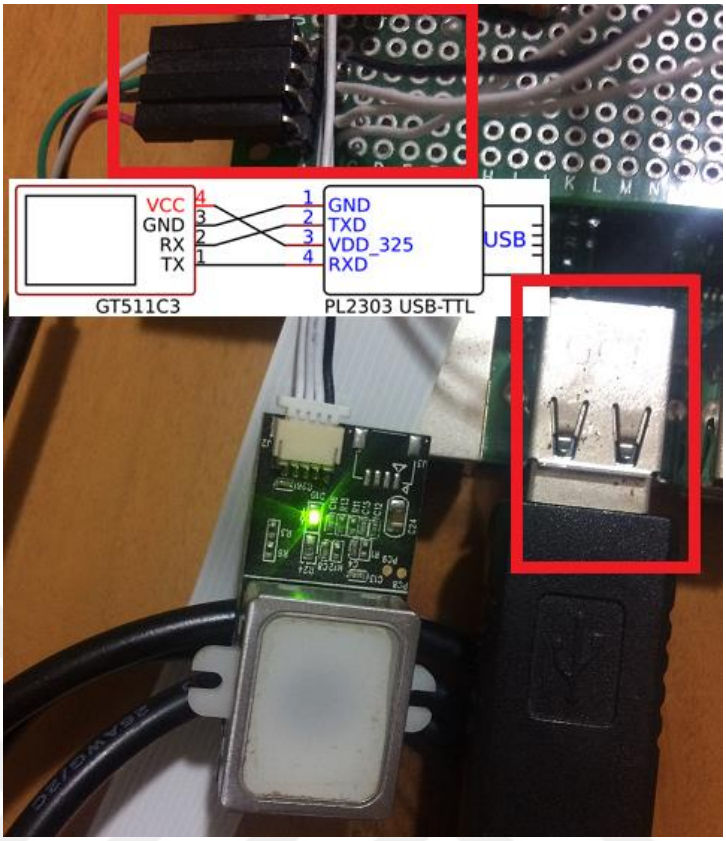
- OpenCV : PiCamera modülü ile bağlantının sağlanarak görüntülerin alınması için kullanılan kütüphanedir. Kütüphane, Python dili için Raspberry Pi üzerinde derlenerek kurulmuştur [27,28]. Kütüphane geliştirilen uygulama içerisine "import cv2" başlık bilgisi ile aktarılmaktadır.
- fingerpi : GT511C3 parmak izi okuyucu ile haberleşme için kullanılan Python kütüphanesidir [29]. Kütüphane Github deposundan indirilerek Raspberry üzerinde proje dosyaları içerisine klasör olarak kopyalanmış ve uygulama içerisine "import fingerpi" başlık bilgisi ile aktarılmaktadır.
- pi-rc522 : RC522 RFID okuyucu modülü ile haberleşmek için kullanılan Python kütüphanesidir [30]. Kütüphane terminal ekranından "pip install pi-rc522" yazarak Raspberry üzerine kurulmuş ve uygulama içerisine "from pirc522 import RFID" başlık bilgisi ile aktarılmaktadır.
- pyzbar : Resim karelerinden karekodun elde edilebilmesi için kullanılan Python kütüphanesidir [31]. Kütühane terminal ekranından "pip install pyzbar" yazarak Raspberry üzerine kurulmuş ve uygulama içerisine "from pirc522 import RFID" başlık bilgisi ile aktarılmıştır.

- face_recognition : Resim kareleri üzerinde yüz algılama ve yüz tanıma işlemlerini gerçekleştiren “dlib” derin öğrenme kütüphanesi üzerine geliştirilmiş bir kütüphanedir [32]. Kütüphane terminal ekranından “pip install face_recognition” yazarak Raspberry üzerine kurulmuş ve uygulama içerisine “from pirc522 import RFID” başlık bilgisi ile aktarılmıştır.
- requests : WebApi ile haberleşme için kullanılan kütüphanedir [33]. Kütüphane terminal ekranından “pip install requests” yazarak Raspberry üzerine kurulmuş ve uygulama içerisine “import requests” başlık bilgisi ile aktarılmıştır.

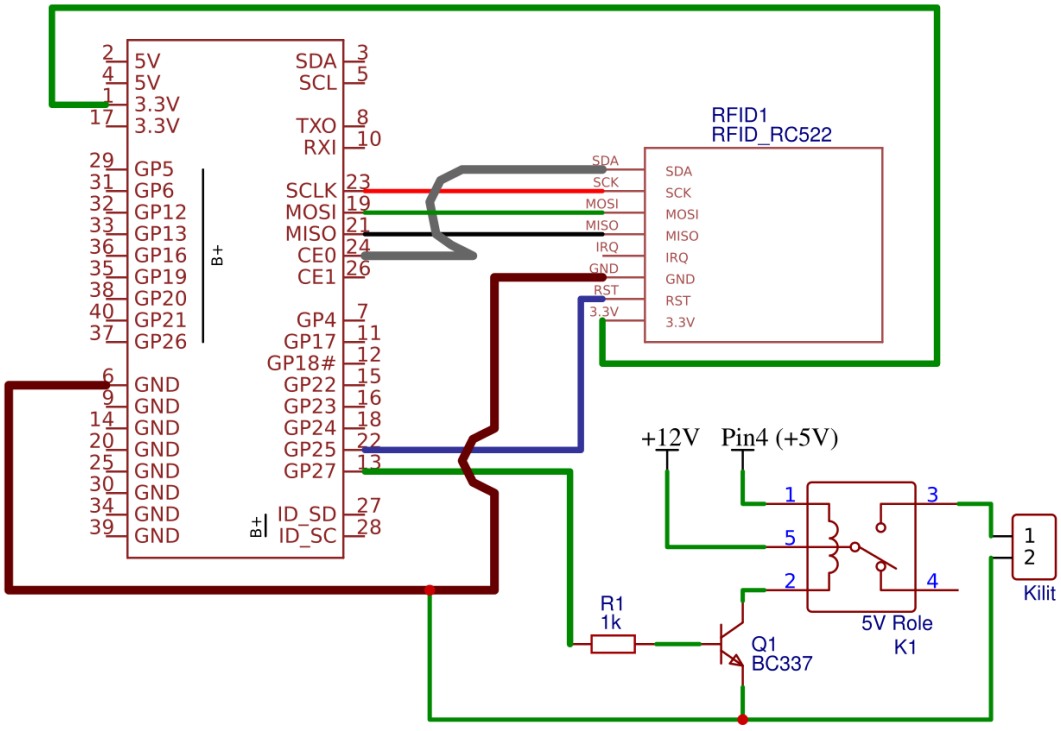
Gerekli kütüphane kurulumlarının ardından kullanılacak modül bağlantıları sağlanmıştır. PiCamera modülü CSI konnektörü ile Raspberry Pi’ye bağlanmıştır. Şekil 6.9’da kamera pin yapıları gösterilmiştir. Gt511c3 TTL parmak izi okuyucu ise USB-TTL dönüştürücü kullanılarak Raspberry USB portuna bağlanmıştır. Parmak izi okuyucu ve USB-TTL kablosu pin yapıları ve bağlantıları Şekil 6.10’da gösterilmiştir. RFID modül, röle ve röle sürücü devresinin tak çalıştır olarak kolay şekilde kullanılabilmesi için Raspberry GPIO pinleri ile uyumlu HAT tasarlanmış ve bağlantı şeması Şekil 6.11’de gösterilmiştir.



Şekil 6.9. PiCamera pin yapısı ve Raspberry Pi bağlantısı.

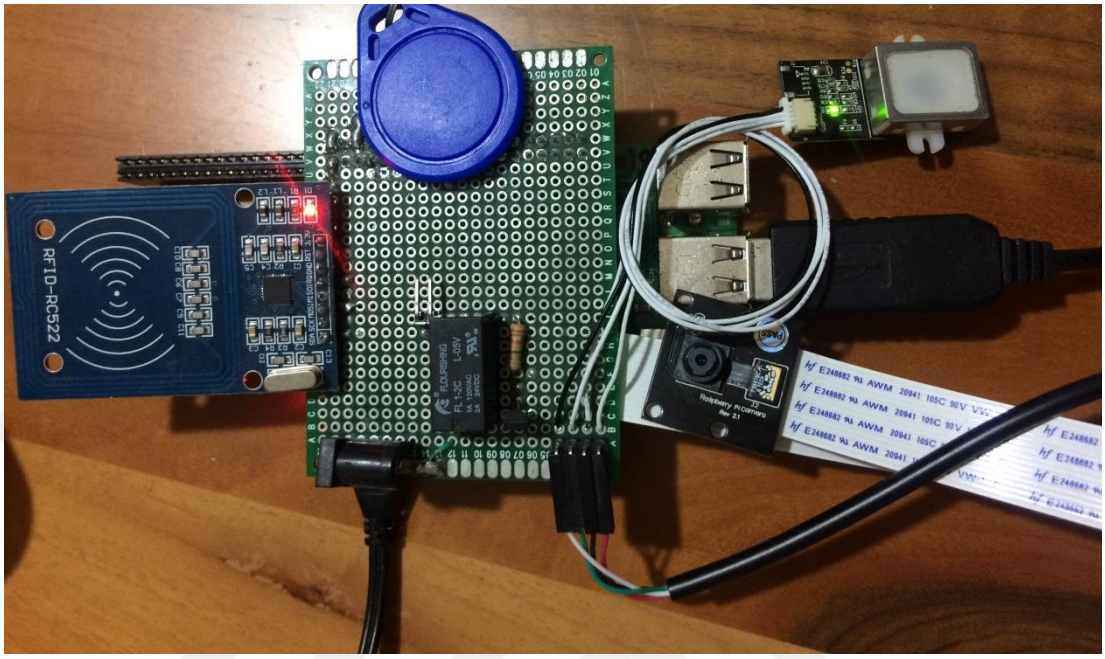


Şekil 6.10. Gt511c3 pin yapısı ve Raspberry Pi bağlantısı.



Şekil 6.11. RFID, röle sürücü Raspberry Pi HAT.

Kamera, parmak izi okuyucu ve HAT montajı yapıldıktan sonra elde edilen devre Şekil 6.12'de gösterilmiştir.



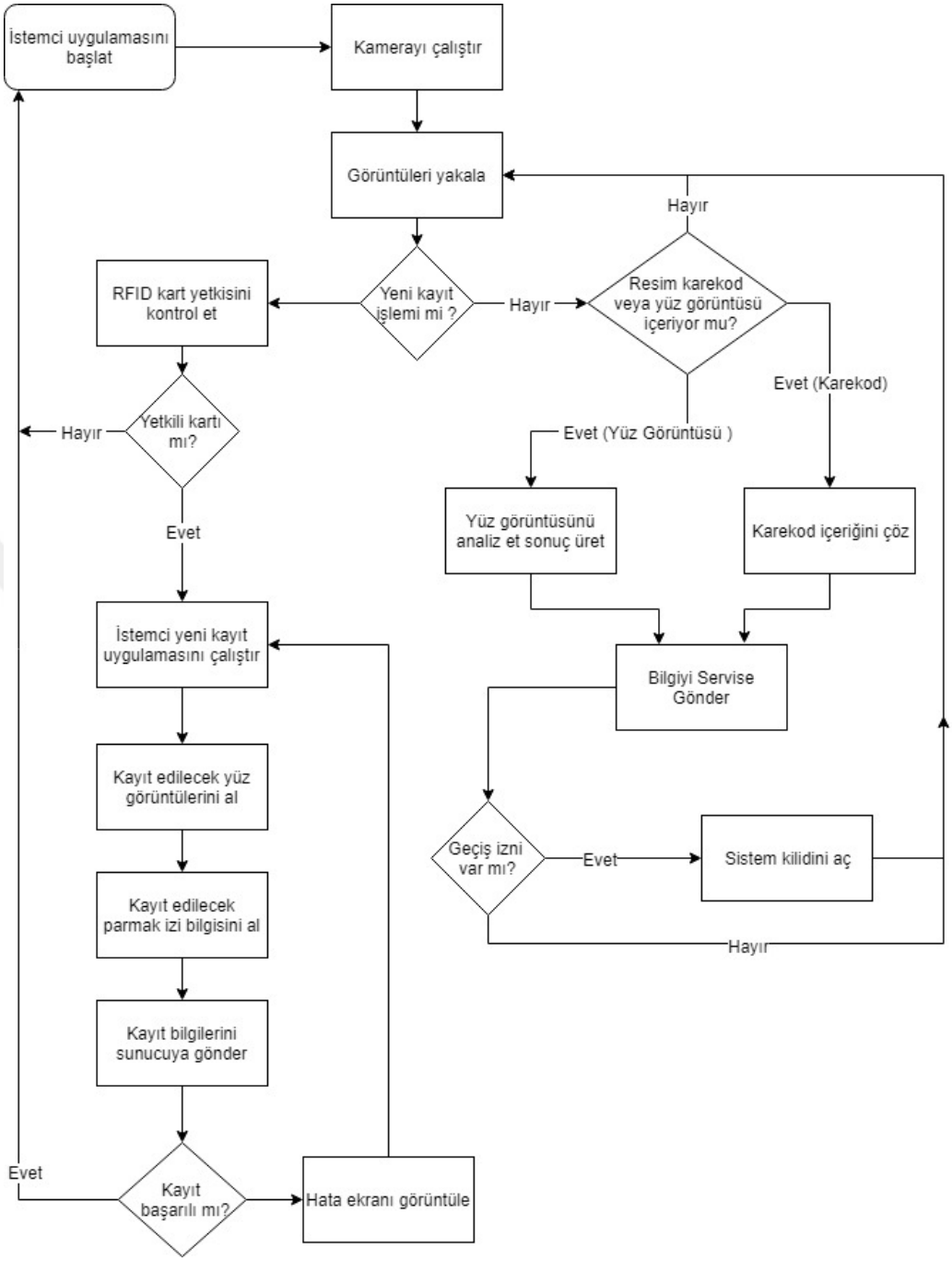
Şekil 6.12. Geçiş sistemi devre kartı.

Devre kart tasarımı tamamlandıktan ve Raspbian işletim sistemi üzerinde gerekli modüller kurulduktan sonra Python kodları tarafından yürütülecek sistem akış şeması oluşturulmuştur. Şekil 6.13'te verilen akış şemasında görülebileceği üzere uygulama açılış ekranında kamerayı tetiklemekte ve sürekli resim karelerini (frame) inceleyerek yüz resmi veya karekod olup olmadığını kontrol etmektedir. Yüz görüntüsü veya karekod görüntüsü algılandığında ilgili kütüphaneler yardımıyla yüz resminin hangi personele ait olduğu veya ilgili karekodun içerik bilgisi web sunucusuna gönderilmekte ve sunucu taraflı kontol işlemleri gerçekleştirilerek geriye uygunluk sonucu döndürülmektedir. Resimden elde edilen bilgi karekod içerik bilgisi ise Çizelge 6.1 ile verilen "ZiyaretciGirisKontrol" servis uç noktasına içerik bilgisi ile sorgu yapılarak uygunluk sonucu beklenir. Sunucu tarafından dönen sonucun uygun olması durumunda röle sürücü devresi enerjilendirilerek kilit açma işlemi gerçekleştirilmektedir. Resimden elde edilen bilgi personele ait benzersiz kimlik bilgisi ise Çizelge 6.1 ile verilen "PersonelGecisKontrol" servis uç noktasına personel kimlik bilgisi ile sorgu yapılarak uygunluk sonucu beklenir; dönen sonucun uygun olması durumunda istemci uygulaması personele ait parmak izi bilgisini

cihaza aktarır ve personelin parmak izini doğrulama işlemini gerçekleştirmesini bekler. Parmak izi bilgisi yüzü tanınan personel ile aynı olması durumunda yine kilit mekanizması enerjilendirilerek kilit açma işlemi sağlanır.

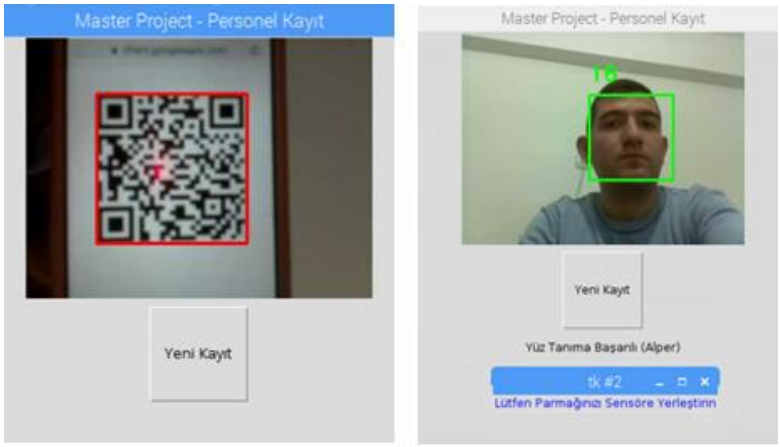
Akış şeması işleyişi için kodlarma 4 dosya olarak tasarlanmış ve kodlanmıştır. Ayrıca dosya yolları, servis adresi bağlantı bilgisi, yüz algılama metod türü ve yüz tanıma eğitimi sonucu oluşturulacak sınıflandırma dosyasına kolay erişim ve değiştirme imkanı sağlamak için "xml" formatında konfigürasyon dosyası hazırlanmıştır. Python ile kodlanan 4 dosya açıklamalarıyla birlikte aşağıda verilmiştir.

1. MakeRequest.py : Webservis ile bağlantı kurabilecek, xml konfigürasyon dosyasından veri okuyabilecek ve belirlenen klasördeki resim datalarını alabilecek hazır metodları bulunduran genel amaçlı modüldür.
2. FPKayitEkran.py :Parmak izi modül kontrolünü sağlayan ve röle sürücü devresini kontrol eden form ekranıdır. Kontrol ve doğrulama işlemleri için MakeRequest.py dosyasını kullanır.
3. FaceRecognition.py : Kameradan alınan görüntülerdeki yüz tespiti, tanıması ve karekod çözme işlemlerini gerçekleştiren form ekranıdır. Kontrol ve doğrulama işlemleri için MakeRequest.py modülünü, parmak izi işlemleri için FPKayitEkran.py dosyasını kullanır.
4. PersonelKayitEkran : Yeni personellerin oluşturulması ve yeni personellere ait yüz resimlerinin eğitilmesi işlemlerini gerçekleştiren form ekranıdır. Kontrol ve eşleştirme işlemleri için MakeRequest.py modülünü, parmak izi işlemleri için FPKayitEkran.py dosyasını kullanır.



Şekil 6.13. İstemci uygulaması çalışma akış şeması.

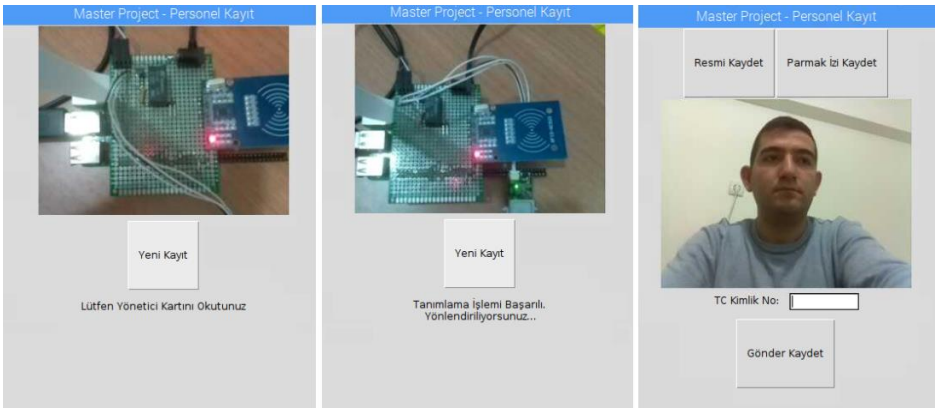
Şekil 6.14'de karekod çözme, yüz tanıma ve parmak izi doğrulama form ekranlarına ait ekran görüntüleri gösterilmiştir.



a b

Şekil 6.14. a) Karekod tanıma, b) yüz ve parmak izi tanıma form ekran görüntüsü.

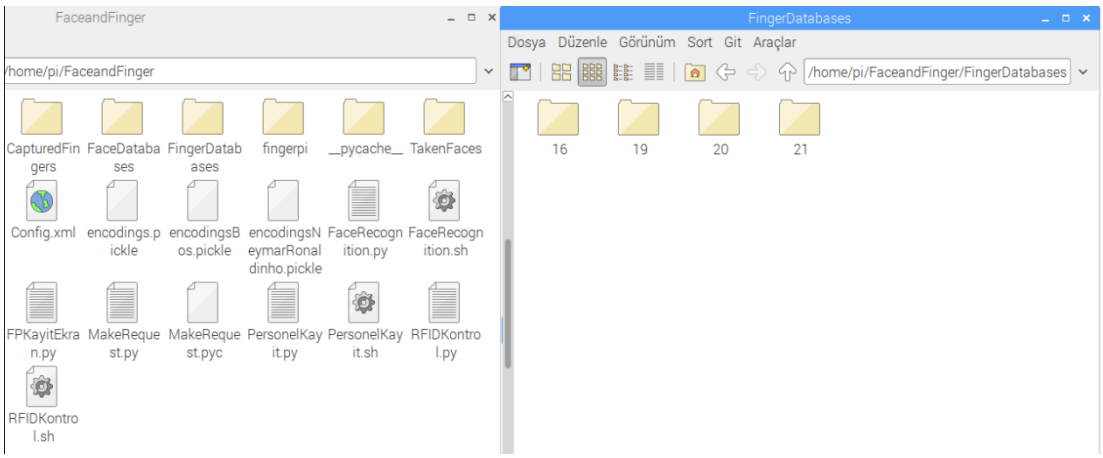
Sistemin personelleri tanıyabilmesi için yüz resimlerinin tanıtılması, parmak izi bilgilerinin tanımlanması ve veri tabanında belirlenen personellerle eşleştirilmesi gerekmektedir. Bu nedenle "FPKayitEkran.py" isminde personel kayıt formu oluşturulmuştur. Personel kayıt formunun açılması için Şekil 6.14'te gösterilen form üzerindeki yeni kayıt butonuna basılır. Butona basıldığında yine sunucu tarafında benzersiz kimlik bilgisi bir personel ile eşleştirilen RFID etiketin okutulması beklenir. Okutulan RFID etiket bilgisi (kart numarası) Çizelge 6.1 ile verilen "YetkiKontrol" servis uç noktasına gönderilerek etikete ait yetki tanımlamasının olup olmadığı kontrol edilir ve sonuç bilgisi döndürülür. Kartın yetkili kartı olması durumunda yeni kayıt formu açılır. Şekil 6.15'te yeni kayıt işleminin işleyişine ait uygulama ekran görüntüleri gösterilmiştir.



a b c

Şekil 6.15. a) Yönetici kartının beklenmesi, b) yetkinin doğrulanması, c) yeni kayıt ekranı ekran görüntüleri.

Açılan yeni kayıt ekranında kaydedilecek personele ait görüntülerin alınabilmesi için resim görüntüleme alanı, anlık görüntüleri yakalamak için resim kaydet ve parmak izi şemasının alınabilmesi için parmak izi kaydet butonları bulunmaktadır. Resim kaydet butonuna basıldığında anlık görüntü Şekil 6.16'da gösterilen dosya sisteminde "TakenFaces" klasörü altına aktarılır. Aynı şekilde parmak izi kaydet butonuna basıldığında parmak izi sensörüne bağlantı yapılarak yeni kayıt için bilgi dokümanında [26] belirtilen modül fonksiyonları çağırılır ve kayıt sonrası elde edilen parmak izi şablon bilgisi dosya sistemi üzerinde "CapturedFingers" klasörü altına aktarılır. Sunucu tarafında tanımlanan ve eşleştirilecek personel TC kimlik numarası yazılarak gönder kaydet butonuna basılır. Bu işlemle birlikte TC kimlik numarası, parmak izi şablon bilgisi ve personele ait çekilen görüntüler Çizelge 6.1 ile verilen "PersonelKayit" servis uç noktasına gönderilerek eşleştirme sonucu beklenir. Eşleştirme işleminin başarılı olması durumunda sunucudan dönen personel eşsiz kimlik bilgisi kullanılarak, kaydedilen yüz görüntüleri "FaceDatabases"; parmak izi şablonu ise "FingerDatabases" klasörüne kaydedilir. Kaydedilen yüz görüntüleri üzerinden CNN öğrenme algoritması çalıştırılarak yapay sinir ağı eğitimi gerçekleştirilir ve tekrar tanıma işlemlerinde kullanılmak üzere kaydedilir. Tüm işlemler başarılı şekilde tamamlandığında geçiş form uygulaması tekrardan açılarak sistem başlangıç konumuna getirilir.



Şekil 6.16. Geçiş kontrol sistemi Raspberry Pi dosya ve klasör yapısı.

BÖLÜM 7

SONUÇLAR

Bu çalışmada tasarlanan geçiş kontrol sistemi (yüz tanıma hariç) hali hazırda kurulu olan bir turnike sistemine başarı ile entegre edilmiş ve başarılı şekilde çalıştığı gözlemlenmiştir.

Sistem IoT teknolojisi ve web servis mimarisi sayesinde yerel veya bulut sunucular üzerinde bulunan sistemler ile de entegre olabilmektedir. Bu tez çalışmasında bulut sunucu kullanılmıştır. Parmak izi tanıma için sensör özellikleri kullanıldığından ve cihazın hata oranı çok düşük olduğundan hiçbir hatalı tanıma gerçekleşmemiştir. Yüz tanıma özelliği kullanılmadığında, 200 adet parmak izi sınırı olan cihaz daha yüksek kapasiteli tanıma gerektiren noktalarda herhangi bir yazılımsal değişikliğe ihtiyaç duyulmaksızın gt511c5 modeli ile değiştirilerek 2000 parmak izi tanıma kapasitesine artırılabilmektedir.

Yüz tanıma işlevselliği için alınan kamera modülü yüksek çözünürlüklü çekimleri desteklemesine rağmen Raspberry üzerinde bulunan yetersiz işlemci ve GPU sebebiyle düşük çözünürlük değerleri ile yakın mesafe tanımalarda kullanılmıştır. Yüz tanımanın daha iyi sonuçlar vermesi için ard arda algılanan 3 adet yüz resminin aynı kişiye ait olduğunun tespit edilmesi sonrasında geçiş kontrol işlemlerinin başlatılması sağlanmıştır.

Geçiş kontrol sisteminin hali hazırda varolan turnike sistemi ve harici bir şirket yönetim uygulamasına entegrasyonu sayesinde personel eksik çalışmalarının, geç işe girişlerin ve erken çıkışlarının tespiti sağlandığı gibi fazla mesailerin ve maaş hesaplanmasının da otomatize edilmesi sağlanabilecektir. Ayrıca özellikle Ar-Ge merkezleri için bakanlıkların sağlamış olduğu teşviklerden faydalanan personellerin çalışma sürelerinin de belirli periyotlarla ve istenilen formatlarda hızlı bir şekilde

elde edilerek gönderilmesi de sağlanabilecektir. Kullanılan karekod teknoloji ile ziyaretçi kontrolü sayesinde geçiş noktalarında hız ve güvenlik sağlandığı gibi kimlik vb. gibi kişisel kartların geçiş noktlarında bırakılmasının da önüne geçilerek KVKK uygunluğu da sağlanmıştır. Karekod ile geçiş yapan kişilerin de sistemde kayıt altına alınması sayesinde hangi ziyaretçinin ne zaman ve ne amaçla ziyaret gerçekleştirdiği bilgilerine kolay erişim sağlanarak kurumsal hafızanın güçlenmesine de fayda sağlanmıştır. Ayrıca gelen ziyaretçilerin de yüz görüntülerinin kamera üzerinden alınması, hatta alınan yüz görüntülerinin de eğitime tabi tutulması ile ziyaretçilerin de yüz tanıma sistemine entegre olması sağlanabilir.

KAYNAKLAR

1. Özeyranlı, E. B., "Biyometrik Tanıma Sistemleri Sunumu", ***Pamukkale Üniversitesi***

2. İnternet: **http://ahmetkakici.github.io/genel/biyometrik-tanima-sistemleri/,** (2004).

3. Edgington, B., "Introducing Hitachi's Finger Vein Technology", ***Hitachi Europe Limited***, (2007).

4. İnternet: **http://www.softschools.com/timelines/history_of_fingerprinting_timeline/287/**

5. İnternet: **http://www.sloughhistoryonline.org.uk/ixbin/hixclient.exe?%3Dtheme_record_id=sl-sl-williamjamesherschel&_IXFIRST_=1&_IXMAXHITS_=1&a=query&f=generic_theme.htm&p=slough**

6. İnternet: **http://yuztanima.info/yuz-tanima-teknolojisinin-tarihi-ii-1970-1991/**

7. Daugman, J. G., "Biometric personal identification system based on iris analysis." ***U.S.Patent No. 5,291,560***, (1994).

8. Shimizu, K., "Optical Trans-Body Imaging: Feasibility of Optical CT and Functional Imaging of Living Body", ***Jpn. J. Med. Philos***. (1992).

9. Shimizu, K. and Yamamoto, K., "Imaging of physiological functions by laser transillumination. In Advances in Optical Imaging and Photon Migration; Alfano, R.R., Fujimoto, J.G., Eds.; OSA: Orlando, FL, USA, (1996).

10. Im, S. K. and Park, H. M. and Kim, Y. W. and Han, S. C. and Kim, S. W. and Kang, C.H., "Biometric Identification System by Extracting Hand Vein Patterns," ***Journal of the Korean Physical Society***, Vol. 38, No. 3, (2001).

11. Kirby, M. and Sirovich, L., "Application of the karhunen-loeve procedure for the characterization of human faces", ***IEEE Pattern Analysis and Machine Intelligence***, vol. 12, no. 1, pp. 103-108, (1990).

12. Xiongwu, X. and Lawrence, O., "Innovations in -ngerprint capture devices", ***Pattern Recognition 36*** (2003)

13. İnternet: **https://sourceafis.machinezoo.com/transparency/pairing**

14. Maltoni, D. and Maio, D. and Jain, A. and Prabhakar, S., "Handbook of Fingerprint Recognition", Springer-Verlag, New York, 2003.

15. Viola, P. and Jones, M., "Rapid Object Detection Using a Boosted Cascade of Simple Features", ***Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)***, Vol. 1, 2001, pp. 511-518.

16. İnternet: **https://medium.com/coinmonks/from-the-rand-tablet-to-differentiating-identical-twins-aa4ba6031bb0**, (2018)

17. İnternet: **https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html**

18. İnternet: **https://hub.packtpub.com/implementing-face-detection-using-haar-cascades-adaboost-algorithm/**

19. Dalal N. and Triggs B., "Histograms of oriented gradients for human detection", ***2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)***, San Diego, CA, USA, 2005, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177

20. İnternet: **https://www.learnopencv.com/histogram-of-oriented-gradients/**

21. İnternet: **https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17**

22. İnternet: **http://deeplizard.com/learn/video/ZjM_XQa5s6s**

23. Maraşlı, F. ve Çıbuk, M., **"**RFID Teknolojisi ve Kullanım Alanları**", BEÜ Fen Bilimleri Dergisi BEU Journal of Science 4(2), 249-275**, (2015)

24. İnternet: **https://www.elektrikport.com/universite/karekod-nedir/12229**

25. İnternet: **https://www.explainthatstuff.com/how-data-matrix-codes-work.html**

26. İnternet: **https://cdn.sparkfun.com/datasheets/Sensors/Biometric/GT-511C3_datasheet_V1%201_20130411[4].pdf**

27. İnternet: **https://opencv.org/**

28. İnternet: **https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/**

29. İnternet: **https://github.com/zafartahirov/fingerpi**

30. İnternet: **https://pypi.org/project/pi-rc522/**

31. İnternet: **https://pypi.org/project/pyzbar/**

32. İnternet: **https://pypi.org/project/face_recognition/**

33. İnternet: **http://docs.python-requests.org/en/master/**

34. Adler F. H., “Physiology of the Eye”, Chapter IV, 143 (1953).

EK AÇIKLAMALAR A.

RASPBERRY PI PYTHON UYGULAMA KODLARI

```
#MakeRequest.py
#Yardımcı Metodlar için Geliştirilen Modül

#Kütüphanelerin Eklenmesi
import time
import json
import base64
import requests
import RPi.GPIO as GPIO
from xml.dom import minidom
import os
#Kütüphanelerin Eklenmesi

#Servis İstekleri İçin Hazırlanan Metodlar
def MakePost(queryString,requestData):
    headers = {'Content-Type': 'application/json;charset=utf-8','dataType':
"json"}
    servisadres=XmlVeriAl("ServisAdres")
    print("Post Begin")
    r= requests.post(servisadres+queryString,
data=json.dumps(requestData),headers=headers)
    print(r)
    isOk=False
    if r.status_code==200:
        isOk=True
    return isOk,json.loads(json.loads(r.text))

def MakeGet(queryString):
    isOk=False
    ret="Sunucu Kontrolü Hata ile Karşılaştı"
    try:
        servisadres=XmlVeriAl("ServisAdres")
        print(servisadres+queryString)
        r= requests.get(servisadres+queryString)
        if r.status_code==200:
            isOk=True
            ret=json.loads(json.loads(r.text))
    except Exception as e:
        printLog(e)
    return isOk,ret
#Servis İstekleri İçin Hazırlanan Metodlar

#XML Parse Eden Fonksiyonlar
def XmlVeriAl(etiketadi):
    try:
        xmldoc = minidom.parse('/home/pi/FaceandFinger/Config.xml')
        item = xmldoc.getElementsByTagName(etiketadi)[0]
        printLog(etiketadi+" : "+EtiketIcerikGetir(item.childNodes))
        return EtiketIcerikGetir(item.childNodes)
    except Exception as e:
        printLog(e)
        printLog("Config Dosyası "+etiketadi+" İçeriği Okunurken Hata
Oluştu.")
        return ""

def EtiketIcerikGetir(nodelist):
    rc = []
    for node in nodelist:
        if node.nodeType == node.TEXT_NODE:
```

```
            rc.append(node.data)
    return ''.join(rc)
#XML Parse Eden Fonksiyonlar

#Dizinde Belirtilen Dosyayı Base64 Formatında Getiren Metoddur
def ReadBS64FileFromFolder(folderName,extension):
    file_content =[]
    for file in os.listdir(folderName):
        if file.endswith(extension):
            try:
                with open( os.path.join( folderName, file ) ,"rb") as fd:

file_content.append(base64.b64encode(fd.read()).decode("utf-8"))
                print("AlınanDosya:"+folderName+file)
            except Exception as e:
                printLog(e)
    return file_content
#Dizinde Belirtilen Dosyayı Base64 Formatında Getiren Metoddur

def printLog(logdata):
    print(logdata)
#MakeRequest.py Dosya Sonu

#FPKayitEkran.py
#Parmak İzi Modülü Haberleşme Kodları


#Kütüphanelerin Eklenmesi
import fingerpi as fp
import time
import pickle
import numpy as np
from PIL import Image
import json
import base64
import cStringIO
import requests
import json
import Tkinter as tki
import os
import argparse
import RPi.GPIO as GPIO
import MakeRequest as Helper
#Kütüphanelerin Eklenmesi

#ÇIKIŞ GPIO Pin Ayarlaması
rolepin = 13
GPIO.setmode(GPIO.BOARD)
GPIO.setup(rolepin, GPIO.OUT)
#ÇIKIŞ GPIO Pin Ayarlaması

#Parmak İzi Modülün Entegre Edilmesi ve Değişkenlerin Belirlenmesi
f = fp.FingerPi()
fpOutputPath="/home/pi/FaceandFinger/CapturedFingers"
root = tki.Tk()
#Parmak İzi Modülün Entegre Edilmesi ve Değişkenlerin Belirlenmesi

#Bilgilendirme Etiketinin Değiştirilmesi
def changeText(innerText):
    root.update()
    text = tki.StringVar()
```

```
    text.set('')
    lab = tki.Label(root, textvariable=text, fg='blue')
    lab.pack()
    text.set(innerText)
    root.update()
    lab.pack_forget()
#Bilgilendirme Etiketinin Değiştirilmesi

#Parmak İzi Var Mı Yok Mu Kontrolü
def IsPressed():
    time.sleep(1)
    changeText("Parmağınızı Kaldırın")
    while True:
        responseispress=f.IsPressFinger()
        if str(responseispress[0]["Parameter"])!="0":
            break
    changeText("Parmağınızı Tekrar Koyun")
    while True:
        responseispress=f.IsPressFinger()
        if str(responseispress[0]["Parameter"])=="0":
            break
    time.sleep(1)
    changeText("İşleniyor...")
    return True
#Parmak İzi Var Mı Yok Mu Kontrolü


#Parmak İzi Cihaza Yeni İz Kaydı Metodu
def YeniKayit(eklenecekid):
    response = f.EnrollStart(eklenecekid)
    if response[0]["ACK"]:
        changeText("Parmağınızı Sensöre Yerleştirin")
        f.CmosLed(True)
        while True:
            responsecp1=f.CaptureFinger(True)
            if responsecp1[0]["ACK"]:
                responsenroll1= f.Enroll1()
                changeText("enroll1:"+str(responsenroll1[0]["ACK"]))
                break
        IsPressed()
        while True:
            f.CmosLed(True)
            responsecp1=f.CaptureFinger(True)
            if responsecp1[0]["ACK"]:
                responsenroll2= f.Enroll2()
                changeText("enroll2:"+str(responsenroll2[0]["ACK"]))
                break
        IsPressed()
        while True:
            f.CmosLed(True)
            responsecp2=f.CaptureFinger(True)
            if responsecp2[0]["ACK"]:
                responsenroll3= f.Enroll3()
                #print(str(responsenroll3))
                if responsenroll3[0]["ACK"]:
                    changeText(str(eklenecekid)+" ID Değeri İle Eklendi")
                    f.CmosLed(False)
                    return True
                else:
                    responsecapture= f.CaptureFinger()
                    if responsecapture[0]["ACK"]:
```

```
                        responseid=f.Identify()
                        changeText("Hata Bu Parmak İzi
"+str(responseid[0]["Parameter"])+" IDsi İle Kayıtlı")

            break
        f.CmosLed(False)
    else:
        changeText(str(eklenecekid)+" ID Değerine Sahip Kayıt Var")
    return False
#Parmak İzi Cihaza Yeni İz Kaydı Metodu

#Parmak İzi Doğrulama Metodu
def PIDogrula():
    sensorid=-1
    f.CmosLed(True)
    changeText('Lütfen Parmağınızı Sensöre Yerleştirin')
    print("Lütfen Parmağınızı Sensöre Yerleştirin")
    dogrulamaAdet=0
    while True:
        responsecapture= f.CaptureFinger()
        if responsecapture[0]["ACK"]:
            responseid=f.Identify()
            if responseid[0]["ACK"]:
                sensorid=int(responseid[0]["Parameter"])
                break
            else:
                dogrulamaAdet=dogrulamaAdet+1
                print("Hata = "+str(responseid[0]["Parameter"]))
                print(dogrulamaAdet)
                if (dogrulamaAdet>2):
                    break
    f.CmosLed(False)
    return sensorid
#Parmak İzi Doğrulama Metodu

#Parmak İzi Cihaza Şablon Aktarma Metodu
def SetTemplate(FPID,Template):
    responsesend=f.SetTemplate(FPID,Template)
    print("Parmak İzi Aktarım İşlemi Sonucu:"+str(responsesend))
    donecekdeger=False
    if responsesend[0]["ACK"]:
        donecekdeger=True
    return donecekdeger
#Parmak İzi Cihaza Şablon Aktarma Metodu

#Cihazdan Parmak İzlerinin Silinmesi
def TumunuSil():
    responsedelete=f.DeleteAll()
    print("Silme İşlemi Sonucu:"+str(responsedelete))
    donecekdeger=False
    if responsedelete[0]["ACK"] or
responsedelete[0]["Parameter"]=="NACK_DB_IS_EMPTY":
        donecekdeger=True
    return donecekdeger
#Cihazdan Parmak İzlerinin Silinmesi

#Parmak İzi Cihazdan Şablon Bilgisinin Klasöre Alınması
def KayitIslemYap(IslemOk,FPID):
    changeText('Baglantı Sağlanıyor...')
    print('Baglantı Sağlanıyor...')
    f.Open(extra_info = True, check_baudrate = True)
```

```
    changeText('Baud Hızı Ayarlanıyor...')
    print('Baud Hızı Ayarlanıyor...')
    f.ChangeBaudrate(115200)
    f.CmosLed(False)
    if TumunuSil():
        if IslemOk:
            if YeniKayit(0):
                response=GetFPTemplate(0)
                print(response)
                if response[0]:
                    with
open("/home/pi/FaceandFinger/CapturedFingers/0FP.bin","w+b") as wrt:
                        wrt.seek(0)
                        wrt.write(response[1]['Data'])
                        changeText("Parmak İzi Aktarılıyor")
                    time.sleep(2)
                    sys.exit(0)
        else:

fpFilePath="/home/pi/FaceandFinger/FingerDatabases/"+str(FPID)+"/0FP.bin"
            print(fpFilePath)
            file_byte=GetFileByte(fpFilePath)
            if SetTemplate(0,file_byte):
                if PIDogrula()!=-1:
                    RoleIslem(FPID)
                else:
                    changeText("Parmak İzi Doğrulama İşlemi Başarısız Oldu")
                    time.sleep(2)
            else:
                changeText("Parmak İzi Aktarım İşlemi Başarısız Oldu")
                time.sleep(2)
            sys.exit(0)
#Parmak İzi Cihazdan Şablon Bilgisinin Klasöre Alınması

#Personel Geçiş Kontrolü Ve Röle Pininin Aktif Edilmesi
def RoleIslem(PersonelID):
    url="/PersonelGecisKontrol?PersonelID="+str(PersonelID)
    isOk,response=Helper.MakeGet(url)
    if isOk:
        if str(response["GecisIzin"])=="True":
            GPIO.output(rolepin, True)
            changeText("Geçiş İzni Onaylandı")
            time.sleep(2)
            GPIO.output(rolepin,False)
        else:
            changeText("Geçiş İzniniz Bulunmamaktadır")
            time.sleep(2)
#Personel Geçiş Kontrolü Ve Röle Pininin Aktif Edilmesi

#Cihazdan Parmak İzi Şablon Bilgisinin Alınması
def GetFPTemplate(TempID):
    responsetemplate=f.GetTemplate(TempID)
    return [responsetemplate[0]["ACK"],responsetemplate[1]]
#Cihazdan Parmak İzi Şablon Bilgisinin Alınması


def GetFileByte(filename):
    return open(filename, "rb").read()

#Klasör İçerisindeki Dosyaların Temizlenmesi
def RemoveFiles(folder):
```

```
    print(folder+" Klasörü Temizleniyor...")
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
            elif os.path.isdir(file_path): shutil.rmtree(file_path)
        except Exception as e:
            print(e)
#Klasör İçerisindeki Dosyaların Temizlenmesi


#Form Parametrelerinin Belirlenmesi Ve Formun Başlatılması
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--fpid", required=True,
       help="Yapılacak İşlemi Belirtmelisiniz")
args = vars(ap.parse_args())
fpid= int(args["fpid"])
KayitIslem=True
if fpid!=-1:
    KayitIslem=False
fpOutputPath="/home/pi/FaceandFinger/CapturedFingers"
RemoveFiles(fpOutputPath)
root.after(2000,KayitIslemYap(KayitIslem,fpid))
root.mainloop()
#Form Parametrelerinin Belirlenmesi Ve Formun Başlatılması
#FPKayitEkran.py Dosya Sonu


#PersonelKayit.py
#Yeni Personel Kayıt Formu Kodları


#Kütüphanelerin Eklenmesi
from __future__ import print_function
from PIL import Image
from PIL import ImageTk
import tkinter as tki
import threading
import datetime
import imutils
from imutils import paths
import cv2
import os
import pickle
from imutils.video import VideoStream
import time
import shutil
import MakeRequest as Helper
import face_recognition
import subprocess
#Kütüphanelerin Eklenmesi

#Form Objesinin Oluşturulması
class PersonelKayitApp:
    #Form Objesinin Başlatılması
    def __init__(self, vs,faceOutPath,fpOutPath):
        #Form Temel Değişkenlerinin ve  Elementlerinin Oluşturulması
        self.isPersonCaptured=True
        self.vs = vs
        self.frame = None
```

```
        self.thread = None
        self.stopEvent = None
        self.faceOutputPath=faceOutPath
        self.fpOutputPath=fpOutPath
        self.root = tki.Tk()
        self.panel = None
        self.root.attributes('-zoomed', True)
        self.ButtonContainer=tki.Frame(self.root)
        self.ButtonContainer.pack(side="top", fill="x")
        self.CameraContainer=tki.Frame(self.root)
        self.CameraContainer.pack(side="top", fill="x")
        self.TxtContainer=tki.Frame(self.root)
        self.TxtContainer.pack(side="top", fill="x")
        self.FaceSaveButton = tki.Button(self.ButtonContainer, text="Resmi
Kaydet",command=self.takePicture,height = 5)
        self.FaceSaveButton.grid(column=1, row = 1,sticky="N")
        self.FPSaveButton = tki.Button(self.ButtonContainer, text="Parmak İzi
Kaydet",command=self.getFP,height = 5)
        self.FPSaveButton.grid(column=2, row = 1,sticky="N")
        self.lblTcKimlikNo=tki.Label(self.TxtContainer,text="TC Kimlik No:")
        self.lblTcKimlikNo.grid(column=1,row=1,sticky="N",padx=5,pady=5)
        self.txtTcKimlikNo = tki.Text(self.TxtContainer,height=1,width=11)
        self.txtTcKimlikNo.grid(column=2,row=1,sticky="N",padx=5,pady=5)
        self.btnSendServer = tki.Button(self.TxtContainer, text="Gönder
Kaydet",command=self.SendtoServer,height = 5)
        self.btnSendServer.grid(column=1, row =
2,sticky="N",columnspan=2,padx=5,pady=5)
        self.ButtonContainer.grid_columnconfigure(0, weight=1)
        self.ButtonContainer.grid_columnconfigure(3, weight=1)
        self.TxtContainer.grid_columnconfigure(0, weight=1)
        self.TxtContainer.grid_columnconfigure(3, weight=1)
        self.CameraContainer.grid_columnconfigure(0, weight=1)
        self.CameraContainer.grid_columnconfigure(2, weight=1)
        #Form Temel Değişkenlerinin ve  Elementlerinin Oluşturulması

        #Video görüntülemenin yeni thread ile başlatılması
        self.stopEvent = threading.Event()
        self.thread = threading.Thread(target=self.videoLoop, args=())
        self.thread.start()
        #Video görüntülemenin yeni thread ile başlatılması

        self.root.wm_title("Master Project - Personel Kayıt")
        self.root.wm_protocol("WM_DELETE_WINDOW", self.onClose)
        #Form Temel Değişkenlerinin ve  Elementlerinin Oluşturulması

    #Video Döngü Metodu Formun Ana Metodudur
    def videoLoop(self):
        print("girdi")
        try:
            while not self.stopEvent.is_set():
                self.frame = self.vs.read()
                self.frame = imutils.resize(self.frame, width=300)
                image = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
                image = Image.fromarray(image)
                image = ImageTk.PhotoImage(image)
                if self.panel is None:
                    self.panel = tki.Label(self.CameraContainer,image=image)
                    self.panel.image = image
                    self.panel.grid(column=1,row=2,sticky="N")
                else:
                    self.panel.configure(image=image)
```

```
                    self.panel.image = image
            except RuntimeError as e:
                print("[INFO] caught a RuntimeError")
        #Video Döngü Metodu Formun Ana Metodudur


        def onClose(self):
            print("[INFO] closing...")
            self.stopEvent.set()
            self.vs.stop()
            self.root.quit()

        #Parmak İzi Formunun Başlatılması
        def getFP(self):
            os.system("/home/pi/FaceandFinger/FPKayitEkran.py -i -1")
        #Parmak İzi Formunun Başlatılması

        #Kameradan Resim Görüntüsü Alınarak Kaydedilmesi
        def takePicture(self):
            ts = datetime.datetime.now()
            filename = "{}.jpg".format(ts.strftime("%Y-%m-%d_%H-%M-%S"))
            p = os.path.sep.join((self.faceOutputPath, filename))
            cv2.imwrite(p, self.frame.copy())
            print("Kaydedildi {}".format(filename))
        #Kameradan Resim Görüntüsü Alınarak Kaydedilmesi

        #Personel Bilgilerinin WebServise Gönderilmesi
        def SendtoServer(self):
            face_files=Helper.ReadBS64FileFromFolder(self.faceOutputPath,".jpg")
            fpfile=Helper.ReadBS64FileFromFolder(self.fpOutputPath,".bin")
            isOk,data=
Helper.MakePost("/PersonelKayit",{'bs64FaceImage':face_files,'bs64FPData':fpfi
le,'TCKN':self.txtTcKimlikNo.get("1.0",tki.END)})
            if isOk:
                print("İşlem Sonucu="+data["Message"])
                IslemKayitID=data["IslemID"]

faceDestinationPath="/home/pi/FaceandFinger/FaceDatabases/"+str(IslemKayitID)

fpDestinationPath="/home/pi/FaceandFinger/FingerDatabases/"+str(IslemKayitID)

self.MoveTemplates([faceDestinationPath,fpDestinationPath],[faceOutputPath,fpO
utputPath])
                if self.StartFaceEncoding():
                    self.stopEvent.set()
                    self.vs.stop()

child=subprocess.Popen("/home/pi/FaceandFinger/FaceRecognition.sh",stdout=subp
rocess.PIPE)
                    time.sleep(20)
                    self.root.destroy()
        #Personel Bilgilerinin WebServise Gönderilmesi

        #Parmak İzi Şablon Bilgisinin ve Resim Dosyalarının Kayıt Sonrası
Klasörüne Taşınması
        def MoveTemplates(self,DestinationPaths,SourcePaths):
            sira=0
            for path in SourcePaths:
                print(path)
                files = os.listdir(path)
                try:
```

```
                print(DestinationPaths[sira]+" Klasörü Oluşturuluyor")
                os.makedirs(DestinationPaths[sira]);
                print("Klasör Oluşturma İşlemi Başarılı")
            except:
                print("Klasör Oluşturulamadı. Bunun Sebebi Klasörün Zaten
Bulunması Olabilir.")
            for f in files:
                print(f+":"+path+" ==> "+DestinationPaths[sira])
                shutil.move(path+"/"+f, DestinationPaths[sira]+"/"+f)
            sira=sira+1
    #Parmak İzi Şablon Bilgisinin ve Resim Dosyalarının Kayıt Sonrası
Klasörüne Taşınması

    #Yüz Tanıma Eğitim Algoritmasının Çalıştırılması
    def StartFaceEncoding(self):
        imagePaths = list(paths.list_images(Helper.XmlVeriAl("DataSet")))
        knownEncodings = []
        knownNames = []
        for (i, imagePath) in enumerate(imagePaths):
            print("Resimler İşleniyor {}/{}".format(i + 1,len(imagePaths)))
            name = imagePath.split(os.path.sep)[-2]
            image = cv2.imread(imagePath)
            rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            boxes =
face_recognition.face_locations(rgb,model=Helper.XmlVeriAl("DetectionMethod"))
            encodings = face_recognition.face_encodings(rgb, boxes)
            for encoding in encodings:
                knownEncodings.append(encoding)
                knownNames.append(name)
            data = {"encodings": knownEncodings, "names": knownNames}
            f = open(Helper.XmlVeriAl("Encoding"), "wb")
            f.write(pickle.dumps(data))
            f.close()
        return True
    #Yüz Tanıma Eğitim Algoritmasının Çalıştırılması

#Belirlenen Klasörün Temizlenmesi
def RemoveFiles(folder):
    print(folder+" Klasörü Temizleniyor...")
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
            elif os.path.isdir(file_path): shutil.rmtree(file_path)
        except Exception as e:
            print(e)
#Belirlenen Klasörün Temizlenmesi


print("Kamera Hazırlanıyor...")
#Kamerayı Çalıştır
vs=VideoStream(usePiCamera=True).start()
time.sleep(2.0)
faceOutputPath="/home/pi/FaceandFinger/TakenFaces"
fpOutputPath="/home/pi/FaceandFinger/CapturedFingers"
#İşlem Klasörlerinin Temizlenmesi
RemoveFiles(faceOutputPath)
RemoveFiles(fpOutputPath)
#Kayıt Form Objesinin Ayarlanması
pba = PersonelKayitApp(vs,faceOutputPath,fpOutputPath)
```

```
#Form Döngüsünün Başlatılması
pba.root.mainloop()

#PersonelKayit.py Dosya Sonu


#FaceRecognition.py Dosya Başlangıcı
#Yüz Tanıma Ana Form Ekranı

#Kütüphanelerin Eklenmesi
from __future__ import print_function
from PIL import Image
from PIL import ImageTk
import tkinter as tki
import threading
import datetime
import imutils
from imutils import paths
import cv2
import os
import pickle
from imutils.video import VideoStream
import time
import shutil
import MakeRequest as Helper
import face_recognition
from pyzbar import pyzbar
import subprocess
from pirc522 import RFID
import subprocess
import json
import RPi.GPIO as GPIO
#Kütüphanelerin Eklenmesi

#Çıkış Pininin Ayarlanması
rolepin = 13
GPIO.setmode(GPIO.BOARD)
GPIO.setup(rolepin, GPIO.OUT)
#Çıkış Pininin Ayarlanması

#Form Objesinin Oluşturulması
class PersonelTanima:
    #Form Objesinin Başlatılması
    def __init__(self, vs,array):

        #Form Temel Değişkenlerinin ve  Elementlerinin Oluşturulması
        self.isPersonCaptured=True
        self.vs = vs
        self.array=array
        self.frame = None
        self.thread = None
        self.stopEvent = None
        self.data = pickle.loads(open(Helper.XmlVeriAl("Encoding"),
"rb").read())
        self.detectionMethod=Helper.XmlVeriAl("DetectionMethod")
        self.root = tki.Tk()
        self.root.attributes('-zoomed', True)
        self.panel = None
        self.rdr = RFID()
        self.util = self.rdr.util()
        self.util.debug = True
```

```
        self.infoText = tki.StringVar()
        self.infoText.set('')
        self.CameraContainer=tki.Frame(self.root)
        self.CameraContainer.pack(side="top", fill="x")
        self.TxtContainer=tki.Frame(self.root)
        self.TxtContainer.pack(side="top", fill="x")
        self.btnSendServer = tki.Button(self.TxtContainer, text="Yeni
Kayıt",command=self.PersonelKaydet,height = 5)
        self.btnSendServer.grid(column=1, row =
1,sticky="N",columnspan=2,padx=5,pady=5)
        self.lblInfo=tki.Label(self.TxtContainer,textvariable = self.infoText)
        self.lblInfo.grid(column=1, row =
2,sticky="N",columnspan=2,padx=5,pady=5)
        self.TxtContainer.grid_columnconfigure(0, weight=1)
        self.TxtContainer.grid_columnconfigure(3, weight=1)
        self.CameraContainer.grid_columnconfigure(0, weight=1)
        self.CameraContainer.grid_columnconfigure(2, weight=1)
        self.stopEvent = threading.Event()
        #Video görüntülemenin yeni thread ile başlatılması
        self.thread = threading.Thread(target=self.videoLoop, args=())
        self.thread.start()
        #Video görüntülemenin yeni thread ile başlatılması
        self.root.wm_title("Master Project - Personel Kayıt")
        self.root.wm_protocol("WM_DELETE_WINDOW", self.onClose)
        #Form Temel Değişkenlerinin ve  Elementlerinin Oluşturulması

    #Video Döngü Metodu Formun Ana Metodudur
    def videoLoop(self):
        try:
            while not self.stopEvent.is_set():
                self.frame = self.vs.read()
                self.frame = imutils.resize(self.frame, width=300)
                self.DecodeBarcode(self.frame)
                tanimasonuc= self.RecognizeFace(self.frame)
                image = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
                image = Image.fromarray(image)
                image = ImageTk.PhotoImage(image)
                if self.panel is None:
                    self.panel = tki.Label(self.CameraContainer,image=image)
                    self.panel.image = image
                    #self.panel.pack(side="top", fill="x")
                    self.panel.grid(column=1,row=2,sticky="N")
                else:
                    self.panel.configure(image=image)
                    self.panel.image = image
                if len(tanimasonuc)>0:
                    if str(tanimasonuc[0])!="Bilinmiyor":
                        if self.all_the_same():

url="/PersonelGecisKontrol?PersonelID="+tanimasonuc[0]
                            isOk,response=Helper.MakeGet(url)
                            print(response)
                            if isOk:
                                self.changeText("Yüz Tanıma Başarılı
("+str(response["PersonelAdi"])+")")
                                if str(response["GecisIzin"])=="True":

os.system("/home/pi/FaceandFinger/FPKayitEkran.py -i "+tanimasonuc[0])
                                else:
                                    time.sleep(2)
```

```
                                    self.changeText("Geçiş İzniniz
Bulunmamaktadır")
                    else:
                        self.array=[0,0,0]
            except RuntimeError as e:
                print("[INFO] caught a RuntimeError")
        #Video Döngü Metodu Formun Ana Metodudur

        #Dizi İçerik Kontrol Metodu
        def all_the_same(self):
            sonuc =len(self.array) < 1 or len(self.array)
==self.array.count(self.array[0])
            print(sonuc)
            return sonuc
        #Dizi İçerik Kontrol Metodu

        #Personel Kayıt Formunun Açılması
        def PersonelKaydet(self):
            self.changeText("Lütfen Yönetici Kartını Okutunuz")
            time.sleep(1)
            self.rdr.wait_for_tag(10)
            (error, data) = self.rdr.request()
            info="Ayrılan Süre Sona Erdi.\nYönlendiriliyor..."
            if not error:
                self.changeText("\nKart Algilandi!")
                (error, uid) = self.rdr.anticoll()
                if not error:
                    kart_uid =
str(uid[0])+str(uid[1])+str(uid[2])+str(uid[3])+str(uid[4])
                    self.changeText(kart_uid)
                    self.changeText("Kart Bilgisi Kontrol Ediliyor")
                    time.sleep(1)
                    url="/YetkiKontrol?KartNo="+kart_uid
                    isOk,response=Helper.MakeGet(url)
                    if isOk:
                        if str(response["isOk"])=="True":
                            self.changeText("Tanımlama İşlemi Başarılı.\n
Yönlendiriliyorsunuz...")
                            self.stopEvent.set()
                            self.vs.stop()

child=subprocess.Popen("/home/pi/FaceandFinger/PersonelKayit.sh",stdout=subpro
cess.PIPE)
                            time.sleep(20)
                            self.root.destroy()
                        else:
                            self.changeText("Karta Ait Yönetici Bulunamadı")
                            time.sleep(2)
                            self.changeText("")
                    else:
                        self.changeText("Kart Kontrol İşlemi Hata İle
Karşılaştı\nLütfen Daha Sonra Tekrar Deneyiniz")
                        time.sleep(2)
                        self.changeText("")
                else:
                    self.changeText("Kart Bilgileri Alınırken Hata Oluştu.\nLütfen
Daha Sonra Tekrar Deneyiniz")
                    time.sleep(2)
                    self.changeText("")
            else:
```

```
            self.changeText("Yönetici Kartı Belirlenen Süre İçerisinde
Algılanmadı")
    #Personel Kayıt Formunun Açılması

    #Tanıma Formu Açıklama Gösterimini Değiştiren Metoddur.
    def changeText(self,text):
        self.infoText.set(text)
        self.root.update_idletasks()
    #Tanıma Formu Açıklama Gösterimini Değiştiren Metoddur.

    #Resim Karelerinden Karekod Çözme İşlemi Yapan Metoddur.
    def DecodeBarcode(self,frame):
        barcodes = pyzbar.decode(frame)
        for barcode in barcodes:
            (x, y, w, h) = barcode.rect
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
            barcodeData = barcode.data.decode("utf-8")
            barcodeType = barcode.type
            text = "{} ({})".format(barcodeData, barcodeType)
            print("barcode:"+text)
            url="/ZiyaretciGirisKontrol?GuidCode="+barcodeData
            isOk,response=Helper.MakeGet(url)
            print(response)
            if str(response)=="True":
                GPIO.output(rolepin, True)
                time.sleep(2)
                GPIO.output(rolepin,False)
    #Resim Karelerinden Karekod Çözme İşlemi Yapan Metoddur.

    #Resim Yüz Tanıma Yapan Metoddur.
    def RecognizeFace(self,frame):
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        rgb = imutils.resize(frame, width=250)
        r = frame.shape[1] / float(rgb.shape[1])
        boxes =
face_recognition.face_locations(rgb,model=self.detectionMethod)
        encodings = face_recognition.face_encodings(rgb, boxes)
        names = []
        for encoding in encodings:
            matches =
face_recognition.compare_faces(self.data["encodings"],encoding)
            name = "Bilinmiyor"
            if True in matches:
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}
                for i in matchedIdxs:
                    name = self.data["names"][i]
                    counts[name] = counts.get(name, 0) + 1
                name = max(counts, key=counts.get)
                recarray=[name]
                self.array=recarray+self.array[:2]
                print(self.array)
            names.append(name)

        for ((top, right, bottom, left), name) in zip(boxes, names):
            top = int(top * r)
            right = int(right * r)
            bottom = int(bottom * r)
            left = int(left * r)
            cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 0), 2)
            y = top - 15 if top - 15 > 15 else top + 15
```

```
            cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,0.75,
(0, 255, 0), 2)
            print(name)
        return names
    #Resim Yüz Tanıma Yapan Metoddur.

    #Form Kapatma Metodudur
    def onClose(self):
        print("Kapatılıyor...")
        self.stopEvent.set()
        self.vs.stop()
        self.root.quit()
    #Form Kapatma Metodudur


print("Kamera Hazırlanıyor...")
#Kamerayı Çalıştır
vs=VideoStream(usePiCamera=True).start()
time.sleep(2.0)
firstarr=[0,0,0]
#Tanıma Form Objesinin Ayarlanması
pba = PersonelTanima(vs,firstarr)
#Form Döngüsünün Başlatılması
pba.root.mainloop()


#FaceRecognition.py Dosya Sonu
```

ÖZGEÇMİŞ

Alper AKKAYA 1989 yılında Merzifon'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. Merzifon Anadolu Lisesi'nden mezun oldu. 2008 yılında Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği'nde öğrenime başlayıp 2012 yılında mezun oldu. 2013 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. Askerlik hizmetini tamamladıktan sonra Bursa'da Teracity Yazılım Ar-Ge Merkezi'nde 3,5 yıl çalıştı. Şu an Bursa'da Özdilek Holding bünyesinde yazılım alanında görev yapmaktadır.

ADRES BİLGİLERİ

Adres : Soğanlı Mah. İstanbul Cad.No:343/3 Osmangazi / BURSA
E-posta : alperakkaya1989@gmail.com