İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Görüntü İşleme ile İnsan Sayımı

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Fulden Doğan Gürcan

Proje Danışmanı: Prof. Dr. Aytuğ Onan

Ağustos 2024

Görüntü İşleme ile İnsan Sayımı

Öz

Bu makale, bilgisayar görüşü kullanarak bir kamera aracılığıyla insanları sayan bir sistem sunmaktadır. Bu sistem, belirli bir alanda hareket eden insanların giriş ve çıkış bilgilerini kaydeder.

Kayıt işlemi, nesnelerin algılanması ve izlenmesi olmak üzere iki adımdan oluşur. Bu sistemde, algılanan nesneler insanlardır. Algılama işlemi daha doğru sonuçlar sağlasa da, maliyeti daha yüksek olduğundan daha ekonomik olan izleme yöntemiyle birlikte kullanılır. Bu izleme işlemi, algılanan kişilerin önceki karelerini izleyerek gerçekleştirilir.

Farklı testler kullanılarak, aynı alandaki çeşitli durumlardan elde edilen gerçek video serileri üzerinde yapılan testler, sistem tarafından %0 ile %100 arasında değişen başarı oranları verir. Bu testler, sistemdeki başarıyı doğrulamak için üst üste binme, farklı yönlere grup halinde hareket eden insanlar, koşarak geçme, geriye doğru yürüme gibi zorlu durumları içerir. Bu yazım kılavuzu, burada belirtilen tüm yazım kurallarına uygun olarak hazırlandığı için aynı zamanda şablon olarak da kullanılabilir. Yazım kurallarının detayları kılavuz içerisinde verilmiştir.

Anahtar Sözcükler: Görüntü İşleme, insan sayımı, nesne algılama, nesne izleme, gerçek zamanlı işleme, araştırma

People Counting with Image Processing

Abstract

This paper presents a system with Computer Vision for counting people through a camera. The system records the entry and exit information of walking people in the placed area.

This record requires two steps: detection and tracking the objects. In this system objects are people. The detection method gives more accurate results than the tracking method. The tracking is more cost-effective than the detection method. So, I use both of them for optimum performance and success. This process was made by tracking the previous frames of detected people. Different tests using a set of real video sequences taken from different situations in the same area give results ranging between 0% and 100% accuracies depending on the different situations like an arm in arm crossing the counting zone. Due to the COVID-19 outbreak, real-world samples consist of different situations we identified at the beginning of the project. Therefore, in some cases, such as people walking in too close each to other, counting cannot be performed correctly.

Problematic situations, such as overlapping, people grouped in different ways, running past, walking backwards etc., were used to validate the success of the system.Bu yazım kılavuzu, burada belirtilen tüm yazım kurallarına uygun olarak hazırlandığı için aynı zamanda şablon olarak da kullanılabilir. Yazım kurallarının detayları kılavuz içerisinde verilmiştir.

Keywords: Image Processing, People Counting, Object Detection, Object Tracking, Real-Time Processing, Research

Teşekkür

Proje çalışmasına olan katkılarından dolayı eşim Özgür Gürcan'a teşekkür ederim.

İçindekiler

Şekiller Listesi

Tablolar Listesi

Kısaltmalar Listesi

NFR	Non-Functional Requirements
UPS	Uninterruptible Power Supply
UI	User Interface
ER	Entity-Relationship
COCO	Common Objects in Context
XML	eXtensible Markup Language

Bölüm 1

Giriş

Bu bölümde, insanların belirli bir alanda sayılmasının öneminden ve bu konuda geliştirilen sistemin özelliklerinden bahsedilecektir. Günümüz teknolojisiyle geliştirilen görüntü işleme teknikleri sayesinde, kamera görüntüleri kullanılarak insan sayımı gibi zorlu görevler etkin bir şekilde gerçekleştirilebilmektedir. Bu projede, bu tür bir sistemin nasıl geliştirildiği ve uygulandığı ayrıntılı olarak ele alınacaktır.

1.1 Arka Plan Bilgisi

Günümüzde birçok yerde insanların anlık ve periyodik olarak sayılması gerekmektedir. Bazı durumlarda belirli yerlerdeki insan sayısını bilmemiz ve duruma göre hareket etmemiz gerekebilir. Örneğin, havaalanları, okullar, alışveriş merkezleri ve afet bölgeleri gibi yerlerde durum böyledir.

Afet durumlarında, deprem ve tsunami gibi felaketlerin ardından gelen kurtarma ekiplerinin binadaki insan sayısını bilmesi, kurtarma çabaları için hayati öneme sahiptir.

Diğer bir durumda, işletmenin ticari bölgesinde, mağazalar, alışveriş merkezinin kalabalık zaman verilerini kullanarak daha fazla müşteriye hizmet etmek için reklam yapabilir ve ekstra çalışanlar işe alabilir.

Güvenlik alanında, havalimanları, otobüs terminalleri ve benzeri yerlerde personel dışında erişilmemesi gereken alanlarda insan varlığını tespit etmek için kullanılabilir. Bu nedenle, kamera olan her yerdeki insan sayısını bilmek gerekiyorsa, insan akışını izleyen bir sistem gereklidir.

Bu soruna sunduğum çözüm, zaten kurulu olan kameraların yardımıyla insan akışını saymaktır. Örneğin, Şekil 1.1'de görüldüğü gibi, kapının önünde zaten kurulu bir kamera bulunmaktadır. Dolayısıyla, sadece bir yazılıma ihtiyacımız var. Bu işlemi herhangi bir ek maliyet olmadan gerçekleştirdim.

Şekil 1.1: Kapı Kamerası

1.2 Problem Tanımı, Boyutu ve Özellikleri

Günümüzde, doğru insan sayısını tespit etmek için birçok farklı yöntem kullanılmaktadır. Bunlardan bazıları parmak izi tarayıcıları, kart okuyucuları ve turnikeler gibi yöntemlerdir. Ancak, bu yöntemlerde bazı zorluklar bulunmaktadır. Bu yöntemlerin zorlukları aşağıda açıklanmıştır.

İnsanların parmak izlerini almak yasal olarak zorlayıcı bir süreçtir, Şekil 1.2'de görüldüğü gibi. Örneğin, bir alışveriş merkezine giriş ve çıkışta insanlardan parmak izi istemek doğru bir yaklaşım değildir. Çünkü insanlar zamanlarını buna harcamak istemezler. Ayrıca, bazı insanlar herhangi bir yerde parmak izlerini vermek istemeyebilirler.

Şekil 1.2: Parmak İzi paragraf Tarayıcı

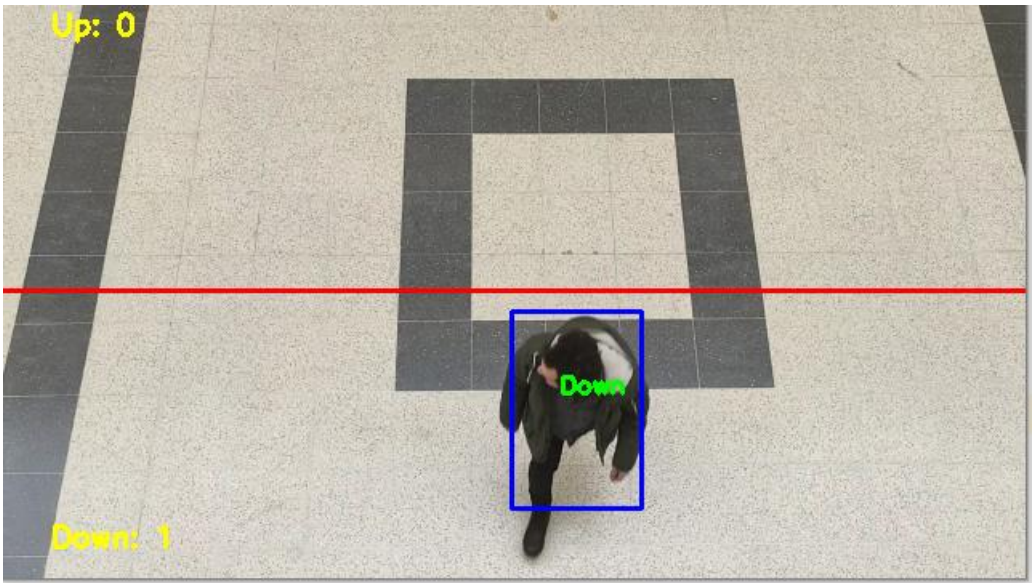
Öte yandan, Şekil 1.3'te görüldüğü gibi, turnike sistemi, giriş ve çıkışın yavaşlığı nedeniyle işlevsiz olacaktır. Bu nedenle, turnike sistemi de verimsizdir.

Şekil 1.3: Turnike Sistemi

Öte yandan, Şekil 1.3'te görüldüğü gibi, turnike sistemi, giriş ve çıkışın yavaşlığı nedeniyle işlevsiz olacaktır. Bu nedenle, turnike sistemi de verimsizdir.

1.3 Motivasyon/İlgili Çalışmalar

Bir çalışmada, insan akışı, açık cv ve python kullanılarak raspberry pi aracılığıyla sayılmıştır (Instructables, 2017). "Çizgi" kavramı, Şekil 1.4'te gösterildiği gibi tanıtılmıştır. Bu fikir, sanal bir çizgi oluşturmayı ve gelen giden insanları belirlemeyi çok kullanışlı hale getirmiştir. Bu nedenle, bu projede bu fikir kullanıldı.



Şekil 1.4: Giriş- Çıkış Çizgisi

Bir blog yazısında, makine öğrenimi ile kalabalıkta insan tanıma işlemi yapılmaktadır (García, 2018). Bu şekilde insan sayısını kaydederek, bu bilgileri panelden web'de görünür hale getirmeyi planlıyoruz. Buradaki panel fikri beni etkiledi ve bu fikri projeye, Şekil 1.5'te gösterilen kullanıcı paneline eklemeye karar verdim.

Şekil 1.5: Gösterge Paneli

Bu çalışmalarda bazı eksiklikler bulunmaktadır. Örneğin, ilk çalışmada kullanıcı paneli eksiktir ve raspberry pi kullanılması gerekir, bu da kurulumu zorlaştırır. Diğer çalışmada ise insanlar anlık çerçevede bulunur. Ancak, belirli bir süre içinde meydana gelen insan akışını saymak istiyorum, o an çerçevede bulunan insan sayısını değil.

1.4 Amaç/Katkı

Projenin amacı, acil durumlar ve reklam amaçları da dahil olmak üzere belirli kullanım durumları için alışveriş merkezleri, okullar, sinema salonları, hastaneler ve uçak çıkışları gibi bazı yerlerdeki kamera girişinden gerçek zamanlı olarak insanları saymaktır. Acil durumlarda içerideki insan sayısına hızlı bir şekilde ulaşarak durumun çözümüne önemli bir katkı sağlayacaktır.

1.5 Proje Kapsamı

Bu projenin kapsamı, kameranın uygun koşullar altında yer aldığı ve insan akışının sayılması gereken yerlerdir. Uygun koşullar, kameranın konumu, açısı ve kalitesidir. Sistem, bu verileri işledikten ve depoladıktan sonra kullanıcı arayüzü üzerinden kontrol etme imkanı sağlar.

1.6 Metodoloji/Araçlar/Kütüphaneler

Çözüm yöntemi ve uygulama, kameradan alınan verilere dayanmakta, bu verilerin yerel makinede işlenmesini ve ardından bu bilgilerin web servisine aktarılmasını içermektedir.

Python IDE olarak spider, NodeJS IDE olarak Visual Studio Code, iletişim uygulaması olarak Slack, paylaşım ve depolama klasörleri için Google DOCS, belge yazımı için Microsoft Word, grafik çizimi için draw.io, Referans olarak Trello, GitHub, Puty, Bibme ve Araştırma için Google Scholar kullanıyorum. OpenCV, NumPy, dlib, imutils, ScilPy, Matplotlib kütüphanelerini kullanıldı.

Bölüm 2

Literatür Taraması

Bu alanda benzer çalışmalar bulunmaktadır. Projemize en yakın 10 çalışma aşağıda listelenmiştir.

Bir çalışmada araştırmacılar, güvenlik için bir binanın içinde tek bir kamera kullanarak gerçek zamanlı insan sayma sistemi önermişlerdir (Kim, Choi, Choi, & Ko, 2002). İki veya daha fazla insan çok yakın olduğunda, görüntüleri birbiriyle örtüşebilir. Kameranın bakış açısından, 2 kişi ardışık olarak bulunduğunda, bir kişi olarak görünebilirler. Bu problem, örtüşme problemi olarak adlandırılır. Örtüşme problemi çözmek için kamera, görüntünün düzgün işlenmesini sağlamak için tavana asılır.

Tokta & Hocaoglu, 2018, optik akış özelliklerinin sınıflandırmasına dayanan hızlı bir insan sayma algoritması sunmaktadır. Takip algoritmaları, bir kişinin belirli bir alana girmiş olup olmadığını belirlemek için kullanılır. Yaklaşımları bireyleri tanımlama veya izleme gerektirmediği için hesaplama açısından verimlidir.

Başka bir makale, hareket eden bireyleri tespit ve izlemeyi sağlayan bir sistem geliştirilmesini sunmaktadır (Rossi & Bozzoli, 1994). Bir dizi kare verildiğinde, sayım hattı, geçen insan sayısını saymak için tasarlanmıştır. Hareket algılama modülü öncelikle sahneye birinin girmiş olup olmadığını belirler; Tahminlerin birleştirilmesi ve eşleşmeleri içeren bir modül, insanları sayım hattını geçene kadar takip eder.

Başka bir makale, sahne sistemi içindeki insan akışını saymak için gelişmiş nesne tespiti ve takip algoritması ile iki yönlü bir sayıcı önermektedir (Yam, Siu, Law, & Chan, 2011). Sistemde kamera, kırk beş derecelik bir açıyla konumlandırılmıştır.

Başka bir makalede, bir kapı veya kapıdan geçen kişi sayısı bir video kamera tarafından sayılır (T.-Y. Chen, Chen, Wang, & Kuo, 2010). Temel fikir, öncelikle kare

farkını kullanarak insanların hareketli kesimlerini tespit etmektir ve sonra yüzleri bulmak için renk farkını kullanmaktır. Yüz tespiti yapıldıktan sonra, yüzü tanımlanan kişi izlenir ve sonra yüz sayma hattına dokunduğunda, o kişi sayılır.

Başka bir makale, kapıdan geçen yaya sayımının otomatik basit iki yönlü yöntemine dayanan görüntü işleme üzerine kuruludur (Cao, Sun, Odoom, Luan, & Song, 2016). Bu yöntemde, doğrudan aşağıya bakacak şekilde kapının tavana asılı olduğu tek bir ortak kamera kullanılır. Bir dizi deneme formülü, ön planda algılanan insan sayısını tahmin etmek için kullanılır. Daha sonra, birleştirme-bölme sayma stratejisi, birleşme veya bölme durumlarını çözmek için kullanılır.

Wahyuni, Alinra, & Setiawan'ın çalışması, iç mekan izleme için otomatik insan sayımı sistemi geliştirmeyi amaçlamaktadır. İnsan tespiti, Yönlü Gradyan (HOG) tabanlı özellik tanıyıcı ve Destek Vektör Makinesi (SVM) sınıflandırma histogramı kullanılarak gerçekleştirilmiştir. Sonuçlar, yanıltıcı algının, kamera ile nesne arasındaki mesafe, az ışık yoğunluğu ve nesne örtüşmeleri tarafından etkilendiğini göstermiştir.

Başka bir makale, koridor boyunca yürüyen insan sayısını gösterir (Pore & Momin, 2016). Kontrollü bir alandan geçen insanları sayma, bu projenin önemli bir konusudur. Algılama, hog tanımlayıcı uygulaması ile insanları bulmaya dayanır ve insanların yörüngeleri, Kalman kanalı uygulaması ile oluşturulur. Son olarak, sistem, Kalman filtresi tarafından üretilen yönler doğrultusunda içeri ve dışarı sayılır.

İş ve güvenlik uygulamaları için oldukça önemli bir belgede, bir alanın içine giren veya çıkan insanları otomatik olarak saymak için önerilmiştir (Saxena & Songara, 2017). Bu makale, sadece bir kamera kullanarak ilgi alanında etkileşimde bulunan birden fazla kişiyi sayabilen otomatik bir kişi sayma sistemini tanıtır. Algoritma, insanları tespit etmek için yüz tanıma, Viola-Jones yöntemini kullanır. Tek bir tavana monte edilmiş kamera kullanılarak, sistem gözlem altındaki bir bölgeye giden insan sayısını sayar. Sayım, yüzleri tespit etmek için görüntüleri analiz ederek gerçekleştirilir.

Bölüm 3

Gereksinimler/Gereksinim Mühendisliği

Gereksinimler iki kısma ayrıldı. İlk kısım fonksiyonel gereksinimler, ikinci kısım ise fonksiyonel olmayan gereksinimlerdir. Fonksiyonel gereksinimler Tablo 3.1'de, fonksiyonel olmayan gereksinimler ise Tablo 3.2'de gösterilmektedir.

3.1 Fonksiyonel Gereksinimler

Fonksiyonel gereksinimler, bir yazılım sistemi veya bileşenin davranışlarını, giriş ve çıkışlarını, iş sürecini, kullanıcı etkileşimini ve sistemin ne yapacağını belirtir.

Tablo 3.1: Fonksiyonel gereksinimler

NO	Functional Requirements Description
FR1	Camera position
FR2	Real-time processing
FR3	Monitoring
FR4	Database
FR5	Reporting
FR6	Software requirements
FR7	Operating manual
FR8	Operating system

3.1.1 Kamera Pozisyonu

Sistemin çalışması için kameranın konumu ve açısı önemli bir kriterdir. Sistemin en uygun kamera açısı, 90 derece yükseklikten olmalıdır.

3.1.2 Gerçek Zamanlı İşleme

Sistem tarafından alınan görüntüler anında işlenmelidir. Bir konuma giren ve çıkan yayalar sayılmalıdır.

3.1.3 İzleme

Güvenlik personeli, alandaki insan sayısını anında görebilmeli ve giren çıkan yayaları takip edebilmelidir.

3.1.4 Veri tabanı

Sistem tarafından işlenen görüntülerin sonuçları bir yerde saklanmalıdır. Bu yer yerel depolama veya veri tabanı olabilir. Sonuç, tamsayı ve tarih formatında olmalıdır.

3.1.5 Raporlama

Sistem, otomatik olarak günlük, haftalık veya aylık raporlar üretebilmeli ve bu raporları talep edildiğinde gösterebilmelidir.

3.1.6 Yazılım Gereksinimleri

Sistemin çalışması için yazılımın kurulması gerekmektedir. Bu yazılım, sistemin kurulmasından önce kurulmalıdır. Bu yazılım tarafımızca geliştirilecektir.

3.1.7 Kullanım Kılavuzu

Kullanıcıların sistemi kullanabilmesi için bir kullanım kılavuzu gereklidir. Kullanıcılar, sistem ilk kurulduğunda bir eğitim almalıdır.

3.1.8 İşletim Sistemi

Sistemin çalışması için bir işletim sistemi gereklidir. Bu işletim sistemi, Windows 10 olarak belirlenmiştir.

3.2 Fonksiyonel Olmayan Gereksinimler

Fonksiyonel olmayan gereksinimler, bir sistemin işleyişini değerlendirmek için kullanılabilecek kriterleri belirtir ve belirli davranışları tanımlayan fonksiyonel gereksinimlerle karşılaştırılırlar. Fonksiyonel gereksinimlerin uygulanması için plan, sistem tasarımında detaylandırılır.

Tablo 3.2: Fonksiyonel gereksinimler

NO	Non-Functional Requirements Description
NFR1	Warning
NFR2	Tracking
NFR3	Image quality
NFR4	Connection
NFR5	Panel
NFR6	Verification
NFR7	Encryption
NFR8	UPS

3.2.1 Uyarı

Alanın kapanma saati sonrasında alanda insan varsa, sistem güvenlik personeline bir uyarı vermelidir.

3.2.2 Takip

Sistem tarafından oluşturulan rapor, her zaman sistem yöneticisi tarafından erişilebilmelidir.

3.2.3 Görüntü Kalitesi

Dengeli bir sistem işleyişi için kamera, insanların diğer nesnelerden ayırt edilebileceği kadar yüksek çözünürlükte görüntü kaydetmelidir. Birçok binanın önündeki kameraların çekim kalitesi zaten buna yeterlidir.

3.2.4 Bağlantı

Verilerin yerel depolamanın dışında tutulması istendiğinde yüksek hızlı bir internet bağlantısı gereklidir.

3.2.5 Panel

Sistem yöneticisinin kullanıcı girişlerini daha kapsamlı ve kolay bir şekilde belirli bir süre boyunca izleyebilmesi için bir gösterge paneli oluşturulmalıdır.

3.2.6 Doğrulama

Sistem yönetici yetkilerine sahip olabilmek için, sistem yöneticisine özel olarak verilmiş bir kullanıcı adı ve şifre gereklidir.

3.2.7 Şifreleme

Sistemin güvenlik açıklarını önlemek için, veriler özel bir anahtarla şifrelenmelidir.

3.2.8 UPS

Sistemi, işlem sırasında herhangi bir güç kaybı durumunda meydana gelebilecek herhangi bir hasardan korumak için kesintisiz güç kaynağı kullanılmalıdır.

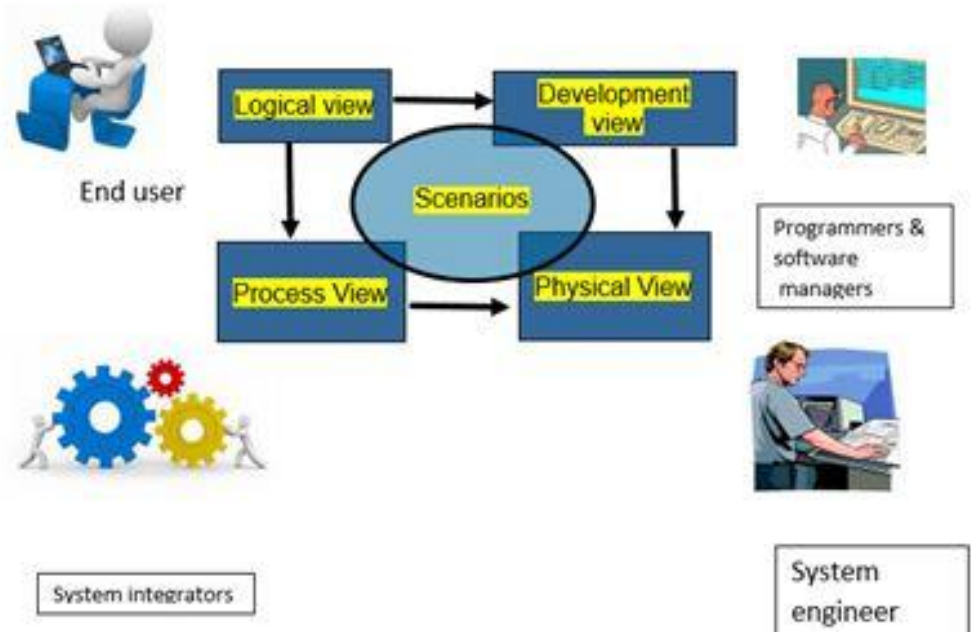
Bölüm 4

Tasarım

Bu bölümde, mimari görünüm, veri tabanı tasarımı/ER diyagramı, UML sınıf diyagramı, UI tasarımı, kullanım senaryoları, sıra diyagramı, etkinlik diyagramı, dağıtım yer almaktadır.

4.1 Mimari Görünüm

Şekil 4.1, mimari görünümün çalışma mantığını göstermektedir.



Şekil 4.1: Mimari Görünüm

4.1.1 Mantıksal Görünüm (Fonksiyonel Görünüm)

Fonksiyonel gereksinimler dikkate alınarak sistem kullanıcı dostu olarak tasarlanmıştır. Sistem, web paneli içeren birçok özellik sunar. Bu panoda, kameralardan işlenen görüntülerle hesaplanan belirli bir zamandaki insan akışı görüntülenir.

4.1.2 Süreç Görünümü

Eş zamanlılık, performans, ölçeklenebilirlik gibi fonksiyonel olmayan gereksinimler dikkate alınmıştır.

4.1.3 Geliştirme Görünümü

Yazılım modül organizasyonu, katmanların hiyerarşisi, yazılım yönetimi, yeniden kullanım, araçların kısıtlamaları gibi faktörler dikkate alınmıştır.

4.1.4 Fiziksel Görünüm (Dağıtım)

Altta yatan donanım hakkında fonksiyonel olmayan gereksinimler dikkate alınmıştır. Dağıtım Diyagramı, Şekil 4.7'de verilmiştir.

4.1.5 Senaryolar

Sistem tutarlılık ve geçerlilik açısından ele alınmıştır. Kullanım Senaryoları (Şekil 4.4), senaryolar olarak kullanılabilir.

4.2 Veri tabanı Tasarımı/ER Diyagramı

Bu projede zaman damgası ve insan akışı verileri gereklidir. Bu nedenle, sadece bu iki sütunla bir tabloya sahibiz.

Tablo 4.1: Kişi sayma tablosu

Kişi Sayma Tablosu		
id	zaman	sayı
1	12/12/2019_16	3
2	12/12/2019_17	20
3	12/12/2019_18	80
4	12/12/2019_19	200
5	12/12/2019_20	400
6	12/12/2019_21	450
7	12/12/2019_22	582
8	12/12/2019_23	650
9	13/12/2019_00	646
10	13/12/2019_01	480
11	13/12/2019_02	213
12	13/12/2019_03	42

4.2.1 Kimlik

O anki zaman damgası için benzersiz bir kimlik.

4.2.1 Zaman

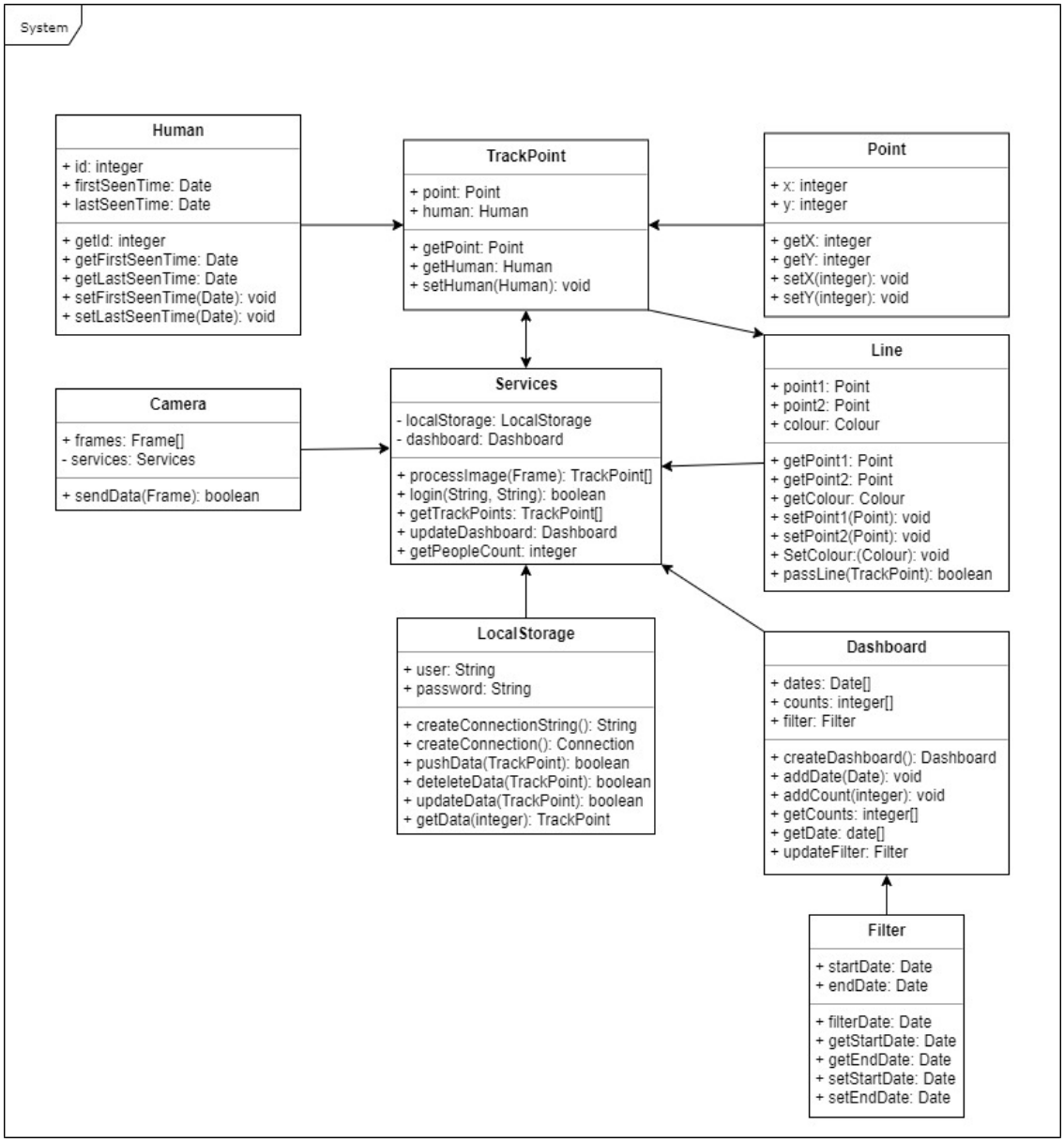
O anki zamana işaret eden sütun. Zaman formatı gg/aa/yy_ss.

4.2.3 Sayı

O anki binada bulunan insan sayısını gösteren sütun.

4.3 UML Sınıf Diyagramı

Şekil 4.2'de sınıf diyagramı bulunmaktadır. "İzlemeNoktası" sınıfı, "İnsan" ve "Nokta" sınıflarını içerir. "İnsan" sınıfı, kamerada görünen her yeni kişi için benzersiz bir kimlik ve insanların göründüğü ve kaybolduğu zamanları içerir. "Nokta" sınıfı, belirtilen kişinin ekranda göründüğü konumun koordinatlarını içerir.



Şekil 4.2: UML Sınıf Diyagramı

"Çizgi" sınıfı, "Nokta" sınıfını kullanarak sanal bir çizgi oluşturur. Bu çizgi, kendi özel renkleriyle kullanıcıya gösterilebilir.

"Kamera" sınıfı, kameradan gelen resimleri "çerçeve" formatında saklar.

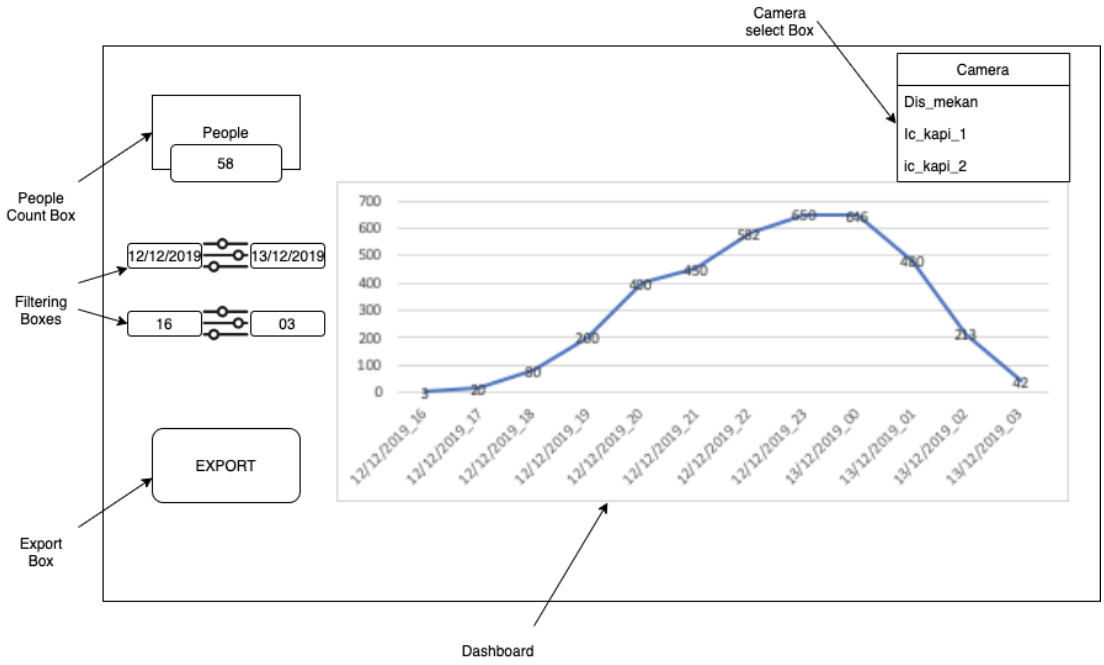
"Filtre" sınıfı, "Panel" sınıfındaki verileri filtrelemek için kullanılır. "Panel" sınıfı, kullanıcı için bir grafik sağlar.

"YerelDepolama" sınıfı, işlenmiş verilerin bir veritabanında saklanmasını sağlar. Bu veri "İzlemeNoktası" sınıfında saklanır.

"Hizmetler" sınıfı, tüm sistemin işleyişinden sorumludur. Kameradan gelen verileri işleyerek "İzlemeNoktaları" sınıfını oluşturur. Bu oluşan verileri "YerelDepolama" sınıfını kullanarak veritabanına gönderir.

4.4 Kullanıcı Arayüzü Tasarımı

Örnek kullanıcı arayüzü tasarımı Sekil 4.3'te gösterilmiştir.



Şekil 4.3: Kullanıcı Arayüzü Tasarımı

4.4.1 İnsan Sayma Kutusu

Örnek yönetici paneli UI tasarımı Şekil 4.3'te gösterilmiştir.

4.4.2 Filtreleme Kutusu

Kullanıcı, panoyu belirli bir zaman aralığı için görmek için zaman aralığını seçebilir.

4.4.3 Dışa Aktarma Kutusu

Kullanıcı, verileri Excel veya pdf gibi seçilen formatlarda yerel cihaza aktarabilir.

4.4.4 Kamera Seçim Kutusu

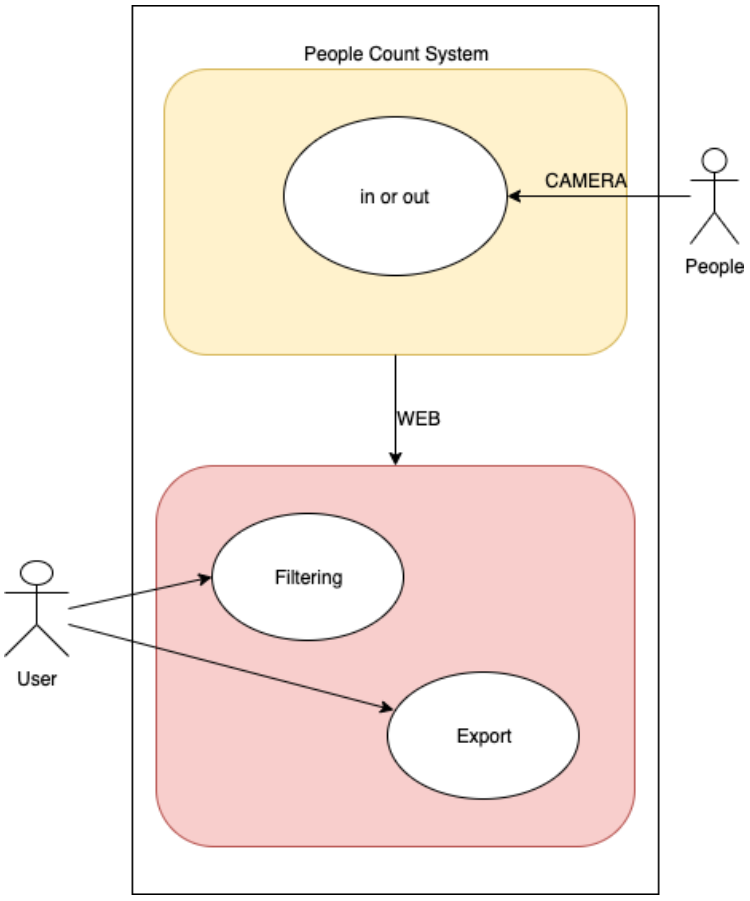
Kullanıcı, analiz için panoda gösterilecek kameraları seçebilir. Bu kameralar sisteme bağlıdır.

4.4.5 Gösterge Paneli

Belirli zaman aralıklarındaki insan sayısını gösteren grafik.

4.5 Kullanım Durumları

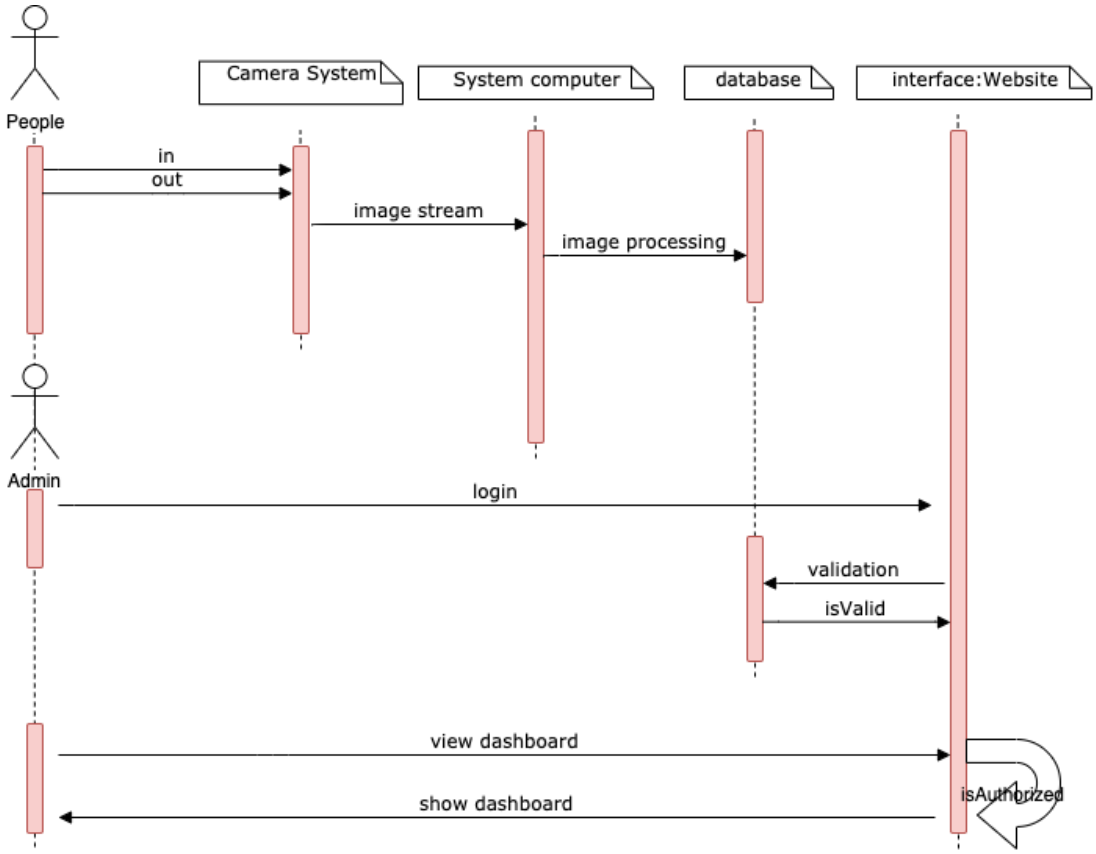
Kamera tarafından alınan görüntü, sisteme gönderilir. Kullanıcı, web sitesinden istediği herhangi bir zaman aralığı için rapor talep eder. Kullanıcı tarafından istenen rapor, belirtilen zaman aralığına göre o anda mevcut olan insan sayısını içerir. Kullanıcı, web sitesinden filtreleme yaparak herhangi bir tarih aralığını veya zaman aralığını seçebilir. Web sitesi, kullanıcının istediği raporları indirmesine izin verir ve raporlar excel, pdf ve word dosya formatlarında kullanılabilir. Bu akış Şekil 4.4'te gösterilmiştir.



Şekil 4.4: Kullanım Durumları

4.6 Sıra Diyagramı

Kameranın yakaladığı görüntü sistem bilgisayarına gönderilir. Sistem bilgisayarı, kameradan gelen görüntüleri işler ve insan akışını hesaplar. Bu veri veritabanına gönderilir. Sistem kullanıcısı web sitesine giriş yapar. Web sitesi girişi doğrular. Sistem kullanıcısı web sitesinden istediği herhangi bir zaman aralığı için rapor talep eder ve panoyu günceller. Kullanıcı tarafından istenen rapor, belirtilen zaman aralıklarına göre o anki insan sayısını içerir. Sistem kullanıcısı web sitesinden herhangi bir tarih aralığını veya zaman aralığını seçebilir. Web sitesi sonuçları sistem kullanıcısına gösterir. Bu akış Şekil 4.5'te gösterilmiştir.

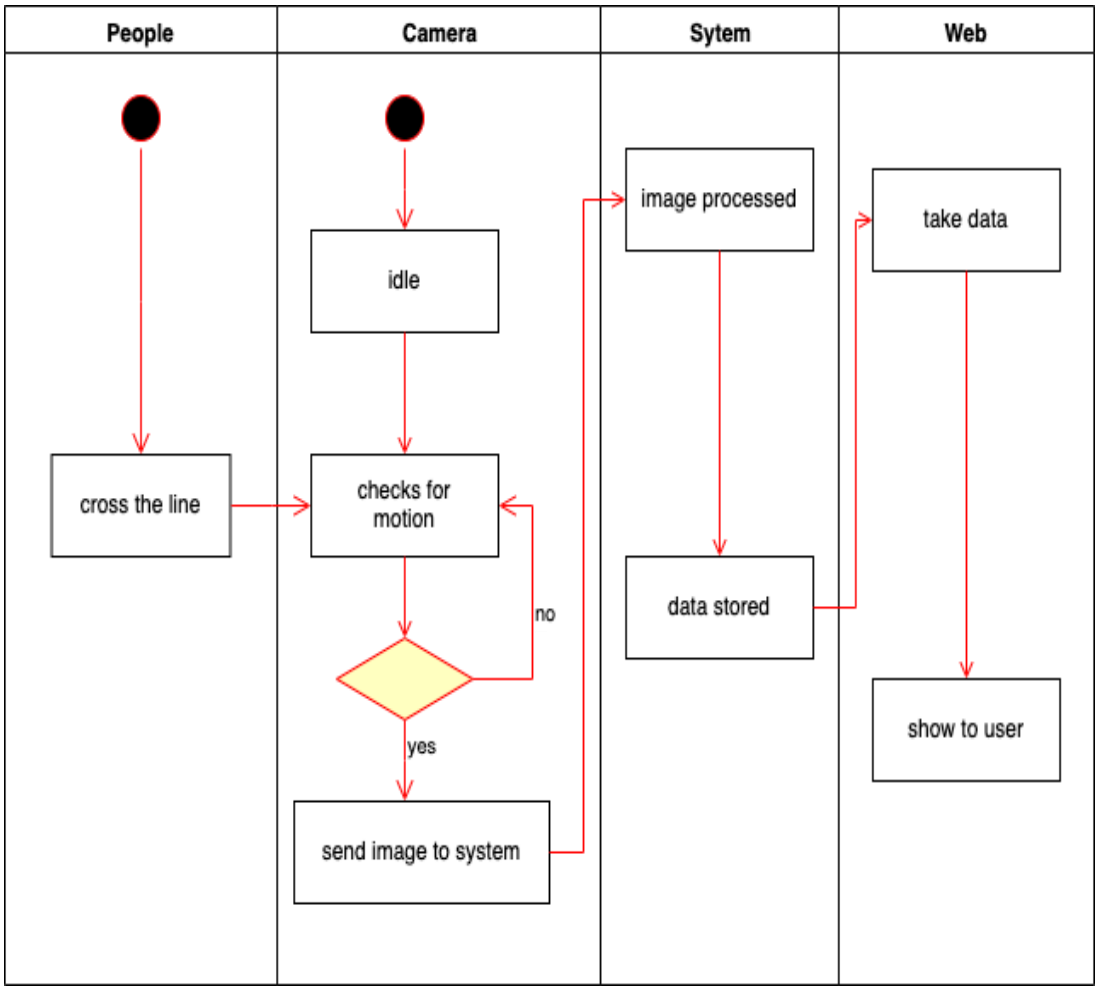


Şekil 4.5: Sıra Diyagramı

Şekil 4.5'e ek olarak, web sitesi, kullanıcının istediği herhangi bir zaman aralığına göre o anda mevcut olan insan sayısını içeren raporları indirmesine izin verir. Raporlar excel, pdf ve word dosya formatlarında kullanılabilir.

4.7 Aktivite Diyagramı

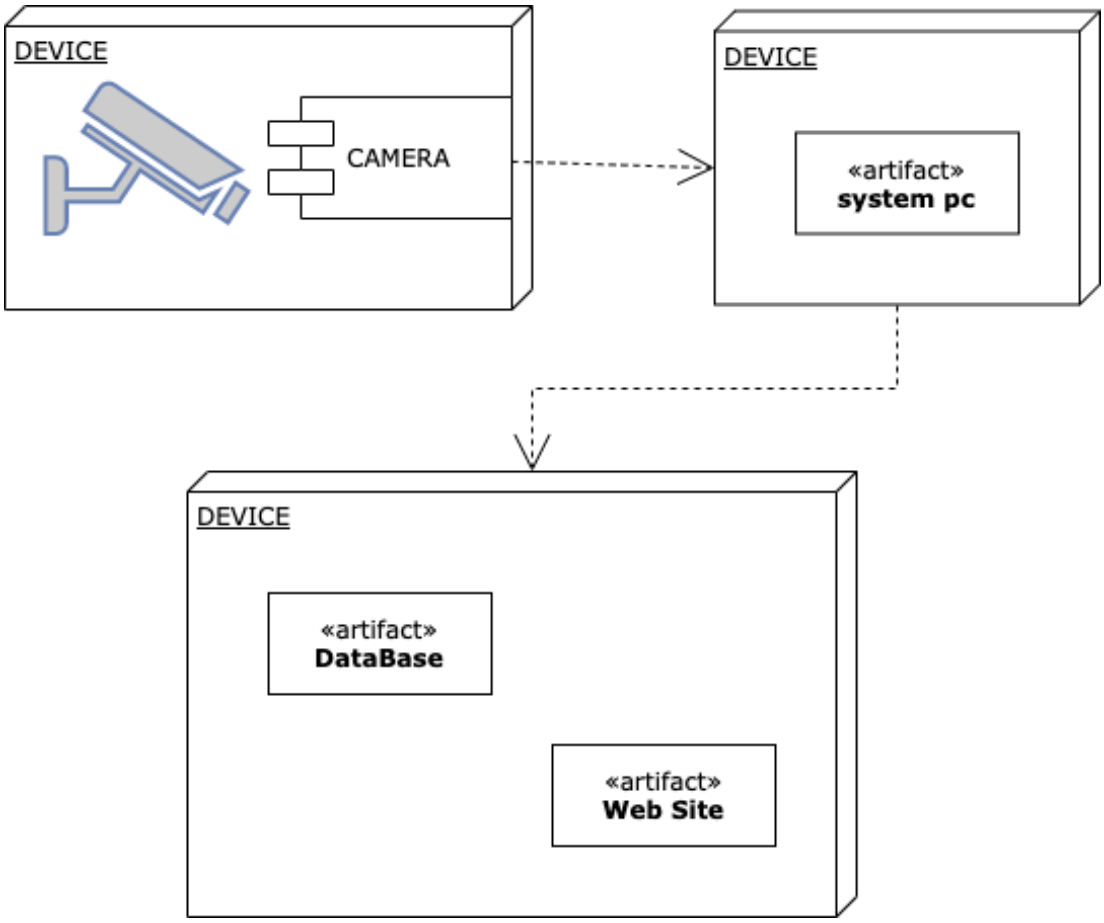
Kamera, bir hareket algılayana kadar bekler. İnsanlar kameradan geçtiğinde, kamera görüntü bilgilerini sistem bilgisayarına gönderir. Sistem, gelen görüntüyü işler. Ardından bu veriler web'e gönderilir. Web, bu bilgileri kullanıcıya gösterir ve bu verinin indirilmesine izin verir. Bu akış Şekil 4.6'da gösterilmiştir.



Şekil 4.6: Aktivite Diyagramı

4.8 Dağıtım Diyagramı

Dağıtım diyagramı Şekil 4.7'de verilmiştir. Kamera, sistem bilgisayarı ve veri tabanı ile web sitesini içeren sunucu cihazına sahibiz.



Şekil 4.7: Dağıtım Diyagramı

Kamera, insanların resimlerini alır. Kameradan gelen resimler, sistem bilgisayarına aktarılır.

Sistem bilgisayarı, kameradan gelen görüntü verilerini işler. İşlenen görüntülerden elde edilen bilgiler, sunucuya gönderilir.

Sunucuya gelen veri, veritabanına uygun bir formata dönüştürülür. Web sitesi, veritabanındaki verileri kullanıcı dostu bir arayüzde görüntüler.

Bölüm 5

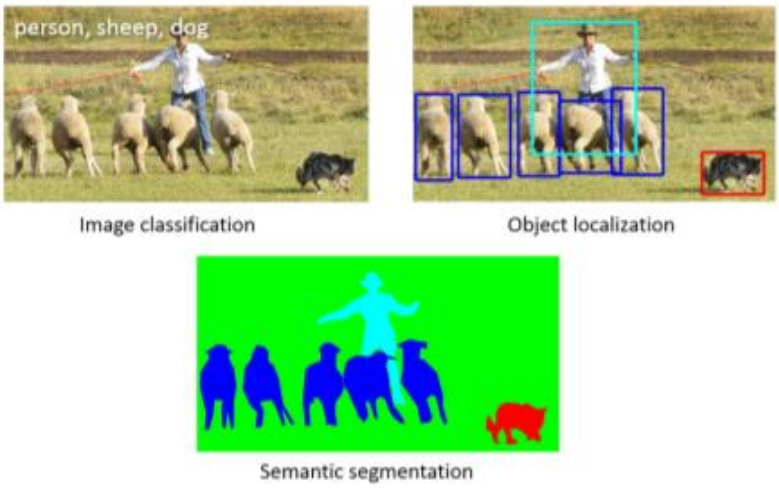
Uygulama

Bu bölümde, geliştirilen sistemin nasıl uygulandığı ve kullanılan yöntemlerin ayrıntıları ele alınacaktır. Projede kullanılan veri seti, yazılım araçları ve kod yapısı adım adım incelenecektir.

5.1 Veri Seti Açıklaması

Bu projede COCO veri seti kullanılmıştır. COCO, büyük çoğunlukla nesne tespiti ve segmentasyonu sağlayan bir veri setidir ("COCO- Common Objects in Context", n.d.). COCO'nun bazı özellikleri şunlardır:

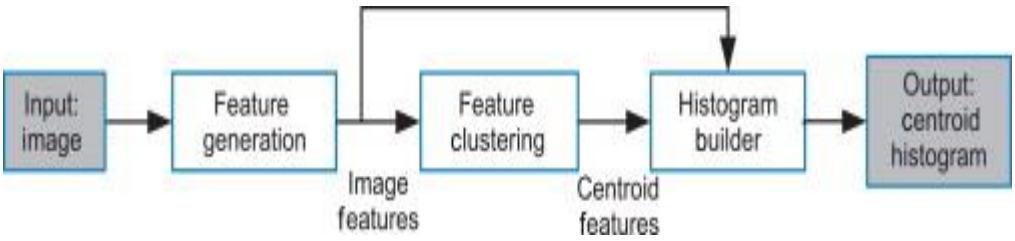
Nesne segmentasyonu, Bağlamda tanıma, Süper piksel eşya segmentasyonu, 330K görüntü (>200K etiketli), 1,5 milyon nesne örneği, 80 nesne kategorisi, 91 eşya kategorisi, -Görüntü başına 5 açıklama, -250,000 kişi anahtar noktaları ile COCO veri seti sayesinde görüntü sınıflandırma, nesne lokalizasyonu ve anlamsal segmentasyon işlemleri gerçekleştirilebilir (Şekil 5.1'de gösterildiği gibi).



Şekil 5.1: Görüntü işlemleri

5.1.1 Görüntü Sınıflandırma

Görüntü sınıflandırma, bir görüntüyü görsel içeriğine göre sınıflandıran bir süreçtir. Örneğin, bir görüntü sınıflandırma algoritması, bir görüntünün bir kişinin figürünü içerip içermediğini belirleyebilir. Nesne tespiti insanlar için basit olsa da, sağlam görüntü sınıflandırma, bilgisayarlı görü uygulamalarında bir sorundur (Kaeli, Mistry, Schaa ve Zhang, 2015). Görüntü sınıflandırma için yüksek düzeyli bir algoritma Şekil 5.2'de gösterilmiştir ve özellikler oluşturma, kümeleme ve histogram oluşturma adımlarını içerir.



Şekil 5.2: Görüntü sınıflandırma adımları (Kaeli, Mistry, Schaa ve Zhang, 2015)

5.1.2 Nesne Algılama

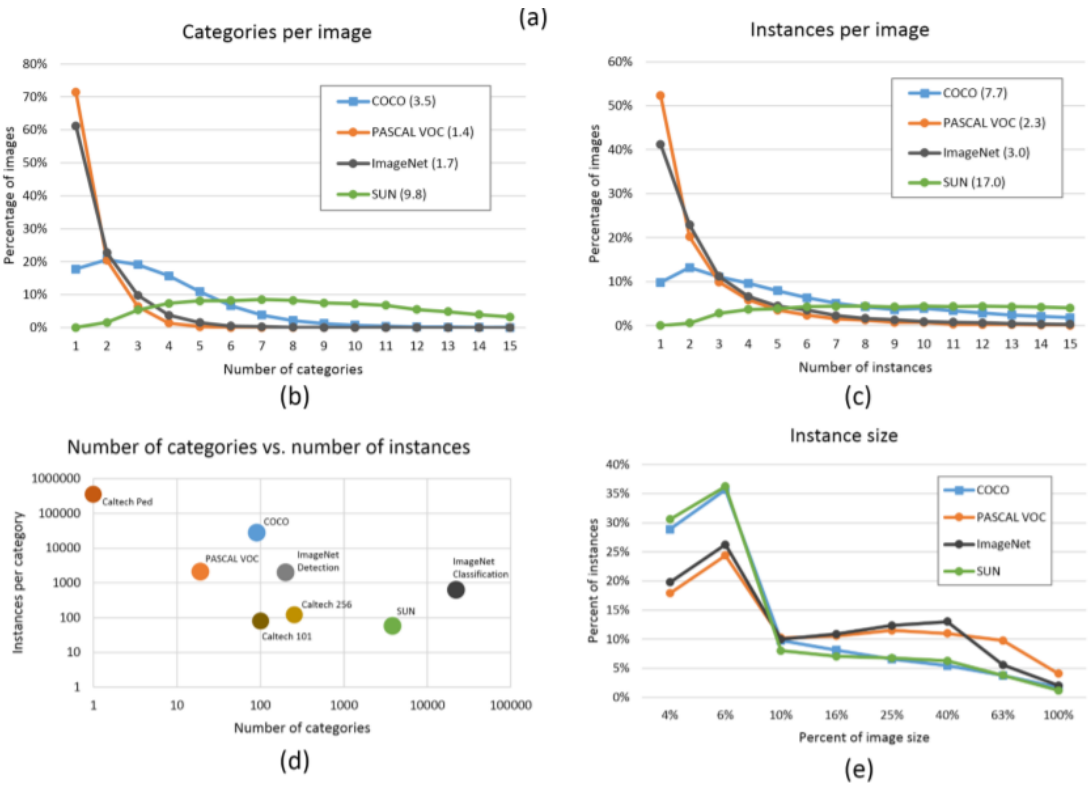
Nesne tespiti, dijital görüntüler ve videolarda anlamsal nesneleri (örneğin, insanlar, binalar veya arabalar) tespit eden bir bilgisayar teknolojisidir (Dasiopoulou, Mezaris, Kompatsiaris, Papastathis ve Strintzis, 2005).

5.1.3 COCO Veri Setinin Avantajları

2014 eğitim doğrulama verilerinde yaklaşık 270k segmentli kişi ve toplamda 886k segmentli nesne örneği vardır. 2015'in kümülatif sürümü toplamda 165482 eğitim, 81208 doğrulama ve 81434 test görüntüsü içerecektir (Lin ve diğerleri, 2015).

COCO veri seti, yaklaşık 270,000 segmentli insan görüntüsü içerir. Ek olarak, farklı kategorilerde toplamda 886,000 fotoğraf segmentlenmiştir.

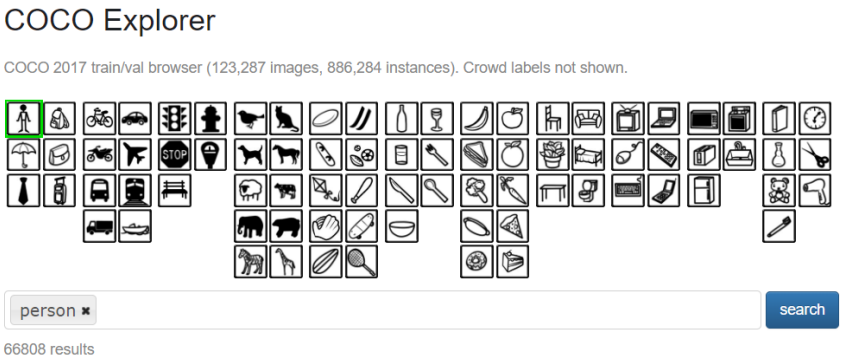
COCO veri seti, diğer veri setlerinden daha kapsamlıdır (Şekil 5.3'te gösterildiği gibi).



Şekil 5.3: Bazı veri setlerinin anlaşılması (Lin ve diğerleri, 2015)

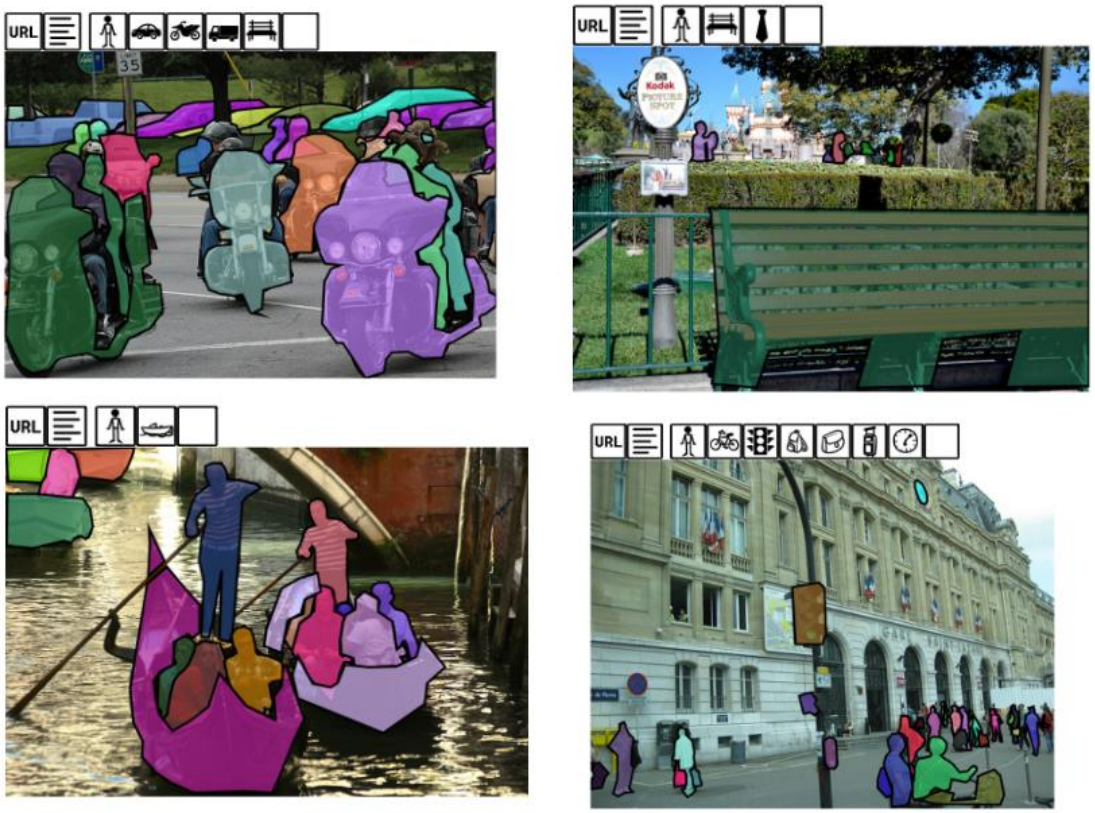
Bu çalışmada ihtiyacımız olan her bir kişiyi tespit etmektir. Diğer nesnelerin tanınması çalışmamız için gerekli değildir. Bu nedenle projemizde insan kategorisindeki örnek sayısının diğer kategorilere göre daha fazla olması daha doğru sonuçlar vermesi açısından önemlidir.

COCO veri setini daha spesifik olarak incelediğimizde, güncel olarak person olarak aradığımızda COCO explorer'da 66808 sonuç bulabiliriz. Bu arama sonucu Şekil 5.4'te gösterilmiştir.



Şekil 5.4: COCO Explorer

Şekil 5.5'te görüldüğü gibi bu sonuçlar sadece insanları içermemektedir. Farklı kategorilerdeki tüm nesneleri tanıyabilir: kişi, bisiklet, araba, motosiklet, uçak, otobüs, tren, kamyon, tekne, trafik ışığı, yangın musluğu, sokak tabelası, dur işareti, parkmetre, bank, kuş, kedi, köpek, at, koyun, inek, fil, ayı, zebra, zürafa, şapka, sırt çantası, şemsiye, ayakkabı, gözlük, el çantası, kravat, bavul, frizbi, kayak, snowboard, spor topu, uçurtma, beyzbol sopası, beyzbol eldiveni, kaykay, sörf tahtası, tenis raketi, şişe, tabak, şarap kadehi, bardak, çatal, bıçak, kaşık, kase, muz, elma, sandviç, portakal, brokoli, havuç, sosisli sandviç, pizza, çörek, kek, sandalye, kanepe, saksı bitkisi, yatak, ayna, yemek masası, pencere, masa, tuvalet, kapı, TV, dizüstü bilgisayar, fare, uzaktan kumanda, klavye, cep telefonu, mikrodalga, fırın, tost makinesi, lavabo, buzdolabı, karıştırıcı, kitap, saat, vazo, makas, oyuncak ayı, saç kurutma makinesi, diş fırçası, saç fırçası.



Şekil 5.5: COCO Explorer'dan örnek kişi görüntüleri ("COCO - Common Objects in Context", n.d.)

5.2 MobileNetSSD

Tek Atışta Nesne Tespiti (SSD), bir görüntüde birden fazla nesneyi tespit etmek için tek bir atış yapar. İki kısımdan oluşur: özellik haritalarını çıkarma ve nesneleri tespit etmek için evrişim filtresi uygulama.

SSD, Google araştırma ekipleri tarafından YOLO ve R-CNN arasında dengeyi korumak için geliştirilmiştir.

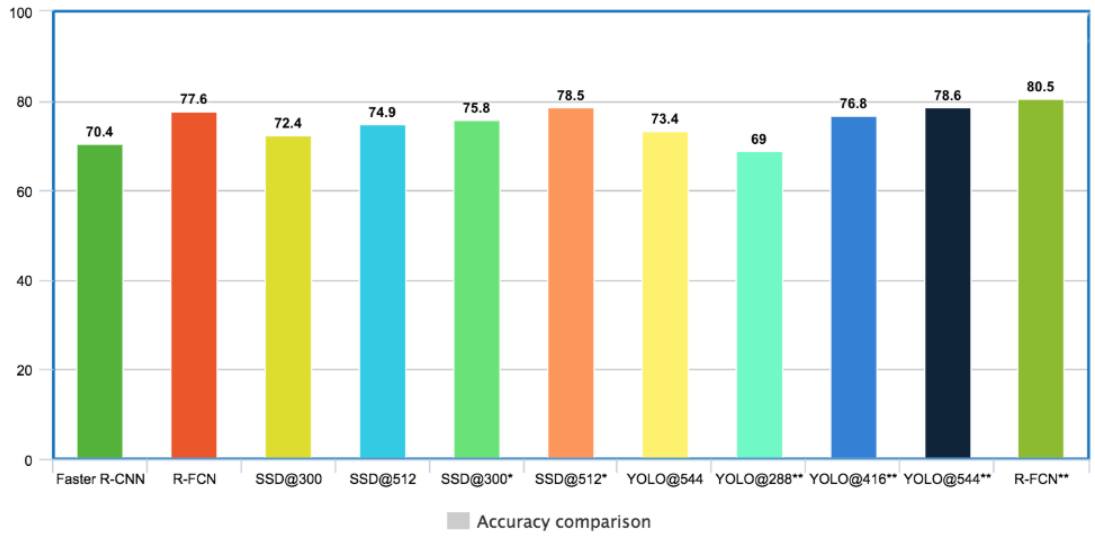
SSD300: Bu modelde giriş boyutu 300×300 ile sabitlenmiştir. Daha düşük çözünürlükte görüntüleri daha hızlı işlemek için kullanılır. SSD512'den daha hızlı çalışır. Ancak SSD300, SSD512'den daha az doğrudur.

SSD512: Bu modelde giriş boyutu 500×500 ile sabitlenmiştir. Daha yüksek çözünürlükteki görüntülerde kullanılır ve diğer modellere göre daha doğru sonuçlar verir.

Ek olarak, SSD, R-CNN'den daha hızlı çalışır. Çünkü R-CNN bir nesneyi tespit etmek için iki atışa ihtiyaç duyar. SSD ise bunu tek atışta yapabilir.

MobileNet SSD yöntemi önce COCO veri seti üzerinde eğitilmiş ve daha sonra PASCAL VOC üzerinde ince ayar yapılmıştır, %72.7 mAP (ortalama doğruluk) seviyesine ulaşmıştır.

Nesne tanıma algoritmalarının doğruluk karşılaştırmaları Şekil 5.6'da da gösterilmiştir. Şekil 5.6'da YOLO'nun doğruluğu daha iyi görünmektedir. Ancak gerçek zamanlı çalışırken görüntüleri yavaş işlediği için projeye uygun değildir.

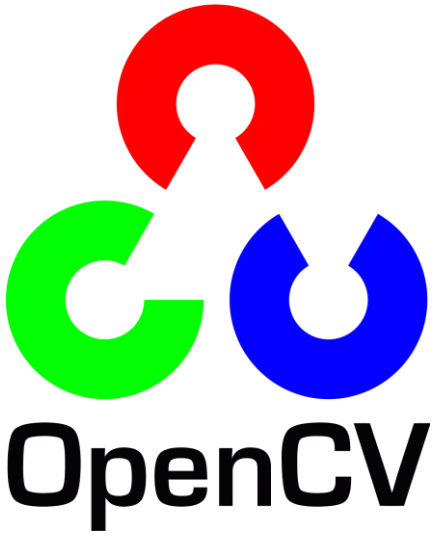


Şekil 5.6: Bazı algoritmalarla doğruluk karşılaştırması ("Single shot object detection SSD using MobileNet and OpenCV", 2020)

YOLO'nun sabit ızgara hücresi en-boy oranı varken, SSD daha iyi doğruluk için çoklu kutular ile farklı en-boy oranları kullanır.

5.3 OpenCV

OpenCV, açık kaynaklı bir bilgisayarlı görü ve makine öğrenimi yazılım kütüphanesidir. OpenCV, bilgisayarlı görü uygulamaları için ortak bir altyapı sağlamak ve ticari ürünlerde makine algılamanın kullanımını hızlandırmak için oluşturulmuştur ("OPENCV About", 2018).



Şekil 5.7: OpenCV

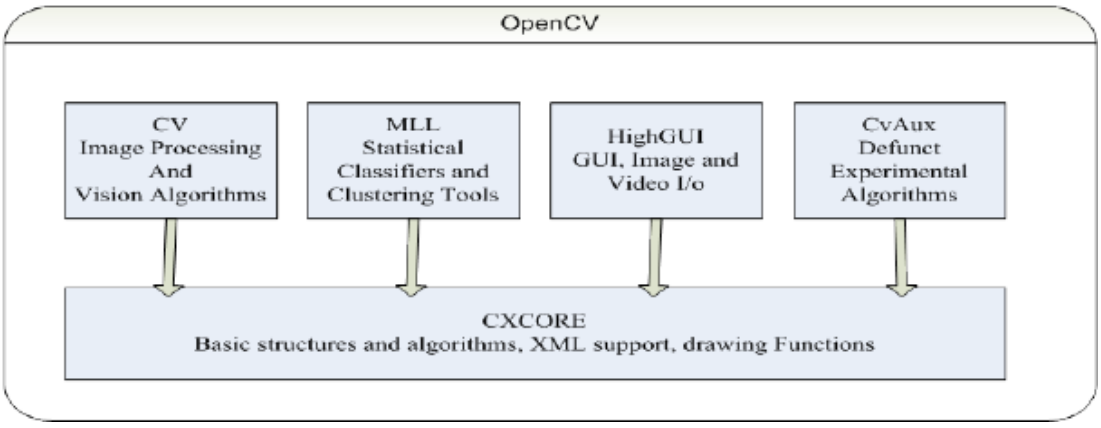
OpenCV, açık kaynaklı bir bilgisayarlı görü ve makine öğrenimi yazılım kütüphanesidir. OpenCV, bilgisayarlı görü uygulamaları için ortak bir altyapı sağlamak ve ticari ürünlerde makine algılamanın kullanımını hızlandırmak için oluşturulmuştur ("OPENCV About", 2018).

Her türlü alanda görüntü işleme için kullanılır. Bugün araç plakalarını okumadan yüz tanıma sistemlerine kadar geniş bir kullanım yelpazesi vardır. Bazı kullanımları şunlardır ("OpenCV", 2012): 2D ve 3D özellik setleri, Hareket tahmini, Yüz tanıma sistemi, Jest tanıma, İnsan-bilgisayar etkileşimi (HCI), Mobil robotlar, Hareket anlama, Nesne tanıma, Segmentasyon ve tanıma, Stereopsis stereo görüş: 2 kameradan derinlik algılama, Hareketten yapı (SFM), Hareket takibi, Artırılmış gerçeklik.

OpenCV programlama dilleri C++, C, MATLAB, Python, Java, C#'da mevcuttur. Yeni gelişmeler, OpenCV'nin doğal dili olan C++ ile yapılmaktadır.

Google, Yahoo, Microsoft, Intel, Sony, Honda, Toyota, IBM gibi büyük kullanıcıları vardır. Bu kadar çok tercih edilmesinin en büyük nedeni, açık kaynak olması ve yazılan kodun basit olmasıdır.

OpenCV beş temel bileşenden oluşur ve bu bileşenler Şekil 5.8'de gösterilmiştir ("OpenCV Structure and Content (Overview)", 2010).



Şekil 5.8: OpenCV beş temel bileşeni

CV (Computer Vision-Seeing with computer) Component: Temel görüntü işleme fonksiyonlarını ve algoritmalarını içerir.

MLL (Machine Learning Library) Component: Makine öğrenimi dalı için gereken fonksiyonları içeren istatistiksel veri kütüphanesidir.

HighgGUI Component: Kütüphane içindeki tanımlanmış nesnelerin oluşturulmasını ve görüntülerin ve videoların kaydedilip silinmesini sağlayan fonksiyonları içerir.

CXCore Component: Genel veri yapıları barındıran, bir görüntü üzerinde çizim yapılmasını sağlayan ve XML desteği sağlayan bir kütüphanedir.

CvAux Component: Ağız takibi, yüz tanıma, şekil eşleştirme gibi algoritmaları içeren bir kütüphanedir.

5.4 Bağımlılık Kurulumu

Sistemimizin çalışması için bazı kütüphanelerin yüklenmesi gerekmektedir. Bunlar OpenCV, imutils, numpy, dlib'dir. PIP modülü sayesinde bu tür kütüphaneleri kolayca yükleyebiliriz. PyCharm IDE kullanarak, tüm bağımlılıklar otomatik olarak yüklenir ("Features - PyCharm", n.d.).

5.5 Arka Uc Uygulamasi

Arka uç için Python ve PyCharm kullanılmıştır. Kullandığımız kütüphaneler Şekil 5.9'da gösterilmiştir. Sistemimizi çalıştırmak için dışarıdan birkaç kütüphane kullanmamız gerekti. Bunlardan bazıları: OpenCV kütüphanesi, Python'da cv2 olarak adlandırılır, Görüntüler üzerinde işlem yapmamıza olanak tanıyan imutils kütüphanesi, Bilimsel hesaplamalar yapmamıza olanak tanıyan numpy kütüphanesi

```
import datetime
import cv2
import dlib
import imutils
import time
import numpy as np
from imutils.video import FPS
from imutils.video import VideoStream
from centroidtracker import CentroidTracker
from trackableobject import TrackableObject
```

Şekil 5.9: Ana kodda ithal edilen kütüphaneler ve paketler

5.5.1 Centroid Tracker

Nesneleri takip etmek ve tespit etmek için bir veri yapısına ihtiyacımız vardı. Veri yapımız sayesinde CentroidTracker, nesnelerimizin takibini yaparken nesnenin geçmiş koordinatları gibi önemli bilgiler sağlar. Bu veri yapısı birkaç kütüphaneden yararlanmıştır. Bunlardan bazıları: Mesafe kütüphanesi, OrderedDict kütüphanesi, Numpy kütüphanesi.

CentroidTracker veri yapısında, bir nesnenin ne zaman kaybolacağını belirten sabit parametreler kullanılmıştır. Ek olarak, bu veri yapısı her tespit edilen nesne için bir kimlik (ID) tutar.

Bu veri yapısı üç önemli fonksiyondan oluşur. Tespit edilen nesne yeni ise, register fonksiyonunu çağırır. Böylece kimlik atama işlemi sağlanır ve sistem tarafından izlenmeye başlanır. Bu nesne çerçeveden çıktığında deregister fonksiyonu çağrılarak nesnenin kaybolması sağlanır.

Update fonksiyonu her çerçevede çalışır ve veri yapımızdaki tüm işlemlerin yapıldığı en önemli fonksiyondur. Bu fonksiyon, nesne çerçeveden kaybolduğunda deregister fonksiyonunu çalıştırır, çerçeveye yeni giren ve yeni tespit edilen nesne için register fonksiyonunu çağırır. Ek olarak, tespit edilen nesnenin her çerçevedeki konumunu belirler ve eski konumunu bir listeye koyarak geçmiş konum bilgisine erişim sağlar.

Centroid tracker sınıfındaki kütüphaneler Şekil 5.10'da gösterilmiştir.

```
from scipy.spatial import distance as dist
from collections import OrderedDict
import numpy as np
```

Şekil 5.10: Centroid tracker sınıfındaki ithal edilen kütüphaneler ve paketler

Bu sınıfta 4 farklı fonksiyon vardır. İlk olarak init fonksiyonunda merkez nesnesi başlatılmıştır. maxDisappeared değişkeni, bir kişinin kaybolmuş olarak işaretlenmesi için gereken çerçeve sayısını belirtir. maxDistance değişkeni, nesneleri ilişkilendirmek için merkezler arasındaki maksimum mesafeyi ifade eder. Nesneler arasındaki mesafe maksimum mesafeden büyükse kaybolmuş olarak işaretlenir.

Register fonksiyonunda, bir nesneyi kaydederken, merkez noktayı saklamak için kullanılabilir bir sonraki nesne kimliği kullanılır.

Deregister fonksiyonunda bir nesne kimliğini kayıttan çıkarmak için, nesne kimliğini her iki ilgili sözlüğümüzden de sileriz.

Bu fonksiyonlar Şekil 5.11'de gösterilmiştir. Update fonksiyonu Şekil 5.12 ve 5.13'te gösterilmiştir.

```
class CentroidTracker:
    def __init__(self, maxDisappeared=50, maxDistance=50):
        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()
        self.maxDisappeared = maxDisappeared
        self.maxDistance = maxDistance

    def register(self, centroid):
        self.objects[self.nextObjectID] = centroid
        self.disappeared[self.nextObjectID] = 0
        self.nextObjectID += 1

    def deregister(self, objectID):
        del self.objects[objectID]
        del self.disappeared[objectID]

    def update(self, rects):...
```

Şekil 5.11: Centroid tracker'ın fonksiyonları

Update fonksiyonu her çerçevede çalışır. Bu durumda, yeni nesneler görünürse register fonksiyonunu çağırır. Nesneler çerçeveden kaybolursa deregister fonksiyonunu çağırır ve nesneyi siler. Ayrıca update fonksiyonu her çerçevede nesnenin koordinatlarını yeniler.

```
def update(self, rects):
    if len(rects) == 0:
        for objectID in list(self.disappeared.keys()):
            self.disappeared[objectID] += 1
            if self.disappeared[objectID] > self.maxDisappeared:
                self.deregister(objectID)
        return self.objects

    inputCentroids = np.zeros((len(rects), 2), dtype="int")

    for (i, (startX, startY, endX, endY)) in enumerate(rects):
        cX = int((startX + endX) / 2.0)
        cY = int((startY + endY) / 2.0)
        inputCentroids[i] = (cX, cY)

    if len(self.objects) == 0:
        for i in range(0, len(inputCentroids)):
            self.register(inputCentroids[i])

    else:
        objectIDs = list(self.objects.keys())
        objectCentroids = list(self.objects.values())

        D = dist.cdist(np.array(objectCentroids), inputCentroids)

        rows = D.min(axis=1).argsort()

        cols = D.argmin(axis=1)[rows]
```

Şekil 5.12: Update fonksiyonu bölüm I

```
            usedRows = set()
            usedCols = set()

            for (row, col) in zip(rows, cols):
                if row in usedRows or col in usedCols:
                    continue

                if D[row, col] > self.maxDistance:
                    continue

                objectID = objectIDs[row]
                self.objects[objectID] = inputCentroids[col]
                self.disappeared[objectID] = 0

                usedRows.add(row)
                usedCols.add(col)

            unusedRows = set(range(0, D.shape[0])).difference(usedRows)
            unusedCols = set(range(0, D.shape[1])).difference(usedCols)

            if D.shape[0] >= D.shape[1]:
                for row in unusedRows:
                    objectID = objectIDs[row]
                    self.disappeared[objectID] += 1

                    if self.disappeared[objectID] > self.maxDisappeared:
                        self.deregister(objectID)

            else:
                for col in unusedCols:
                    self.register(inputCentroids[col])
        return self.objects
```

Şekil 5.13: Update fonksiyonu bölüm II

5.5.2 Trackable Object

TrackableObject veri yapımız, CentroidTracker veri yapımızdaki merkezlerin sayılıp sayılmadığını takip eden bir veri yapısıdır.

Nesne kimliğini saklıyoruz, ardından mevcut merkez noktasını kullanarak merkez noktalarının bir listesini başlatıyoruz. Sayılan değişken, nesnenin zaten sayılıp sayılmadığını kontrol eder. Bu sınıf Şekil 5.13'te gösterilmektedir.

```
class TrackableObject:
    def __init__(self, objectID, centroid):
        self.objectID = objectID
        self.centroids = [centroid]
        self.counted = False
```

Şekil 5.14: Trackable object sınıfı

5.5.3 Ana Kod

Sistemin kurulumu sırasında bazı ayarlamalar yapılması gerekmektedir. Bu ayarlamalar, bulunan koşullara göre yapılan ayarlamalar sonucunda doğru çalışma oranını artırır. Bu ayarlar, işlenen videoların kaydedilip kaydedilmeyeceği, videonun gerçek zamanlı işlenip işlenmeyeceği, nesnelerin her çerçevede belirlenmesi için minimum güven değeri ve bilgisayarın gücüne bağlı olarak bu değerlerin belirlenmesidir. Bu değerler, sistem kurulumu sırasında kamera görüntü kalitesi ve diğer koşullara bağlı olarak değişir. Bu ayarları yaptıktan sonra, Python programlama dilinin, videoları işleyen bilgisayara yüklenmesi gerekmektedir.

Bu ayarları yaptıktan sonra COCO, veri setinden daha yüksek doğruluğa sahip bir model üzerinde çalışır. Bu modelin, görüntüyü işleyecek bilgisayarda bulunması ve bu modelin konumunun sisteme verilmesi gerekmektedir. OpenCV sayesinde, bu eğitilmiş modeli video işleme için kullanabiliriz.

Yaklaşımımız, belirli sayıda çerçevede nesneleri tespit etmek ve bu nesnelerin geri kalan çerçevelerinde tespit edilen nesneyi takip etmektir. Bu şekilde, yüksek maliyetlerden kaçınırız. OpenCV sayesinde tespit edilen nesnenin türü ve muhtemelen neye ait olduğu hakkında bilgi sahibi oluruz. Sistemi kurarken, minimum güven değeri ile elimine ederiz. Projemiz, tür olarak insanların algılanmasıyla sınırlıdır.

Tespit edilen nesneler bu işlemle takip edilir. Her çerçevede nesnenin konumları güncellenir. Ve nesnenin ortasını referans noktası olarak seçeriz. Ancak, tüm takip edilen nesnelerin geçmiş konum bilgilerini CentroidTracker veri türümüzde takip ederiz.

Takip edilen nesne, belirlediğimiz sanal çizgiyi geçtikten sonra, geçmiş konumların ortalamasına bakarak yönünü belirleriz. Geçmiş konumların ortalamasına baktığımız için, yön tepsimiz oldukça doğrudur.

Sanal çizgiyi geçen nesnelerin giriş ve çıkış bilgilerini ve yönünü kayıtlara geçiririz. Ayrıca, işlenen videoya zamanında inen ve çıkan kişi sayısını yazarız.

Bu, ana süreçlerin gerçekleştirildiği ve programın genel olarak çalışmasını sağlayan ana koddur. DEBUG değişkeni sayesinde gerekli değişkenleri görmemizi sağlar ve bakım yapılabilirliği sağlar. SAVEOUTPUT değişkeni, işlenen videonun kaydedilmesini sağlar. Program, görüntüyü almak için iki farklı şekilde giriş alabilir, bunlardan biri kamera ve diğeri önceden kaydedilmiş bir videodur. CAPTUREFROMWEBCAM değişkeni, görüntünün nereden alınacağını belirler. Input değişkeni, kamera kullanılmadığında videonun yolunu belirtir. minConfidence değişkeni, tespit edilen nesnenin istenilen turda bulunma olasılığı yüzdesini belirtir. Bir nesneyi tespit etmenin maliyeti, bir nesneyi takip etmekten çok daha fazladır. Bu yüzden çoğu çerçevede nesneyi takip ederiz ve belirli sayıda çerçeveden tespit ederiz. skip_frames değişkeni, bir nesnenin kaç çerçeve tespit edileceğini belirtir. Şekil 5.14'te gösterilmiştir.

```
DEBUG = True
#save videos
SAVEOUTPUT = True
CAPTUREFROMWEBCAM = False

input='videos/yakin.mp4'
minConfidence=0.25
skip_frames=30
```

Şekil 5.15: Yapılandırma

```
net = cv2.dnn.readNetFromCaffe("mobilenet_ssd/MobileNetSSD_deploy.prototxt",
"mobilenet_ssd/MobileNetSSD_deploy.caffemodel")
```

Şekil 5.16: Kod Parçası I

Bu kod parçasında, eğitilmiş modelin çıktısı yüklenir.

```
if CAPTUREFROMWEBCAM is False:
    vs = cv2.VideoCapture(input)
else:
    vs = VideoStream(src=1).start()
```

Şekil 5.17: Kod Parçası II

Bu kod parçasında, OpenCV'nin videocapture fonksiyonu kamera görüntüsü alınmak istendiğinde çalışır. Webcam görüntüsü alınmak istendiğinde videostream fonksiyonu çalışır.

```
frame = vs.read()
if CAPTUREFROMWEBCAM is False:
    frame = frame[1]
if frame is None:
    break
```

Şekil 5.18: Kod Parçası III

Bu kod parçasında, yeni çerçeve sürekli olarak çalışır ve yeni bir çerçeve bulunamadığında program sona erer.

```
if SAVEOUTPUT is not None and writer is None:
    fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    writer = cv2.VideoWriter('output/output{:}.avi'.format(datetime.datetime.now()), fourcc, 30,(W, H),
True)
```

Şekil 5.19: Kod Parçası IV

Bu kod parçasında, işlenen videoyu kaydetmek istendiğinde bir dosyaya kaydedilir. Daha sonra belirtilen çerçevede bir tespit yapılır. Şekil 5.15'te gösterilmiştir.

```
if totalFrames % skip_frames == 0:
    trackers = []

    #127.5 => Renklerin ortalaması
    #0.007843 => 1/127.5
    blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
    net.setInput(blob)
    detections = net.forward()

    for i in np.arange(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2] #bulunan doğru olma olasığı

        if confidence > minConfidence:
            idx = int(detections[0, 0, i, 1])#Bulunan nesnenin türü

            if idx != 15:#Person
                continue

            #Bulunan nesnenin konumu, kutu şeklinde değerleri
            box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
            (startX, startY, endX, endY) = box.astype("int")

            tracker = dlib.correlation_tracker()
            rect = dlib.rectangle(int(startX), int(startY), int(endX), int(endY))
            tracker.start_track(rgb, rect)

            trackers.append(tracker)
```

Şekil 5.20: Tespit döngüsü

Çerçeve sayımız belirtilen çerçeve aralığında değilse, tespit edilen nesneler izlenmeye devam eder. Şekil 5.20'de gösterilmiştir.

```
#Tracking
else:

    for tracker in trackers:
        tracker.update(rgb)
        pos = tracker.get_position()
        startX = int(pos.left())
        startY = int(pos.top())
        endX = int(pos.right())
        endY = int(pos.bottom())
        rects.append((startX, startY, endX, endY))
```

Şekil 5.21: Takip döngüsü

Daha sonra her çerçevede her nesnenin yönü, geçmiş konumun ortalamasından mevcut konuma göre tahmin edilir. Belirli bir çizgiyi geçtiğinde, tahmin ettiğimiz yöne göre giriş ve çıkış bilgileri belirlenir. Bu çizgi, ekranın tam ortasında yer alır. Şekil 5.21'de gösterilmiştir.

```
for (objectID, centroid) in objects.items():

    directionText = ''
    to = trackableObjects.get(objectID, None)
    if to is None:
        to = TrackableObject(objectID, centroid)
    else:
        y = [c[1] for c in to.centroids] # Dikeye bakıyor, cismin konumların
        direction = centroid[1] - np.mean(y)
        to.centroids.append(centroid)
        if not to.counted:
            if direction < -10 and centroid[1] < H // 2: #-10 is threshold
                totalUp += 1
                directionText = 'Up'
                directionTextArray.append('Up')
                to.counted = True

            elif direction > 10 and centroid[1] > H // 2:#10 is threshold
                totalDown += 1
                directionText = 'Down'
                directionTextArray.append('Down')
                to.counted = True

    trackableObjects[objectID] = to
```

Şekil 5.22: Yön tespiti

```
Total Down:1.00 Time: 2020-04-18 22:35:25.119318
Total Down:2.00 Time: 2020-04-18 22:35:26.474244
Total Down:3.00 Time: 2020-04-18 22:35:30.969731
Total Down:4.00 Time: 2020-04-18 22:35:31.709445
 Total Up:1.00 Time: 2020-04-18 22:35:34.684427
Total Down:5.00 Time: 2020-04-18 22:35:40.207767
Total Down:6.00 Time: 2020-04-18 22:35:44.269742
```

Şekil 5.23: Veri Formatı

Her giriş ve çıkış işlemi gerçekleştirildiğinde, geçen kişi sayısı ve tarih yukarıdaki veri formatında kaydedilir.

Bölüm 6

Test/Deneyler

Bu bölümde, geliştirilen sistemin çeşitli test ve deneylerle nasıl değerlendirildiği ayrıntılı olarak ele alınacaktır. Karşılaşılan problemler ve bu problemlere getirilen çözümler adım adım incelenecektir.

6.1 Problemler ve Çözümler

Bu projeyi yaparken birçok sorunla karşılaştım. Bu bölümde bu sorunlara değinilecektir.

6.1.1 İnsan Tanıma

Projenin ana zorluğu nesneleri tanımak ve insanları ayırt etmekti. Bunun için kendini kanıtlamış COCO veri setini tercih ettik. Bu projeyi yaparken birçok sorunla karşılaştım. Bu sorunların çoğunu çözdüm.

6.1.2 Yakın İnsanlar

İnsanlar genellikle yakından seyahat ettikleri için birden fazla kişiyi tanımak zordur. Bu nedenle, minConfidence değerini azaltarak bu sorun aşılmaya çalışıldı. Elbette bunun da dezavantajları vardır. Örneğin, insan benzeri ancak insan olmayan nesneleri insan olarak tanımlayabiliriz. Ancak bu büyük bir sorun değildir çünkü hastane, alışveriş merkezi, apartman girişi gibi kullanım alanlarımızda genellikle benzer nesneler bulunmaz.

Şekil 6.1: Yakın insanlar

6.1.3 Birbirine Bağlı İnsanlar

6.1.1'deki soruna ek olarak, insanlar birbirine bağlı yürüyebilir. Bu durumda, tek bir kişi olarak algılanırlar.

6.1.4 Birbirinin Ardından Gelen İnsanlar

İnsanlar sırayla geçerken, arkada kalan kişi kameradan görülemediği için tespit edilemez. Ancak kamera açısını dik konumda ayarlayarak bu sorunu aşıldı.

6.1.5 Yatay Yürüyen İnsanlar

Bazı durumlarda, insanlar belirli bir çizgiden yatay yerine dikey olarak geçebilir. Yönü belirlemek için tespit edilen yayanın önceki durumuna bakarız. Bu tür yatay geçiş durumlarında, yön tespitinde hassasiyet yetersiz olabilir ve yönü doğru tespit edemeyebilir. Bunun için bu hassasiyet değerini değiştirerek bu sorunu aşabiliriz.

6.1.6 Birden Fazla Kişi Geçme

6.1.4'teki çözümde kullanılan hassasiyet değerini çok düşürdüğümüzde, biri tam sınır çizgisinde geri hareket ederse, yanlış yön hesaplanabilir. Bu iki sorunu çözmek için optimum bir değer kullanırız.

6.1.7 Aynı Anda Çok Fazla İnsan

Alışveriş merkezi ve hastane gibi yerlere çok sayıda giriş olabilir. Bu durumlarda, aynı anda birçok kişi tespit edilip takip edilmelidir. Tespit işlemi maliyetli olduğundan, işlemciyi zorlar. İşlemcinin yükünü azaltır ve tespit işlemini daha az sıklıkla yaparız, böylece her bilgisayarda kararlı bir şekilde çalışır.

6.1.8 Ters Geçiş

Biri ters/yana doğru yürüyorsa yanlış tespit edilebilir. Ancak önceki konumlarına bakarak bu sorunu aşıldı.

6.1.9 İşletim Sistemi Uyumsuzluğu

Gerçek kamera olarak kullanmamızı sağlayan 3. taraf telefon kamera uygulaması macOS sisteminde çalışmamaktadır. Bu yüzden bir Windows dizüstü bilgisayar bulmak zorunda kalındı.

Bölüm 7

Sonuç

Bu projede, bir kameradan insanları saymak için Bilgisayarlı Görme içeren bir sistem tasarlandı. Bu sistem, belirli bir alanda hareket eden insanların giriş ve çıkış bilgilerini kaydeder.

Bu kayıt, tespit edilen kişilerin önceki çerçevelerinin izlenmesiyle yapıldı. Tespit yöntemi, takip yönteminden daha doğru sonuçlar verir. Takip, tespit yönteminden daha maliyetlidir. Bu nedenle, optimum performans ve başarı için her ikisi de kullanılır. COVID-19 salgını nedeniyle, gerçek dünya örnekleri projenin başında belirlediğim farklı durumları içermektedir. Bu nedenle, insanların birbirine çok yakın yürüdüğü bazı durumlarda sayım doğru yapılamaz.

Proje, üst üste binen insanlar, farklı şekillerde gruplanmış, koşarak geçen, geri yürüyen vb. gibi farklı durumlarda %0 ile %100 arasında başarıyla çalışır.

Kaynaklar

Cao, J., Sun, L., Odoom, M. G., Luan, F., & Song, X. (2016). Counting people by using a single camera without calibration. 2016 Chinese Control and Decision Conference (CCDC), 2048–2051. https://doi.org/10.1109/ccdc.2016.7531321

Chen, L., Ali Babar, M., & Nuseibeh, B. (2013). Characterizing Architecturally Significant Requirements. IEEE Software, 30(2), 38–45. https://doi.org/10.1109/ms.2012.174

Chen, T., Chen, T., & Chen, Z. (2006). An Intelligent People-Flow Counting Method for Passing Through a Gate. 2006 IEEE Conference on Robotics, Automation and Mechatronics, 1–6. https://doi.org/10.1109/ramech.2006.252623

Chen, T.-Y., Chen, C.-H., Wang, D.-J., & Kuo, Y.-L. (2010). A People Counting System Based on Face-Detection. 2010 Fourth International Conference on Genetic and Evolutionary Computing, 699–702. https://doi.org/10.1109/icgec.2010.178

Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V.-K., & Strintzis, M. G. (2005). Knowledge-assisted semantic video object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10), 1210–1224. https://doi.org/10.1109/tcsvt.2005.854238

Pore, S. D., & Momin, B. F. (2016). Bidirectional people counting system in video surveillance. *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 724–727. https://doi.org/10.1109/rteict.2016.7807919

Rossi, M., & Bozzoli, A. (1994). Tracking and counting moving people. ***Proceedings of 1st International Conference on Image Processing*****, 3, 212–216. https://doi.org/10.1109/icip.1994.413857**

Saxena, S., & Songara, D. (2017). Design of people counting system using MATLAB. ***2017 Tenth International Conference on Contemporary Computing (IC3)*****, 1–3. https://doi.org/10.1109/ic3.2017.8284344**

Tokta, A., & Hocaoglu, A. K. (2018). A Fast People Counting Method Based on Optical Flow. ***2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*****, 1–4. https://doi.org/10.1109/idap.2018.8620838**

Wahyuni, E. S., Alinra, R. R., & Setiawan, H. (2017). People counting for indoor monitoring. ***2017 International Conference on Computing, Engineering, and Design (ICCED)*****, 1–5. https://doi.org/10.1109/ced.2017.8308112**

Yam, K.-Y., Siu, W.-C., Law, N.-F., & Chan, C.-K. (2011). Effective bi-directional people flow counting for real time surveillance system. ***2011 IEEE International Conference on Consumer Electronics (ICCE)*****, 863–864. https://doi.org/10.1109/icce.2011.5722907**

COCO - Common Objects in Context. (n.d.). Retrieved March 22, 2020, from cocodataset.org website: http://cocodataset.org/#home

Kaeli, D., Mistry, P., Schaa, D., & Ping Zhang, D. (2015). ***Heterogeneous Computing with OpenCL 2.0*****. ScienceDirect. https://www.sciencedirect.com/book/9780128014141/heterogeneous-computing-with-opencl-2-0**

García, J. (2018, March 27). People Counting with OpenCV, Python & Ubidots. Retrieved from Ubidots Blog website: https://ubidots.com/blog/people-counting-with-opencv-python-and-ubidots

Instructables. (2017, November). Person Counting System Using Opencv and Python. Retrieved from Instructables website: https://www.instructables.com/id/Person-Counting-System-Using-Opencv-and-Python

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., … Dolí, P. (2015). Microsoft COCO: Common Objects in Context. Retrieved from https://arxiv.org/pdf/1405.0312.pdf

OpenCV. (2012, November 4). Retrieved April 12, 2020, from Wikipedia website: https://en.wikipedia.org/wiki/OpenCV

OPENCV About. (2018). Retrieved March 26, 2020, from Opencv.org website: https://opencv.org/about/

OpenCV Structure and Content (Overview). (2010, May 23). Retrieved March 26, 2020, from Learning OpenCV website: https://learningopencv.wordpress.com/2010/05/23/opencv-structure-and-content-overview/

Single shot object detection SSD using MobileNet and OpenCV. (2020, January 12). Retrieved April 12, 2020, from Honing Data Science website: https://honingds.com/blog/ssd-single-shot-object-detection-mobilenet-opencv/

Features - PyCharm. (n.d.). Retrieved June 14, 2020, from JetBrains website: https://www.jetbrains.com/pycharm/features/