

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**MOBİL CİHAZLAR İLE DERİN ÖĞRENME MİMARİSİ KULLANARAK
GERÇEK ZAMANLI VİDEO NESNE İZLEME**

YÜKSEK LİSANS TEZİ

Ulaş TOSUN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Programı

AĞUSTOS 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**MOBİL CİHAZLAR İLE DERİN ÖĞRENME MİMARİSİ KULLANARAK
GERÇEK ZAMANLI VİDEO NESNE İZLEME**

YÜKSEK LİSANS TEZİ

**Ulaş TOSUN
504161359**

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Telekomünikasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Bilge GÜNSEL

AĞUSTOS 2019

İTÜ, Fen Bilimleri Enstitüsü'nün 504161359 numaralı Yüksek Lisans Öğrencisi Ulaş TOSUN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “MOBİL CİHAZLAR İLE DERİN ÖĞRENME MİMARİSİ KULLANARAK GERÇEK ZAMANLI VIDEO NESNE İZLEME” başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Prof. Dr. Bilge GÜNSEL KALYONCU**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Doç. Dr. Yusuf YASLAN**
İstanbul Teknik Üniversitesi

Dr. Öğretim Üyesi Tuba AYHAN
MEF Üniversitesi

Teslim Tarihi :
Savunma Tarihi :





Sevgili aileme,



ÖNSÖZ

Gelişen teknoloji ile birlikte mobil cihazlar ve bu mobil cihazlarla birlikte video gibi çoklu medya öğeleri günlük yaşantımızda daha çok yer almaya başlamıştır. Bu sebeple çoklu medya öğelerinin işlenmesi ve anlamlandırılması git gide önem kazanmaktadır. Özellikle görüntülerdeki nesnelerin bulunması ve takibi başlı başına bir araştırma konusu haline gelmiştir. Bu tez çalışmasında, yapay zekâ desteğiyle birlikte videolardaki nesnelerin bulunması, izlenmesi ve bu sürecin mobil cihazlar üzerinde gerçekleştirilmesi sağlanmıştır.

Başta tez çalışmam olmak üzere bütün yüksek lisans hayatım boyunca bana yol gösteren ve desteklerini esirgemeyen danışmanım Prof. Dr. Bilge Günsel'e sonsuz teşekkürlerimi sunuyorum. Performans raporlama metriklerini hesaplama kodunu paylaşan Müh. M. Bünyamin Sağman'a katkılarından ötürü teşekkür ederim. Ayrıca sevgili dostum Araş. Gör. Osman Can Soygenç'e tez çalışmam boyunca verdiği her türlü manevi destekten ötürü bir teşekkürü bir borç bilirim.

Ayrıca, doğduğum günden bu yana arkamda sarsılmaz bir dağ misali durarak hayata güvenle bakmamı ve bu günlere gelmemi sağlayan aileme desteklerinden ötürü tüm kalbimle teşekkür ediyorum.

Bu tez çalışması İTÜ Çoğulortam İşaret İşleme ve Örüntü Tanıma laboratuvarının olanakları kullanılarak yapılmıştır ve MYL-2018-41407 numaralı ve "Derin Öğrenme ile Gerçek Zamanlı Video Takibi" başlıklı BAP projesi tarafından desteklenmiştir. Desteklerinden ötürü BAP birimine teşekkürü bir borç bilirim.

Mayıs 2019

Ulaş TOSUN
Elektronik ve Haberleşme Mühendisi



İÇİNDEKİLER

Sayfa

ÖNSÖZ	vii
İÇİNDEKİLER	ix
KISALTMALAR	xi
ÇİZELGE LİSTESİ.....	xiii
ŞEKİL LİSTESİ.....	xv
ÖZET	xvii
SUMMARY	xix
1. GİRİŞ	1
1.1 Tezin Amacı	1
1.2 Literatür Araştırması	2
1.2.1 Çift aşamalı yöntemler	2
1.2.2 Tek aşamalı yöntemler	3
1.2.3 CNN omurga ağ yapıları	4
1.3 Hipotez	5
2. MOBILENET VE SSD	7
2.1 MobileNet.....	7
2.1.1 Derinlemesine ayrılabilen konvolüsyon	7
2.1.2 MobileNet'in ağ mimarisi	11
2.2 SSD.....	13
2.2.1 SSD'nin ağ mimarisi	13
2.2.2 SSD'nin eğitimi ve detayları	15
2.3 SSD'nin MobileNet mimarisi ile birleştirilmesi	17
3. MOBİL KAMERA HAREKETİNİN MODELLENMESİ	23
3.1 İvmeölçer Verisinden Mobil Kamera Hızının Elde Edilmesi	23
3.2 Afın Kamera Hareketi Parametrelerinin Hesaplanması	29
3.3 Hesaplanan Kamera Hareket Parametrelerinin Kestirim ile Doğrulanması	30
4. İVMEÖLÇER GERİBESLEMESİ İLE SSD NESNE İZLEME	37
4.1 Kamera Hareketi Kullanılarak Nesne İzleyici Performansının Arttırılması	37
4.2 Önerilen Yöntemin Android Ortamında Gerçeklenmesi.....	42
5. PERFORMANS TEST SONUÇLARI	45
5.1 Eğitilmiş Model Dosyası İle Eğitim Seti Üzerinde Elde Edilen Başarım.....	45
5.2 VOT Videolarında Nesne İzleme Performansı	54
5.3 Cep Telefonu İle Çekilen Videolarda Performans	56
5.4 Gerçeklemenin Nesne İzleme Hızı.....	66
6. SONUÇLAR VE ÖNERİLER	69
KAYNAKLAR	71
ÖZGEÇMİŞ	75



KISALTMALAR

CNN	: Convolutional Neural Network
SGD	: Stochastic Gradient Descent
SVM	: Support Vector Machine
SVD	: Singular Value Decomposition
RPN	: Region Proposal Network
NMS	: Non-Maxima Supression
RoI	: Region of Interest
mAP	: mean Average Precision
IoU	: Intersection over Union
BN	: Batch Normalization
ReLU	: Rectified Linear Unit
BB	: Bounding Box



ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : MobilNetv1 ağ mimarisi [14].....	12
Çizelge 2.2 : Kesilmiş MobilNetv1 ağ mimarisi.	19
Çizelge 2.3 : Kesilmiş MobilNetv1 ağ mimarisi (Çözünürlük çarpanı = 1.34).....	19
Çizelge 2.4 : Kesilmiş MobilNetv1 ağ mimarisi (Çözünürlük çarpanı = 1.34, genişlik çarpanı = 2)..	20
Çizelge 2.5 : SSD ile birleştirilmiş MobileNetv1 mimarisi.....	21
Çizelge 2.6 : SSD'nin farklı temel CNN mimarileri kullanılarak hazırlanmış sürümlerinin karşılaştırılması [14].....	21
Çizelge 5.1 : VOT 2016 veritabanı nesne izleme sonuçları.	56
Çizelge 5.2 : Oluşturulan veri tabanındaki videolar ve özellikleri.	57
Çizelge 5.3 : 31 karelik örnek videoya ait detaylı sonuçlar.....	60
Çizelge 5.4 : Oluşturulan veri tabanı üzerinde elde edilen sonuçlar.	63



ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : R-CNN'in yapısı [2].....	2
Şekil 1.2 : Fast R-CNN'in yapısı [4].	3
Şekil 1.3 : Nesne tespit yöntemlerinin karşılaştırılması [25].....	5
Şekil 2.1 : Standart konvolüsyon filtreleri [22].	7
Şekil 2.2 : Sırasıyla derinlemesine ayrılabilen konvolüsyon filtreleri ve noktasal konvolüsyon filtreleri [22].	8
Şekil 2.3 : Standart konvolüsyonun detayları [26].	9
Şekil 2.4 : Derinlemesine ayrılabilen konvolüsyonun detayları [26]..	9
Şekil 2.5 : Noktasal konvolüsyonun detayları [26]..	10
Şekil 2.6 : Solda standart konvolüsyonun, sağda ise derinlemesine ayrılabilen konvolüsyonun ReLU ve BN katmaları ile birlikte kullanımı [22]..	11
Şekil 2.7 : SSD'nin nesne tanıma yapısı [13]..	13
Şekil 2.8 : SSD ve YOLO v1'in ağ mimarilerini karşılaştırılması [13].	14
Şekil 2.9 : SSD mimarisinin MobileNet mimarisi ile birleştirilmesi [13].....	18
Şekil 3.1 : Android telefonlar tarafından kullanılan ivmeölçerlerin eksen doğrultuları [27].....	24
Şekil 3.2 : Yerçekimi ivmesinin farklı eksenlere etkisi [28].	24
Şekil 3.3 : (a) Sabit bir cihaza ait filtrelenmiş ivme verisinin zamanla değişimi. (b) Aynı ivme verisinin histogramı.	26
Şekil 3.4 : Ölçümlerle hesaplanan kamera ve izdüşen bir imge pikselinin hareketlerinin zaman içerisindeki değişimi.	28
Şekil 3.5 : Yerçekimi ivmesinin ilerleyen video karelerinde yarattığı bozucu etki... ..	28
Şekil 3.6 : Temel kamera hareketleri[30].	29
Şekil 3.7 : İki ardışık video karesi arasındaki eşleştirilmiş köşeler.....	33
Şekil 3.8 : Örnek bir video için hesaplanan kamera hareketi parametreleri.....	35
Şekil 3.9 : Köşe eşleştirme ve ivmeölçer kullanılarak hesaplanan kamera yer değiştirme hareketi parametresinin video boyunca değişimi.....	35
Şekil 3.10 : İki farklı yöntemle hesaplanan kamera hareketi parametrelerinin eş zamanlı çizimi.	36
Şekil 4.1 : Nesne izleme performansının artırılmasına dair akış diyagramı.....	38
Şekil 4.2 : Örnek bir videonun her bir karesindeki nesnelilik skoru ve örtüşme oranı.	39
Şekil 4.3 : Örnek bir videonun her bir karesi için hesaplanan iki konum tahmininin birbiri ile olan örtüşme oranı.	40
Şekil 4.4 : Önerilen yöntemin nesne izleme performansını gösteren bir örnek.....	40
Şekil 4.5 : Örnek bir videonun 12, 13, 14 ve 15. kareleri.....	41
Şekil 4.6 : Geliştirilen Android uygulamanın çalışırken görüntüsü.	42
Şekil 4.7 : Geliştirilmesi hedeflenen Android uygulamasına ait sözde kod.....	43
Şekil 5.1 : MS COCO 2014 veri setinden rastgele seçilmiş 4 resim üzerindeki nesne tespit sonuçları.	46

Şekil 5.2 : MS COCO 2014 eğitim setinde MobileNetv1 SSD'nin sınıflara göre sağladığı "recall" değerleri.....	48
Şekil 5.3 : MS COCO 2014 doğrulama setinde MobileNetv1 SSD'nin sınıflara göre sağladığı "recall" değerleri.....	49
Şekil 5.4 : MS COCO 2014 eğitim setinde MobileNetv1 SSD'nin sınıflara göre sağladığı "precision" değerleri.....	50
Şekil 5.5 : MS COCO 2014 doğrulama setinde MobileNetv1 SSD'nin sınıflara göre sağladığı "precision" değerleri.....	51
Şekil 5.6 : MS COCO 2014 eğitim setinde MobileNetv1 SSD'nin sınıflara göre sağladığı F1 skoru.	52
Şekil 5.7 : MS COCO 2014 doğrulama setinde MobileNetv1 SSD'nin sınıflara göre sağladığı F1 skoru.	53
Şekil 5.8 : MS COCO 2014 veri seti için genelleştirilmiş sonuçlar.	54
Şekil 5.9 : VOT 2016 veri setindeki bazı videoların (a) doğruluk değerleri, (b) dayanıklılık değerleri.	55
Şekil 5.10 : Oluşturulan veri setinden alınmış örnek video kareleri.....	57
Şekil 5.11 : MobileNet SSD ile tespit edilen nesne BB'leri ve nesnelilik skorları. ..	58
Şekil 5.12 : 31 karelik örnek videodan bazı video kareleri.	59
Şekil 5.13 : MobileNet SSD + İvmeölçer ile 5. ve 26. video karelerinde yapılan nesne tespiti.....	60
Şekil 5.14 : MobileNet SSD (yeşil), ivmeölçer geri beslemesinin katkısı (sarı), kaçırılanlar (kırmızı) (a) Merkez piksel kayması eşiği 40 piksel. (b) IoU eşiği 0.5.	61
Şekil 5.15 : Sabit nesnen hareketli kamera durmuna ait video kareleri.....	62
Şekil 5.16 : Örnek bir videoya ait piksel cinsinden kamera hareketi kompanze edilmiş ve edilmemiş nesne hareketi.	62
Şekil 5.17 : Veritabanındaki videolardan örtüşme durumuna örnek oluşturan video kareleri.	64
Şekil 5.18 : Veritabanındaki videolardan poz değişimi durumuna örnek oluşturan video kareleri.	65
Şekil 5.19 : Veritabanındaki videolardan bulanıklık durumuna örnek oluşturan video kareleri.	65
Şekil 5.20 : Veritabanındaki 10. videoya ait 6 video karesi.	66
Şekil 5.21 : İki farklı SSD mimarisinin hız yönünden karşılaştırılması.	67

MOBİL CİHAZLAR İLE DERİN ÖĞRENME MİMARİSİ KULLANARAK GERÇEK ZAMANLI VİDEO NESNE İZLEME

ÖZET

Yeni gelişen teknolojilerle ve elektronik donanımlardaki gelişmelerle birlikte başta akıllı telefonlar olmak üzere mobil cihazların performansı ve buna bağlı olarak da kullanım alanları gitgide artmaktadır. Günümüzde özellikle akıllı telefonlar en az bilgisayarlar kadar çok kullanılmaktadır ve insanoğlunun günlük yaşantısının vazgeçilmez bir unsuru haline gelmiştir.

Öte yandan literatürdeki pek çok nesne izleme ve nesne tespit yöntemi sadece bilgisayarlar üzerinde kullanıma uygun olacak şekilde tasarlanmış olup işlem gücü nispeten daha düşük cihazlar olan mobil cihazlar üzerinde kullanılmaya uygun değildir. Bunun bir sonucu olarak nesne tespit ve izleme alanında geliştirilen pek çok yöntem akıllı telefonlarda kullanılamamaktadır. Bu durum nesne izleme algoritmalarının birçok günlük uygulamada kullanılmasını engellemektedir.

Bu tez çalışmasında, mobil cihazlarda çalışmaya uygun, düşük işlem gücüne ihtiyaç duyan MobileNet SSD nesne tespit algoritması kullanılarak, videolarda gerçek zamanlı nesne izleme yapabilecek bir mobil uygulama hedeflenmiştir. Mobil cihazlarda çalışmanın bir avantajı bilgisayarlarda bulunmayan çeşitli algılayıcılara sahip olmalarıdır. İvmeölçer, ışık algılayıcısı, jiroskop ve GPS gibi örnekendirilebilecek olan bu algılayıcılar aynı zamanda nesne izleme performansını arttırmak amacıyla da kullanılabilirler. Bu tez çalışmasında, nesne izleme performansının artırılabilmesi amacıyla, ivmeölçerden alınan veri kullanılarak hedef izleme gerçekleştirilmiştir. Bu kapsamda öncelikle ivmeölçerden gelen veri çeşitli frekans karakteristiğine sahip gürültülerden arındırılmıştır. Ardından elde edilen anlamlı ivme verisi hız verisine dönüştürülmüştür. Kameranın üzerinde bulunduğu mobil cihaza ait olan hız verisi kullanılarak, kamera hareketi modellenmiştir. Bu sayede hareketli kamera durumunda nesne izleme performansının artırılması hedeflenmiştir. Çalışmada SSD MobileNet, özellikle hız yönünden benzerlerinden olumlu anlamda ayrılması ve mobil cihazlarda gerek duyulan hızlarda çalışabilmesi nedeniyle tercih edilmiştir.

Kamera hareketinin doğrulukla hesaplanabildiği bir ortamda nesne izleme performansını arttırmak amacıyla çok çeşitli yöntemlere başvurulabilir. Bu tez çalışmasında öncelikle hesaplanan kamera hareketinin nesne hareketi üzerindeki etkisi kompanse edilerek gerçek nesne hareketi modellenmiştir. Ardından elde edilen nesneye ilişkin hareket bilgisi SSD MobileNet nesne tespit algoritmasıyla birleştirilerek nesne izleme performansı artırılmıştır.

Tez çalışmasında SSD MobileNet derin öğrenme ağı yeniden eğitilmemiş, kullanıma açık olan ve MS COCO veri seti üzerinde eğitilmiş, 80 farklı nesneyi tespit edebilen bir MobileNet SSD eğitim modeli kullanılmıştır. Öncelikle MS COCO eğitim seti üzerinde nesne tespit performansı incelenmiş, ardından literatürde sıklıkla kullanılan

VOT 2016 video veri tabanındaki başarımlar raporlanmıştır. Ayrıca tez çalışması için özel olarak hazırlanan ve mobil telefon ile günlük hayatın içinden çekilmiş videolar içeren özel bir veri seti üzerinde de sonuç gösterilmiştir. Bunun yanı sıra bir mobil nesne sezme ve izleme uygulaması Android ortamında gerçekleştirilerek başarımlar raporlanmıştır.

Sonuç olarak bu tez çalışmasında öncelikle, bilgisayarlar üzerinde kullanılan nesne tespit ve izleme yöntemlerinin başta akıllı telefonlar olmak üzere mobil cihazlara da entegre edilerek günlük hayatın içerisinde daha fazla yer alabileceğinin gösterilmesi hedeflenmiştir. Ayrıca mobil cihazlar üzerinde çalışmanın en önemli kazanımlarından birisi olan algılayıcılardan, özel olarak ivmeölçerden, alınan veri kullanılarak hareketli kamera ile izlenen nesnelerin tespitinde performansın artırılabilmesi gösterilmiştir.



DEEP LEARNING ARCHITECTURES FOR REAL-TIME VIDEO OBJECT TRACKING BY MOBILE DEVICES

SUMMARY

With the advances in new technologies and developments in electronic hardware, the performance of mobile devices, especially smartphones, and their usage areas are increasing. Nowadays, especially smart phones have been used as much as computers and have become an indispensable element of human life. According to market share data, people use smartphones more than computers for accessing to internet and also most of the video data have been capturing by smartphones.

In addition, manufacturers improves processing capacity of modern smartphones in every year. Smartphones, especially flagship models, have sufficient processing power to run simple neural networks that are used by mobile applications. Despite to improvements on their hardware, computational power of mobile devices is still less than computers that have a modern graphical processing unit in terms of running complex neural networks such as CNNs. On the other hand, complex neural networks such as CNNs are used by mobile applications for various tasks. For example, camera applications detect faces for face beauty perfection. They classify scenes and detects objects to optimize camera parameters and capture better images.

However, most of the existing state-of-the-art object tracking and object detection methods such as Mask R – CNN and Retina Net are designed for use only on computers and are not suitable for use on mobile devices because of their demanding processing units. Consequently, most of the research efforts on video object tracking and detection cannot be adopted to mobile devices although they are more common in everyday life than computers. This prevents development of the applications that employ object detection and tracking to find more space in daily life. In order to overcome these limitations, MobileNet SSD is introduced as a real-time object detection algorithm. MobileNet SSD architecture includes an object detection network called SSD located at the top of a backbone neural network called MobileNet. MobileNet SSD is capable of working in real-time on mobile devices, because its CNN layers perform a special type of convolution called depth wise separable convolution. This is why we prefer MobileNet SSD as the baseline algorithm within the context of this thesis. YOLO is another deep network architecture designed to speed up the object detection. It is reported that YOLOv3 is capable of real-time object detection. Despite their speed, neither MobileNet SSD nor YOLOv3 provide the detection accuracy as high as the conventional deep object detectors. This is mainly because both of them pass over the input image once that significantly reduces the computational complexity. However, as it is expected, there is a tradeoff between the detection accuracy and the computational complexity. Recently a number of new deep architectures including lite R-CNN are introduced to improve accuracy while decreasing the computational complexity.

In the context of this thesis, we also aim to improve the accuracy of MobileNet SSD without increasing the computational load. Unlike the existing approaches, we propose to employ a number of different measurements for object detection along with the video data extracted by the mobile camera. Because, one of the positive things about working on mobile devices is that many of the mobile devices have various sensors that are not available on computers. In particular, we propose usage of these sensors, such as accelerometers, light sensors, gyroscopes and GPS to improve object tracking performance. The combination of image data and environmental sensor data is also proposed by some other application areas such as autonomous cars, drones and other robotic applications.

To improve object tracking performance various environmental sensors could be used at the same time or standalone. However, using multiple sensor results with many different problems such as noise and synchronization. In order to simplify the real-time object monitoring, in the context of the thesis, it is decided to use only the accelerometer in order to increase object tracking performance in case of moving camera. In particular acceleration measurements taken from the accelerometer can be used for the calculation of velocity. In this context, the data obtained from the accelerometer is filtered and the noise with various frequency characteristics is eliminated. It is observed that the acceleration data is distorted by two types of noise. The first one is measurement noise eliminated by low pass filtering because it has higher frequency components. The second one is gravity effects having a lower frequency characteristic thus eliminated by high pass filtering. After noise elimination, it can be reached to the velocity data by applying a discrete integral operation on the measurements. We propose to use the velocity to extract the camera motion under the assumption that the mobile device and its camera move synchronized. This is followed by the affine motion parameter estimation that provides the motion parameters of mobile camera and also allows us to classify the camera motion as pedestal, track, etc. for different semantic indexing applications. It is possible to reach actual object's motion, if it is desired, by eliminating the camera motion but conventionally the object tracking deals with the estimation of the new object location with respect to the previous frame rather than estimating actual object motion. The proposed object tracking method employs MobileNet SSD guided by the motion estimated from the accelerator measurements. It is shown that fusing the sensor measurements into the deep object detector increases the tracking accuracy especially under motion blur and severe illumination changes where the deep detector fails to track.

In addition, since the camera motion parameters are estimated by using the velocity data, the accuracy of these estimations is tested by using a feature based motion estimation model. The model detects the corner points on the input video frame by using Harris corner detector. Then, for each corner point, FREAK descriptor assigns a binary feature vector. This is followed by matching the corner points of successive video frames by minimizing Hamming distance between the descriptors. Finally, under the assumption of 2D affine motion, affine motion parameters corresponding the mobile camera motion are estimated by a least squares estimator. In this work, it is shown that the camera motion parameters calculated by the feature based estimation and by using the accelerator measurements are very close, that confirms the accuracy of our measurements.

The stand-alone object detection performance of MobileNet SSD is evaluated on MS COCO 2014 dataset, since the deep detector is trained on this data set. The idea behind this is to report the highest detection accuracy achieved by the employed deep object

detector. Accuracy is reported by precision, recall and F1 measure metrics, which are commonly used in object detection benchmarking challenges. In order to observe the improvement achieved by MobileNet SSD guided by the accelerometer measurements, the object tracking performance is reported on VOT 2016 dataset that is a commonly used benchmarking data set in object tracking. Accuracy and robustness metrics are reported for each video sequence as well as the mean obtained on the full data set. Finally, object tracking performance achieved on a special dataset that is prepared for this thesis study is evaluated and reported. The dataset contains daily life videos captured by a smartphone in different scenarios.

In addition, this method was investigated in terms of speed compared to similar methods. This comparison is supported with measurements on mobile devices and results have been reported.

In addition to all, 3 Android applications were written for this thesis. These applications were prepared by using Android Studio environment. OpenCV Android library was used for developing these applications and implementation details was reported. One of these applications is the main application and it runs on real time. Other two run offline and they were developed for offline tests. Matlab is the environment which is used for result analysis and plotting data.

As a result, in this thesis, it is aimed that object detection and tracking methods used on computers in general are integrated into mobile devices, especially smart phones, which are more common in daily life than computers. In addition, thanks to sensors, one of the most important gains of working on mobile devices, it is aimed to increase object tracking performance in case of moving camera. All these goals have been reached under various assumptions and details are included in the thesis study.

In this study, the other details of this method such as the combination of SSD and MobileNet and its principles were given in detail. In addition, other modern methods have been mentioned under the title of literature study. Also, the method used in the thesis study has been compared with the similar ones in literature in terms of both speed and accuracy. Organization of thesis is as follows. Firstly, an introduction and literature survey is given in Chapter 1. Then, SSD and MobileNet is discussed in detail in Chapter 2. Camera motion analysis is explained in Chapter 3 and combination of motion data and MobileNet SSD is explained in Chapter 4. After, test result for different data sets and conditions is given in Chapter 5. Finally, the conclusion is given in Chapter 6.

1. GİRİŞ

Gelişen teknoloji ile birlikte başta akıllı telefonlar olmak üzere mobil cihazlar günlük hayatımıza hızla yerleşmektedir. Her yıl dünya genelinde yüz milyonlarca akıllı telefon yeni sahipleri ile buluşmaktadır. Artan mobil cihaz sayısının bir sonucu olarak resim ve video gibi çoklu medya öğeleri de hızla günlük hayatımıza girmiştir. Geldiğimiz noktada insanoğlu her gün çok büyük boyutlarda video içeriği üretmektedir. 2021 yılı itibariyle dünya genelindeki internet trafiğinin %82'sinin videolar tarafında oluşturulması beklenmektedir [1]. Bu kadar çok videonun anlamlandırılması ve işlenmesi ise başlı başına bir problemdir. Videoların anlamlandırılmasında videodaki nesnelerin tespit edilmesi ve hareketlerinin izlenmesi gerek duyulan temel bir işlemdir. Videolarda nesne tespiti ve izlenmesi için literatürde pek çok yöntem geliştirilmiş olup bunların pek azı mobil cihazlarda çalışmaya uygundur. Buradaki zorluk mobil cihazların bilgisayarlara kıyasla daha düşük işlem gücüne ve depolama kapasitesine sahip olmasıdır. Bir diğer önemli zorluk, mobil uygulamaların çevrimiçi ve olabiliyorsa gerçek zamanlı işlem gereksinimidir. Bu sebeple son dönemde mobil cihazlarda da başarı ile çalışan nesne izleme yöntemleri geliştirilmeye çalışılmaktadır.

1.1 Tezin Amacı

Mobil cihazların ivmeölçer ve jiroskop gibi cihazın hareketini algılayabilen algılayıcılara sahip olması, bu algılayıcılardan gelen verinin anlamlandırılması ve kameradan gelen görüntü verisi ile birleştirilmesi sayesinde nesne izleme sistemlerinin hassasiyeti arttırılabileceği görüşünü oluşturmuştur. Bu doğrultuda tez çalışması kapsamında mobil cihaz ivmeölçer algılayıcısından alınan veri kullanılarak nesne izleme performansının arttırılması hedeflenmiştir. Bu amaçla MobileNet SSD nesne sezici olarak kullanılarak takip edilmek istenen nesnenin konumu hakkında bir tahmin elde edilmiştir. Paralelde ivmeölçerden gelen veri kullanılarak önce kamera hareketi parametreleri kestirilmiş, ardından kamera hareketi kompanze edilerek nesnenin konumu hakkında ikinci bir tahminde bulunulmuştur. Daha sonra bu iki tahmin nesne

skoru ve örtüşme oranı kriter olarak kullanılarak birleştirilmiş ve nesne izleme performansı arttırılmıştır.

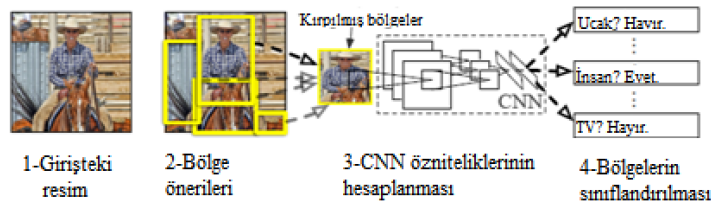
1.2 Literatür Araştırması

Tez çalışması kapsamında önerilen nesne izleyici ivmeölçer ile elde edilen kamera hareketi verisinin nesne sezici ile birleştirilmesiyle elde edilmiştir. Nesne sezici olarak bir nesne tespit yöntemi kullanılmıştır. Bu sebeple literatür araştırması bölümünde literatürdeki nesne tespit yöntemleri ve bu yöntemlerle birlikte kullanılan omurga ağ yapıları incelenmektedir.

Görüntüde nesne tespiti hakkındaki çalışmalar yıllar öncesine dayanmaktadır. Ancak derinlikli ağ mimarileri kullanılan güncel metotlar kendi içlerinde çift aşamalı ve tek aşamalı yöntemler olarak ikiye ayrılabilir. Çift aşamalı yöntemler olası nesne pozisyonlarını ve bu pozisyonların sınıflandırılmasını iki farklı aşamada yaparken tek aşamalı yöntemler bu iki işlemi birlikte yapar. Bu yöntemlerin ortak özellikleri ise hepsinin CNN kullanmasıdır.

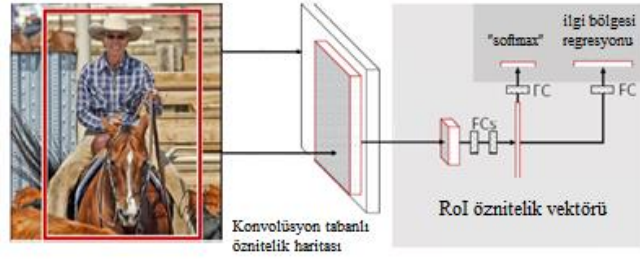
1.2.1 Çift aşamalı yöntemler

Günümüzdeki neredeyse tüm modern çalışmaların temeli sayılabilecek olan R-CNN ağ mimarisi [2] çift aşamalı bir nesne sezme yöntemidir. Şekil 1.1’de de görüldüğü gibi, R-CNN önce seçici-arama [3] kullanarak olası nesne konumlarını tespit eder. Daha sonra bu olası konumları iki sınıflı doğrusal SVM kullanarak nesne tipi bazında sınıflandırır ve ilgili sınıfa özel regresyon ile nesneyi çevreleyen kutunun (BB) konumunun doğruluğunu arttırır.



Şekil 1.1 : R-CNN'in yapısı [2].

R-CNN'in geliştirilmesi sonucunda oluşturulan Fast R-CNN [4] Şekil 1.2’de de görüldüğü gibi sınıflandırma için doğrusal SVM yerine “softmax” kullanarak doğruluğu arttırmıştır. Ayrıca, SVD kullanarak yöntemin hızında da %30 artış sağlamıştır.



Şekil 1.2 : Fast R-CNN'in yapısı [4].

Fast R-CNN de tıpkı R-CNN gibi olası nesne konumlarını seçici arama kullanarak belirler. Ancak, Faster R-CNN [5] seçici arama yerine RPN kullanarak hem hızı hem de doğruluğu arttırmıştır.

Öte yandan Faster R-CNN de tıpkı kendinden önceki yöntemler gibi her bir nesne aday bölge için tek tek işlem yapmaktadır ve bu durum işlem yükünü arttırmaktadır. Bu yapıya eklenen piramit mimarisi, R-FCN [6], sayesinde pozisyona duyarlı skor haritaları kullanılarak neredeyse tüm hesaplama paylaşımlı yapılabilmiş bu sayede hız önemli oranda arttırılmıştır. Faster R-CNN'e obje bazında bölütleme sonucu üreten ikinci bir ağ kolu eklenerek Mask R-CNN [7] geliştirilmiş ve nesne sezme başarımı yükseltilmiştir.

İki aşamalı yöntemler, gövdeyi oluşturan omurga ağ ve öneri ağı ile üst katman olarak yerleştirilmiş baş ağ yapısı olmak üzere, üç farklı katman grubunu birlikte kullanır. Omurga ağ giriş görüntülerinden öznetelik haritalarını çıkarır. Öneri ağı (RPN) özneteliklerden olası nesne pozisyonlarını üretirken baş kısmı ise bu pozisyonları nesne etiketlerine göre sınıflandırır. RPN sayesinde gövde kısımları genel olarak hızlı çalışmaktadır ancak baş kısımları nispeten yavaştır. Light-Head R-CNN [8] mimarisi derinlemesine ayrılabilen konvolüsyon kullanarak ve RoI bükme yaparak kafa kısmını hızlandırmış ve aynı zamanda doğruluğu da arttırmıştır.

1.2.2 Tek aşamalı yöntemler

Tek aşamalı yöntemler, çift aşamalı yöntemlere göre doğaları gereği genel olarak daha hızlıdır. Tek aşamalı nesne tespit yöntemlerinde YOLO v1 [9] DarkNet mimarisini kullanır ve türünün başarılı örneklerindendir. Bu ağ mimarisinin geliştirilmesi ile sırasıyla YOLO v2 [10] ve YOLO v3 [11] olmak üzere iki yeni mimari önerilmiştir.

Bir diğer tek aşamalı yöntem olarak Multibox [12] tabanlı SSD veya tam adıyla "Single Shot Multibox Detector" [13] önerilmiştir. Bu tez çalışmasında

MobileNetv1[14] omurga ağ mimarisi ile tümleştirilmiş SSD ağ yapısı derin nesne sezici olarak kullanılmıştır. Amaçlanan doğruluk-hız ödünleşimini dengeleyen güncel bir ağ mimarisinin kullanımıdır. MobileNetv1 adından da anlaşılacağı üzere özellikle mobil cihazların sunduğu nispeten düşük işlem gücüne sahip ortamlarda hızlı bir şekilde çalışacak şekilde tasarlanmış özel bir CNN mimarisidir [14]. Küçük ve yüksek hızlı CNN'ler tasarlayabilmek amacıyla daha önce Inception [15] ve Xception [16] gibi başka mimariler de geliştirilmiştir, bu bakımdan MobileNetv1 mobil cihazları hedefleyen ilk mimari değildir. Literatürde ise genel olarak hızlı ve verimli ağlar elde etmek için iki farklı yöntem kullanılır, bunlar önceden eğitilmiş ağı sıkıştırmak veya direkt olarak küçük bir ağı eğitmek üzerinedir. MobileNet ikinci gruba dâhildir yani kendisi direkt olarak küçük boyutlu bir ağ olarak tasarlanmıştır. Ayrıca literatürdeki pek çok yöntem sadece ağ boyutunu küçültmeyi hedeflerken MobileNet ağı hem küçültmeyi hem de hızlandırmayı hedeflemektedir, bu bakımdan pek çok benzerinden ayrılmaktadır. Bu hedefine ulaşmak için yaptığı en temel değişiklik ise standart konvolüsyon yerine derinlemesine ayrılabilen konvolüsyon işlemini kullanmasıdır.

Tek aşamalı yöntemlerin iki aşamalı yöntemler kadar yüksek doğrulukta nesne tespiti yapamama nedenleri araştırılarak en yüksek doğruluğa sahip tek aşamalı yöntemlerden olan RetinaNet [17] önerilmiştir. Performans düşüklüğünün temel sebebi tek aşamalı yöntemlerin eğitim aşamasında aşırı derecede sınıf dengesizliği yaşaması olarak tespit edilmiştir. İki aşamalı yöntemlerde gövde kısmı arka planın çok büyük bir kısmını elemektedir, bu da sınıfların nispeten dengeli olmasını sağlamaktadır. Oysa tek aşamalı yöntemlerde eğitimde kullanılan verinin çok büyük bir kısmı arka plandan ibarettir ve bu da sınıf dengesizliğine yol açmaktadır. Bu sorunun çözümü için odaklanmış-hata formülasyonu kullanılarak iyi öğrenilmiş örneklerin hata fonksiyonuna daha az katkı yapması, öte yandan kötü öğrenilmiş örneklerin hata fonksiyonuna daha çok katkı yapması sağlanmış ve başarıya ulaşılmıştır.

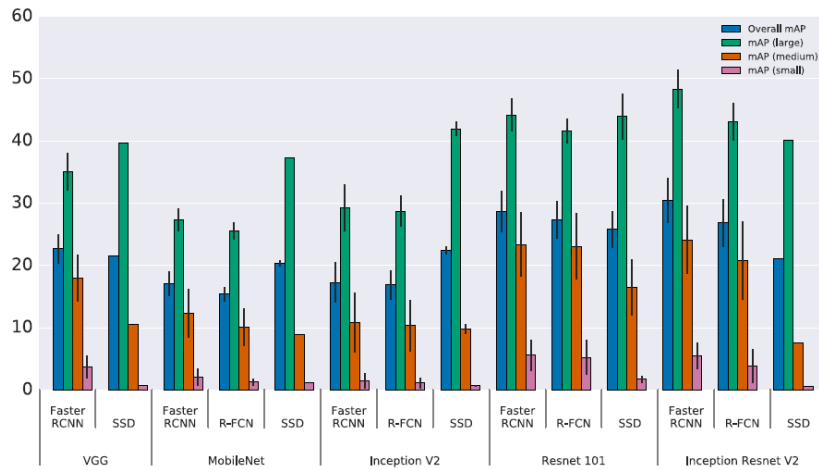
1.2.3 CNN omurga ağ yapıları

Önceki altbölümlerde bahsedilen nesne tespit yöntemleri bazı çok kullanılan CNN omurgalarını temel alarak tasarlanmışlardır. Günümüze kadar VGG [18], SPP-net [19], Inception [15] ve Xception [16] gibi pek çok mimari geliştirilmiş ve kullanılmış olsa da günümüzde en çok kullanılan mimariler ResNet [20] ve FPN'dir [21].

Bu omurgalardan ResNet[20], CNN katmanları arasına kısa devre bağlantıları ekleyerek yeni bir ağ mimarisi tanımlamıştır. Bu yeni mimari sayesinde çok derin ve daha iyi öğrenen CNN'ler tasarlamak mümkün hale gelmiştir. Ayrıca konvolüsyon işlemleri öncesinde mimaride darboğazlar yaratarak hızı da arttırmayı başarmıştır. Daha sonra FPN piramit benzeri CNN mimarileri yaratarak farklı ebattaki nesnelerin tespitini kolaylaştırmıştır.

Ayrıca, mobil cihazlar için özel olarak tasarlanmış omurgalar da mevcuttur. Bunlara ShuffleNet [22] ve MobileNet v1 [14] örnek olarak verilebilir. MobileNetv1 [14] bu tez çalışmasında kullanılan omurgadır ve derinlemesine ayrılabilen konvolüsyon kullanarak CNN mimarisini hızlandırır. Ardından MobileNetv2 [23], MobileNetv1'den [22] farklı olarak ResNet[19] benzeri kısa devre bağlantıları kullanarak hızda artış sağlamıştır. Son olarak, MobileNetv3 [24] omurgayı mobil telefon işlemcileri için optimize ederek hız artışını bir adım ileriye taşımıştır.

Şekil 1.3'te nesne tespit yöntemlerinden bazılarının mAP skoru yönünden farklı büyüklükteki nesneler için karşılaştırması, fikir vermesi amacıyla, gösterilmektedir. Şekilde aynı nesne tespit yöntemiyle birlikte kullanılan farklı omurgaların ilgili nesne tespit yönteminin performansını doğrudan etkilediği görülmektedir.



Şekil 1.3 : Nesne tespit yöntemlerinin karşılaştırılması [25].

1.3 Hipotez

Tez çalışmasında mobil cihazlarda çalışabilecek kadar düşük işlem gücüne ihtiyaç duyan ve mobil cihazlardaki sensörlerden alınan veriyle nesne izleme performansını arttıran bir yöntem önerilmiştir. Bu amaçla benzerlerine göre çok daha düşük işlem gücüne ihtiyaç duyması nedeniyle MobileNetv1 SSD isimli bir nesne tespit yöntemi

nesne sezici olarak kullanılmıştır. Bu nesne sezici yardımıyla izlenmek istenen nesnenin konumu dair tahminlerde (BB) bulunulmuştur. Eş zamanlı olarak neredeyse tüm mobil cihazlarda bulunan ivmeölçer sensöründen gelen veri işlenerek kamera hareketi modellenmiştir. Modellenen kamera hareketi kullanılarak kamera hareketi kompanzasyonu yapmış, izlenmek istenen nesnenin konumu hakkında ikinci bir tahminde (BB) bulunulmuştur. Bu iki farklı tahmin nesne skorunu ve örtüşme oranını karar kriterleri olarak kullanarak birleştirilmiştir.

Test aşamasında öncelikle MobileNetv1 SSD'nin nesne tespit ve izleme performansları ayrı ayrı incelenmiştir. Ardından, bu tez çalışması için özel olarak hazırlanan ve günlük hayatın içinden çekilmiş videolardan oluşan bir video veri tabanı üzerinde bu tez çalışmasında önerilen yöntemin performansı raporlanmıştır. Bu raporlamada önerilen yöntemin özellikle dayanıklılık metriği üzerinde kayda değer bir iyileşme sağladığı görülmüştür. Test aşamasının sonunda önerilen yöntem hız yönünden incelenmiş ve nesne sezici olarak MobileNetv1 SSD kullanmanın doğru bir tercih olduğu gösterilmiştir.

Yine tez çalışması kapsamında ikisi çevrimdışı ve birisi gerçek zamanlı olmak üzere 3 farklı Android uygulaması geliştirilmiştir. Bu uygulamalardan çevrimdışı olanlar önerilen yöntemi tam olarak gerçeklerken gerçek zamanlı olan ise önerilen yöntemi kısmen gerçeklemektedir.

Tez çalışmasının 2. Bölümünde nesne sezici olarak kullanılan MobileNetv1 SSD'nin detaylarına yer verilmektedir. 3. Bölümde ivmeölçerden gelen veri kullanılarak kamera hareketi kestirimi ve kamera hareketi kompanze edilerek nesne konumunun tespitinde kullanılan yöntem anlatılmaktadır. 4. Bölümde dernelikli nesne sezici ve ivmeölçer kararlarının tümleştirilmesi ile nesne tespiti formüle edilmektedir. 5. Bölümde tez çalışması kapsamında önerilen yöntemin performans analizi ve sonuç raporlama yapılmaktadır. Son olarak, 6. Bölümde sonuçlar ve ileriye yönelik öneriler özetlenmektedir.

2. MOBILENET VE SSD

Önceki bölümlerde de bahsedildiği gibi bu tez çalışmasında MobileNetv1 omurga ağ yapısı ile hızlandırılmış bir SSD mimarisi kullanılmaktadır. Bu bölümde önce diğer mimarilere göre neden ve ne kadar hızlı olduğuna değinilerek MobileNetv1'den bahsedilecek, ardından ise SSD'den bahsedilecektir. Son olarak ise MobileNetv1 mimarisinin SSD ile nasıl birleştirildiğine yer verilip bu bölüm sonlandırılacaktır.

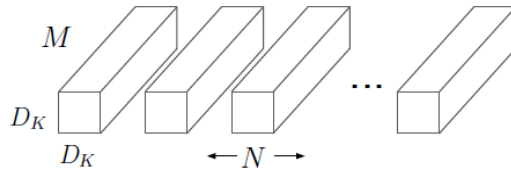
2.1 MobileNet

Derinlemesine ayrılabilen konvolüsyon işlemi MobileNetv1 mimarisinin en can alıcı noktası olduğu için MobileNetv1 mimarisinin detaylarına geçmeden önce derinlemesine konvolüsyon işlemi detaylıca ele alınacaktır.

2.1.1 Derinlemesine ayrılabilen konvolüsyon

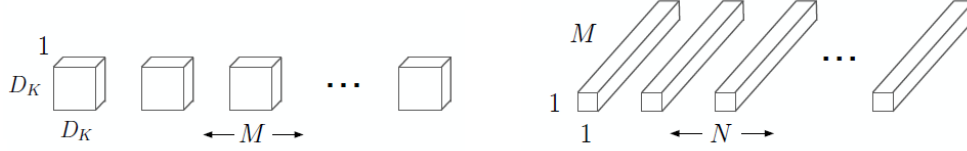
Derinlemesine ayrılabilen konvolüsyon faktöriyel konvolüsyonun özel bir formudur. MobileNet mimarisi içinde derinlemesine konvolüsyonu her bir giriş kanalına basit bir filtre uygulamak için kullanır. Derinlemesine konvolüsyon standart konvolüsyona göre çok daha verimlidir ancak giriş kanallarını sadece filtreler, yeni öznitelikler çıkarmak için onların doğrusal kombinasyonlarını çıkarmaz. Bu sebeple derinlemesine konvolüsyonun çıkışlarına noktasal konvolüsyon uygulanarak lineer kombinasyonlar yani yeni öznitelikler elde edilir.

Şekil 2.1'de standart konvolüsyon filtreleri konvolüsyon gösterilmiştir. Buna göre standart konvolüsyonda bütün giriş kanalları tek aşamada işlenir.



Şekil 2.1 : SSD mimarisinin MobileNet mimarisi ile birleştirilmesi [13].

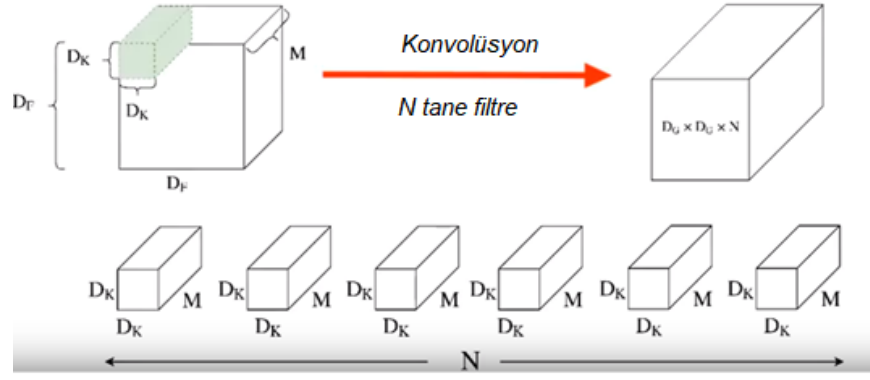
Şekil 2.2’de ise derinlemesine ayrılabilen konvolüsyon filtreleri gösterilmiştir. Derinlemesine ayrılabilen konvolüsyonda ise önce tüm giriş kanalları tek tek işlenir daha sonra ise noktasal konvolüsyon ile kanallar doğrusal olarak birleştirilir. Noktasal konvolüsyon filtreleri adını kullandıkları konvolüsyon çekirdeğinin çözünürlüğünün 1x1 olmasından almaktadır.



Şekil 2.2 : Sırasıyla derinlemesine ayrılabilen konvolüsyon filtreleri ve noktasal konvolüsyon filtreleri [14].

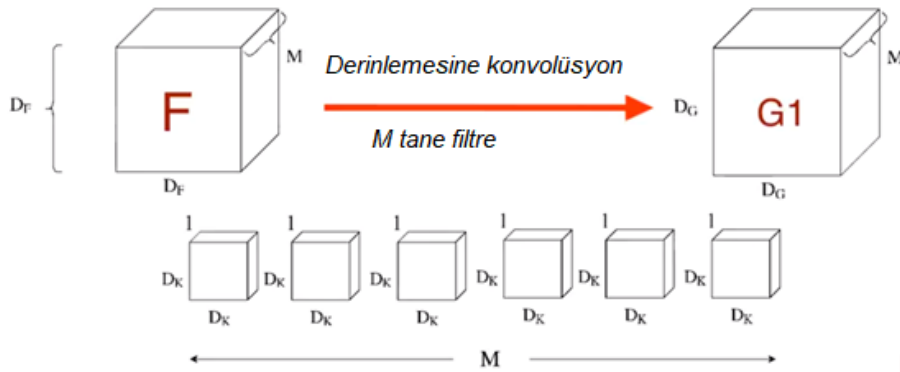
Konvolüsyon işlemi kendi içinde toplama ve çarpma işlemlerinden meydana gelmektedir. Bu sebeple konvolüsyon işleminin hesaplama maliyetini belirlemek için kullandığı çarpma ve toplama işlemi sayılarını belirlemek yeterli olacaktır. Şimdi ise her iki konvolüsyon tipini kullanabilmek için yapılması gereken çarpma ve toplama işlemi sayılarını inceleyelim. Bu sayılar aynı zamanda konvolüsyon işleminin ihtiyaç duyduğu işlem gücü ve MobileNet mimarisinin de hızı ile doğru orantılıdır.

Hesabı yapabilmek için Şekil 2.3’te görselleştirildiği üzere $D_F \times D_F$ ebatlarında M kanallı bir girişe standart konvolüsyon uygulandığı ve çıkışın da $D_G \times D_G$ ebatlarında N kanallı olduğu varsayılmıştır. Ayrıca hesabı basitleştirmek adına giriş ve çıkış ebatlarının aynı olduğu, yani $D_F = D_G$ olduğu varsayılmıştır. Buna göre girişteki tek bir noktaya standart konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı $D_K^2 \times M$ olarak ifade edilebilir. Bu durumda bütün girişe standart konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı $D_G^2 \times D_K^2 \times M$ olur. Bu işlem her bir çıkış kanalı için tekrarlanacağı için standart konvolüsyon için gerekli toplam çarpma işlemi sayısı $N \times D_G^2 \times D_K^2 \times M$ olarak hesaplanır.



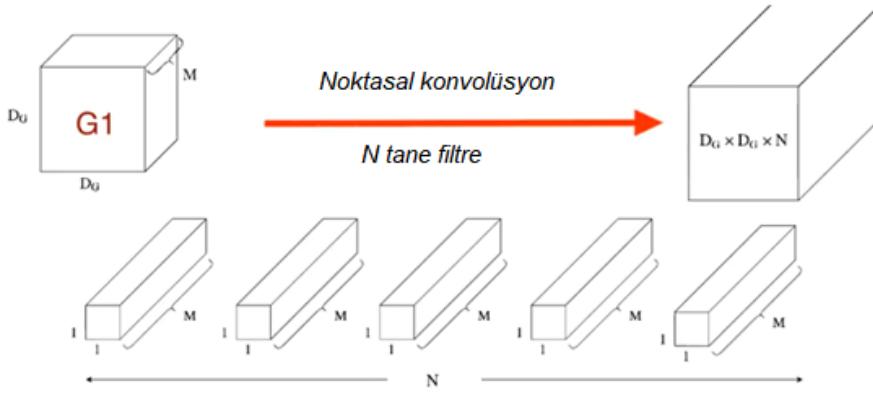
Şekil 2.3 : Standart konvolüsyonun detayları [26].

Derinlemesine ayrılabilen konvolüsyonu yapabilmek için gerekli çarpma işlemi sayısı ise derinlemesine konvolüsyon ve noktasal konvolüsyon için gerekli çarpma işlemleri sayısının ayrı ayrı hesaplanıp toplanması ile bulunabilir. Hesabı derinlemesine konvolüsyon için yapabilmek için Şekil 2.4'te görselleştirildiği üzere $D_F \times D_F$ ebatlarında M kanallı bir girişe derinlemesine konvolüsyon uygulandığı ve çıkışın da $D_G \times D_G$ ebatlarında ve yine M kanallı olduğu varsayılmıştır. Ayrıca hesabı basitleştirmek adına giriş ve çıkış ebatlarının aynı olduğu, yani $D_F = D_G$ olduğu varsayılmıştır. Buna göre girişteki tek bir noktaya derinlemesine konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı D_K^2 olarak ifade edilebilir. Bu durumda bütün girişe derinlemesine konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı $D_G^2 \times D_K^2$ olur. Bu işlem her bir çıkış kanalı için tekrarlanacağı için derinlemesine konvolüsyon için gerekli toplam çarpma işlemi sayısı $M \times D_G^2 \times D_K^2$ olarak hesaplanır.



Şekil 2.4 : Derinlemesine ayrılabilen konvolüsyonun detayları [26].

Hesabın noktasal konvolüsyon için devam ettirilebilmesi için Şekil 2.5'te görselleştirildiği üzere $D_G \times D_G$ ebatlarında M kanallı bir girişe noktasal konvolüsyon uygulandığı ve çıkışın da yine $D_G \times D_G$ ebatlarında, ancak N kanallı olduğu varsayılmıştır. Şekil 2.4 ve Şekil 2.5'ten de görülebileceği üzere hesap yapılırken noktasal konvolüsyonun, derinlemesine konvolüsyonun çıkışına uygulandığına dikkat edilmelidir. Buna göre girişteki tek bir noktaya noktasal konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı M olarak ifade edilebilir. Bu durumda bütün girişe noktasal konvolüsyon uygulanabilmesi için yapılması gereken çarpma işlemi sayısı $M \times D_G^2$ olur. Bu işlem her bir çıkış kanalı için tekrarlanacağı için noktasal konvolüsyon için gerekli toplam çarpma işlemi sayısı $N \times M \times D_G^2$ olarak hesaplanır.



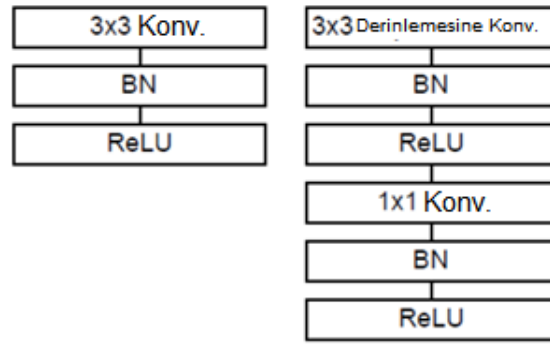
Şekil 2.5 : Noktasal konvolüsyonun detayları [26].

Yukarıda da belirtildiği gibi derinlemesine ayrılabilen konvolüsyonun ihtiyaç duyduğu toplam çarpma işlemi sayısı derinlemesine konvolüsyon ve noktasal konvolüsyonun ihtiyaç duyduğu çarpma işlemi sayılarının toplamıdır. Bu toplam yukarıda bulunan sonuçlar doğrultusunda yapılır ve ortak çarpan parantezine alınırsa, derinlemesine ayrılabilen konvolüsyon için gerekli çarpma işlemi sayısı $M \times D_G^2 \times (N + D_K^2)$ olarak bulunur. Bulunan sonuçlar kullanılarak standart konvolüsyonun ihtiyaç duyduğu çarpma işlemi sayısının derinlemesine ayrılabilen konvolüsyona oranı ise $(N \times D_K^2) / (N + D_K^2)$ olarak hesaplanır. Açıkça görüleceği üzere standart konvolüsyonda ihtiyaç duyulan çarpma işlemi sayısı çıkış kanalı sayısı ile orantılı olarak artmaktadır ve çıkış kanalı sayısı arttıkça standart konvolüsyon çok daha maliyetli hale gelmektedir.

Örneğin, 3×3 konvolüsyon çekirdeğinin kullanıldığı ve çıkışın 1024 kanallı olduğu bir durumda derinlemesine konvolüsyon yaklaşık olarak 9 kat daha az çarpma işlemine ihtiyaç duyacaktır. Bir başka deyişle konvolüsyon işlemini gerçekleştiren işlemci 9 kat daha az çarpma yapacak, yani 9 kat daha hızlı çalışacaktır.

2.1.2 MobileNet'in ağ mimarisi

MobileNetv1 mimarisi yukarıda da belirtildiği üzere derinlemesine ayrılabilen konvolüsyon işlemi üzerine kurulmuştur. Derinlemesine ayrılabilen konvolüsyon işlemi MobileNetv1'in ağ mimarisi içinde standart konvolüsyona benzer şekilde kullanılır. Farklı olan nokta ise derinlemesine ayrılabilen konvolüsyonun kendi içinde iki bölümden oluşmasıdır. Şekil 2.6'da da görülebileceği gibi standart konvolüsyona benzer şekilde hem derinlemesine konvolüsyondan sonra hem de onu izleyen noktasal konvolüsyondan sonra ReLU ve BN katmanları kullanılmıştır.



Şekil 2.6 : Solda standart konvolüsyonun, sağda ise derinlemesine ayrılabilen konvolüsyonun ReLU ve BN katmanları ile birlikte kullanımı [14].

Çizelge 2.1'de ise mimarinin detayları görülmektedir. Bu çizelgede çizelgeyi basitleştirmek adına Relu ve BN katmanlarına yer verilmemiştir, ancak her bir konvolüsyon katmanından sonra bu katmanlar da ağ mimarisi içinde kullanılmıştır. Mimari toplamda 28 katmandan oluşmaktadır ve sadece ilk katmanda standart konvolüsyon kullanılmıştır. Kalan katmanlar birbirini izleyen derinlemesine konvolüsyon ve noktasal konvolüsyondan oluşmaktadır. Son üç katmanda sınıflandırma yapabilmek adına havuzlama ve “softmax” gibi katmanlara yer verilmiştir.

Çizelge 2.1 : MobilNetv1 ağ mimarisi [14]. SK standart konvolüsyonu, DK derinlemesine konvolüsyonu ve NK da noktasal konvolüsyonu belirtmektedir.

Tip / Atlama	Filtre Ebatları	Giriş Ebatları
SK / 2	3 x 3 x 3 x 32	224 x 224 x 3
DK / 1	3 x 3 x 32	112 x 112 x 32
NK / 1	1 x 1 x 32 x 64	112 x 112 x 32
DK / 2	3 x 3 x 64	112 x 112 x 64
NK / 1	1 x 1 x 64 x 128	56 x 56 x 64
DK / 1	3 x 3 x 128	56 x 56 x 128
NK / 1	1 x 1 x 128 x 128	56 x 56 x 128
DK / 2	3 x 3 x 128	56 x 56 x 128
NK / 1	1 x 1 x 128 x 256	28 x 28 x 128
DK / 1	3 x 3 x 256	28 x 28 x 256
NK / 1	1 x 1 x 256 x 256	28 x 28 x 256
DK / 2	3 x 3 x 256	28 x 28 x 256
NK / 1	1 x 1 x 256 x 512	14 x 14 x 256
¹ DK / 1	3 x 3 x 512	14 x 14 x 512
¹ NK / 1	1 x 1 x 512 x 512	14 x 14 x 512
DK / 2	3 x 3 x 512	14 x 14 x 512
NK / 1	1 x 1 x 512 x 1024	7 x 7 x 512
DK / 2	3 x 3 x 1024	7 x 7 x 1024
NK / 1	1 x 1 x 1024 x 1024	7 x 7 x 1024
Ortalama Havuzlama / 1	Havuz 7 x 7	7 x 7 x 1024
Tam Bağlı / 1	1024 x 1000	1 x 1 x 1024
Softmax / 1	Sınıflandırıcı	1 x 1 x 1000

¹ Bu iki katmandan birisinin peşi sıra 5 adet kullanılmıştır.

Mimarinin kalan tüm katmanları ise derinlemesine konvolüsyon ve onu izleyen noktasal konvolüsyonlardan oluşmaktadır. Çizelge 2.1’de de görüldüğü gibi “down sampling” işlemi, derinlemesine konvolüsyon işleminin atlamalı olarak uygulanması ile yapılmıştır. Sol sütunda her bir katmanda uygulanan atlama miktarı görülmektedir. Örneğin, dördüncü katmanda iki adım atlama ile derinlemesine konvolüsyon işlemi yapılmıştır. Yani bir öznitelik hücresine derinlemesine konvolüsyon filtresi uygulandıktan sonra onun yan komşusuna değil de iki yan komşusuna filtre uygulanmıştır. Bunun sonucu olarak dördüncü katmanda 112 x 112 x 64 olan giriş beşinci katmanda 56 x 56 x 64 olmuştur, yani “down sampling” işlemi de derinlemesine konvolüsyon ile birlikte yapılmıştır.

Son olarak ise MobileNetv1 mimarisinin büyüklüğünü ve doğal olarak da hızını kontrol edebilmek ve hız / başarımlar oranını ayarlamak için neler yapıldığından bahsetmek gerekmektedir. Bu amaçlarla genişlik çarpanı ve çözünürlük çarpanı adında iki parametre belirlenmiştir. Genişlik çarpanı her bir katmanın giriş ve çıkış kanal

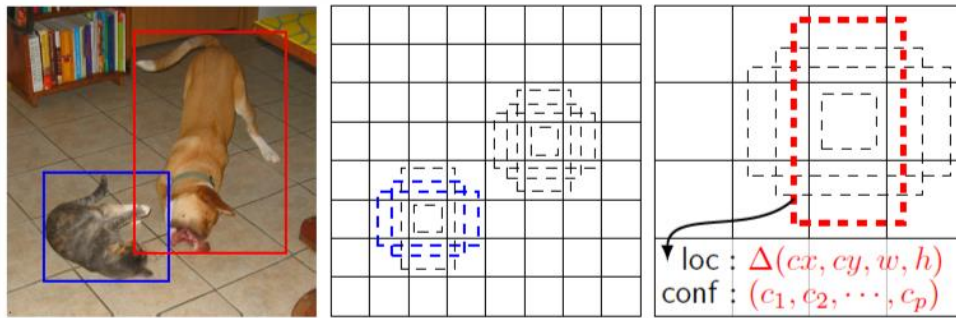
sayılarını aynı oranda büyütür veya küçültür. Çözünürlük çarpanı ise girişin ve doğal olarak da bütün katmanların çözünürlüğünü aynı oranda büyütür veya küçültür. Her iki çarpanın da hıza etkisi kareseldir ancak bu çarpanları çok küçük değerlerde kullanmak başarımı ciddi anlamda olumsuz etkilemektedir.

2.2 SSD

SSD tek aşamalı bir nesne tespit yöntemidir ve temelleri Multibox'a [12] dayanmaktadır. Kısaca temel bir CNN mimarisinin ürettiği öznetelik haritalarını farklı çözünürlükler için değerlendirerek nesne tespiti yaptığı söylenebilir. Temel CNN mimarisi olarak neler kullanıldığı ve farklı çözünürlükteki öznetelik haritalarının nasıl değerlendirildiği bu başlığın ilk alt başlığında açıklanmıştır. İkinci alt başlıkta ise SSD'nin nasıl eğitildiği ve detayları verilmiştir.

2.2.1 SSD'in ağ mimarisi

SSD tek aşamalı bir yöntem olarak seçici arama veya RPN gibi yöntemler kullanarak olası nesne pozisyonlarını çıkarmaya çalışmaz. Bunun yerine öncelikle Şekil 2.7'deki gibi çeşitli boyutlarda öznetelik haritaları çıkartır. Sonra bu öznetelik haritalarındaki her bir noktaya belirli büyüklük ve en-boy oranlarında varsayılan kutular atar. Daha sonra atanmış olan her bir varsayılan kutu için ayrı ayrı işlem yapar.

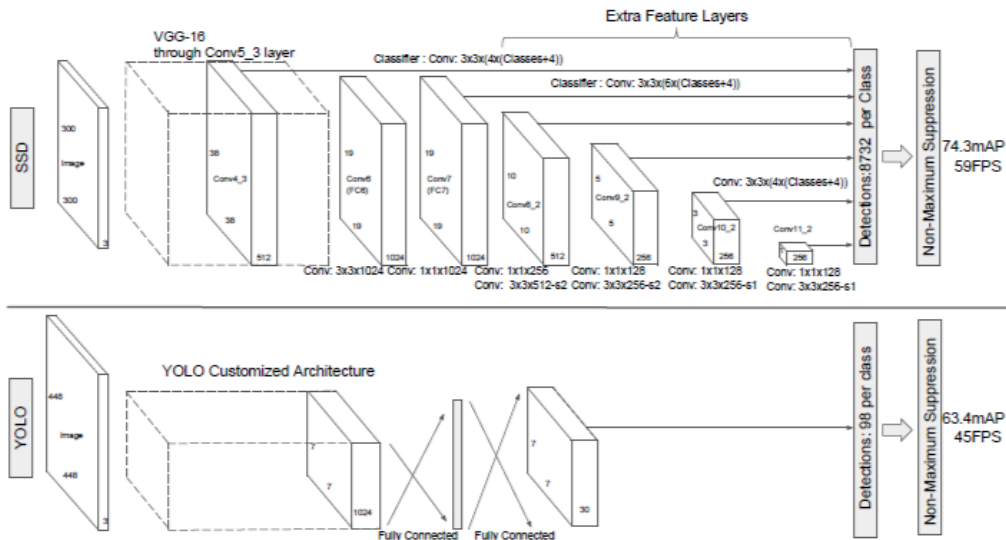


Şekil 2.7 : SSD'nin nesne tanıma yapısı [13]. Soldan sağa sırasıyla ground truth kutuları, 8x8 öznetelik haritası, 4x4 öznetelik haritası ve bu haritalar üzerindeki varsayılan kutular görülmektedir.

Şekil 2.7'i detaylıca incelemek gerekirse giriş resminde büyük bir köpek ve ebat olarak daha küçük bir kedi bulunmaktadır. Ayrıca giriş resmine ait 8x8 ve 4x4 boyutlarında iki farklı öznetelik haritası çıkartılmış ve bu haritalardaki noktalara farklı boyut ve en-boy oranlarında varsayılan kutular atanmıştır. Daha sonra ise her bir kutu için işlem yapılmıştır. Yapılan bu işlemin detaylarına ileride değinilecektir. Kedi köpeğe kıyasla

daha küçük ebatlı olduğu için çözünürlüğü daha büyük olan 8x8 harita sayesinde tespit edilmiştir. Öte yandan, daha büyük ebatlı olan köpek ise çözünürlüğü daha küçük olan 4x4 harita sayesinde tespit edilmiştir. Bunun sebebi ise giriş resminin 8x8 öznitelik haritasında 64 nokta tarafından, 4x4 öznitelik haritasında ise 16 nokta tarafından temsil edilmesidir. Yani kullanılan öznitelik haritasının çözünürlüğü azaldıkça giriş resmi daha az nokta tarafından temsil edilmektedir, bir başka deyişle öznitelik haritasındaki her bir noktanın giriş resminde temsil ettiği alan büyümektedir. Bunun bir sonucu olarak Şekil 2.7’de olduğu gibi genel olarak büyük nesneler küçük çözünürlüklü haritalar tarafından küçük nesneler ise büyük çözünürlüklü haritalar tarafından daha rahat tespit edilir.

Piramit benzeri yapay sinir ağı mimarilerini kullanan CNN’lerde girişten çıkışa doğru gidildikçe havuzlama ve “down sampling” gibi yöntemler kullanılarak öznitelik haritalarının çözünürlüğü düşürülür ve genelde sadece çıkıştaki en küçük çözünürlüklü harita kullanılarak işlem yapılır. Ancak bu durumda yukarıda anlatılan sebeplerden ötürü küçük ebatlı nesnelerin tespiti zor olabilir. Bu durumun en tipik örneklerinden birisi YOLO v1’dir [9]. Her ne kadar ilerleyen sürümlerde bu sorun çözülmüş olsa da YOLO v1 küçük ebatlı nesnelerde iyi sonuçlar vermemektedir. Öte yandan SSD Şekil 2.8’de de görülebileceği gibi farklı çözünürlüklü öznitelik haritalarını da kullanarak çıkardığı ilave öznitelikler sayesinde küçük ebatlı objelerde YOLO v1’ye göre nispeten daha iyi sonuç vermektedir.



Şekil 2.8 : SSD ve YOLO v1’in ağ mimarilerini karşılaştırılması [13]. SSD’nin kullandığı ilave öznitelik haritaları kolaylıkla görülebilir.

Yukarıda neden farklı çözünürlüklerde öznitelik haritalarının çıkarıldığı ve bunların nasıl kullanıldığından bahsedilmiştir. Şimdi ise öznitelik haritalarının nasıl çıkarıldığından ve mimarinin genelinden bahsedilecektir. Özniteliklerin çıkarılması için temel bir CNN mimarisi kullanılır. Burada temel CNN mimarisi olarak herhangi yüksek kaliteli CNN mimarisi seçilebilir. Yapılması gereken ise seçilen temel mimarinin çözünürlük değerlerine de dikkat edilerek sınıflandırma aşaması için barındırdığı katmanlardan itibaren kesilmesidir. Çünkü seçilen temel mimari sınıflandırma amacıyla değil, sadece öznitelik çıkarımı amacıyla kullanılacaktır. Sınıflandırma ise SSD'nin kendisi tarafından yapılacaktır. Bu bakımdan ele alındığından SSD bahsi geçen temel mimariden bağımsız tasarlanmıştır ve herhangi bir temel CNN mimarisi ile birlikte kullanılabilir. Şekil 2.8'de de görüldüğü gibi SSD'nin orijinal sürümünde temel CNN mimarisi olarak VGG-16 kullanılmıştır. Ancak, VGG-16 yerine ResNet ve ResNet + FPN kullanan sürümleri de mevcuttur. Görüldüğü gibi öznitelik çıkarımı yapan temel CNN mimarisinden bağımsız tasarlanmış olması SSD'ye büyük bir esneklik katmıştır.

2.2.2 SSD'nin eğitimi ve detayları

Yukarıda SSD'nin genel yapısı, kullanılan temel CNN mimarisi ve nesne tespiti için oluşturulan varsayılan kutuların detaylarına değinildi. Şimdi ise her bir varsayılan kutu için ne gibi işlemler yapıldığına değinilecektir. Kısaca özetlemek gerekirse SSD her bir varsayılan kutuda tespit edilmesi hedeflenen her bir nesne sınıfı için ayrı ayrı sınıf skoru ve kutu kayması hesaplar. Daha sonra NMS kullanarak nihai sonuca erişir.

Detaylarına inmek gerekirse SSD eğitim aşamasında öncelikle olası varsayılan kutular ile ground truth kutularını birbirine eşler. Benzerlerinden farklı olarak ise bu eşlemeyi karşılıklı olarak yapar. Önce her bir ground truth kutusunu kendisi ile en iyi IoU veren varsayılan kutuya eşler. Sonra ise her bir varsayılan kutuyu ise kendisi ile 0.5'ten daha büyük IoU veren tüm ground truth kutularına eşler. Daha sonra eğitim aşamasını sadece bu eşlemeleri kullanarak devam ettirir.

Eğitim aşamasında ise pozisyon kaybını ve nesne skoru kaybını birlikte kullanan bir kayıp fonksiyonu kullanılır.

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \quad (2.1)$$

Denklem 2.1’de N eşlenmiş varsayılan kutu sayısını, α ise pozisyon kaybı ile nesne skoru kaybı arasındaki ağırlıklandırma katsayısını ifade eder. Pozisyon kaybı L_{loc} ise tahmin edilmeye çalışılan kutu l ile ground truth kutusu g arasındaki “smooth $L1$ ” kaybıdır. Nesne skoru kaybı olan L_{conf} ise tahmin edilmeye çalışılan sınıfın nesne skoru c ile tüm sınıfların nesne skorunun “softmax” kullanılarak işlenmesiyle elde edilir.

Denklem 2.2’de pozisyon kaybının detayları görülebilir. Buradaki parametreler varsayılan kutu d ’nin merkez koordinatlarının kayması olan cx ve cy ile genişliği ve yüksekliği olan w ve h ’dir.

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k smoothL1(l_i^m - \hat{g}_j^m) \quad (2.2)$$

Denklem 2.3’te $\hat{g}_j^{cx}$ ifadesinin açılımı görülebilir.

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \quad (2.3)$$

Denklem 2.4’te $\hat{g}_j^{cy}$ ifadesinin açılımı görülebilir.

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \quad (2.4)$$

Denklem 2.5’te $\hat{g}_j^w$ ifadesinin açılımı görülebilir.

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad (2.5)$$

Denklem 2.6’da $\hat{g}_j^h$ ifadesinin açılımı görülebilir.

$$\hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right) \quad (2.6)$$

Denklem 2.7’de nesne skoru kaybının detayları mevcuttur. Buradaki c parametresi ilgili sınıfın skorudur.

$$L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad (2.7)$$

Denklem 2.8’de $\hat{c}_i^p$ ifadesinin açılımı mevcuttur.

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (2.8)$$

Bütün öğrenme ve katsayı güncelleme işlemlerinde 0.9 momentum ve 0.0005 “weight decay” ile SGD kullanılır. SGD bir fonksiyonun minimumunu bulmak için kullanılan, birinci dereceden, yinelemeli bir optimizasyon algoritmasıdır. Denklem 2.9’da eski katsayı θ_j^{eski} ve toplam kayıp fonksiyonu $J(\theta)$ kullanılarak yeni katsayı θ_j^{yeni} ’nin nasıl hesaplandığı görülmektedir. Buradaki α öğrenme oranıdır ve toplam kaybın yeni katsayıyı hangi oranda etkileyeceğini belirler. γ ise momentumdur ve eski katsayının yeni katsayıyı hangi oranda etkileyeceğini belirler. “Weight decay” ise λ parametresi tarafından temsil edilmektedir.

$$\theta_j^{yeni} = \gamma \theta_j^{eski} - \alpha \left(\frac{\partial}{\partial \theta_j^{eski}} J(\theta) - \lambda \theta_j^{eski} \right) \quad (2.9)$$

Tüm bu işlemlerin sonucunda SSD’nin farklı boyutlardaki öznetelik haritalarını da kullandığını göz önünde bulundurulursa oluşan pek çok varsayılan kutu negatif örnek olacaktır. Bu durumda ciddi bir sınıf dengesizliğine yol açacaktır. Bunu engellemek için “hard negative mining” yöntemi kullanılarak negatif örneklerin pozitif örneklere oranının en fazla 3 olması sağlanır.

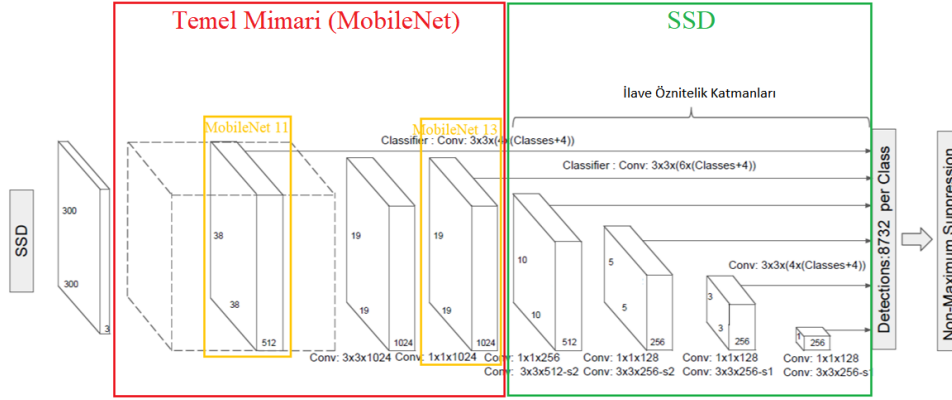
Ayrıca eğitim aşamasında her bir hücre için üretilen varsayılan kutu sayısının arttırılması daha iyi sonuçlar vermektedir. Ancak bu durum nesne tespit sürecini yavaşlatmaktadır. Bu sebeple üretilen varsayılan kutu sayısını sınırlandırmakta fayda vardır.

2.3 SSD’nin MobileNet mimarisi ile birleştirilmesi

Önceki alt başlıklarda MobileNetv1’in ve SSD’nin detaylarından ve ağ yapılarından bahsedilmiştir. Bu alt başlıkta ise SSD ve MobileNetv1’in nasıl birleştirildiğinden bahsedilecektir. Burada SSD bir nesne tespit yöntemi olduğunu, MobileNetv1’in ise bir CNN mimarisi olduğunu hatırlatmak gerekmektedir.

Önceki alt başlıklarda SSD’nin öznetelik çıkarımı için kullanılan temel CNN mimarisinden bağımsız olarak tasarlandığından bahsedilmiştir. SSD ve MobileNetv1’in birleştirilmesi de aslında bu durumdan faydalanılarak yapılmıştır.

SSD'nin öznitelik çıkarımı için kullandığı VGG-16 mimarisi yerine MobileNetv1 mimarisi kullanılmıştır. Sınıflandırma işlemi ise yine farklı çözünürlüklerdeki öznitelik haritalarını kullanarak SSD tarafından yapılmaktadır. Şekil 2.9'da da görülebileceği gibi kullanılan farklı çözünürlükteki öznitelik haritalarından dördü SSD tarafından temel CNN mimarisinin sonuna eklenen katmanlar iken, kalan ikisi temel CNN mimarisinin sonuncu ve sondan üçüncü katmanlarıdır.



Şekil 2.9 : SSD mimarisinin MobileNet mimarisi ile birleştirilmesi [13].

Şekil 2.9'da da görüldüğü üzere toplam 28 katmana sahip olan MobileNetv1 mimarisi 13. katmandan kesilmiş yani sadece ilk 13 katmanı kullanılmıştır [25]. 11. ve 13. katmanlar ise temel CNN mimarisine ait olmasına rağmen karar vermede kullanılan öznitelik haritalarıdır. Çizelge 2.2'de ise bu kesilmiş MobileNetv1 mimarisinin katmanları görülmektedir. Çizelgenin en altında 14. katmanın giriş boyutlarına yer verilmiştir çünkü bu boyutlar aynı zamanda 13. yani sonuncu katmanın çıkış boyutlarıdır.

Çizelge 2.2 ve Şekil 2.9 birlikte incelendiğinde boyutların tutarsız olduğu göze çarpmaktadır. Şekil 2.9'a göre, birleştirme işleminin yapılabilmesi için MobileNetv1'in 11. katmanının çıkış boyutları $38 \times 38 \times 512$ ve 13. katmanının çıkış boyutları da $19 \times 19 \times 1024$ olmalıdır. Hâlbuki Çizelge 2.2'ye göre MobileNetv1'in 11. katmanının çıkış boyutları $28 \times 28 \times 256$ ve 13. katmanının çıkış boyutları da $14 \times 14 \times 512$ 'dir. MobileNetv1 alt başlığı altında MobileNetv1 mimarisinin genişlik çarpanı ve çözünürlük çarpanı isimli iki adet serbest parametreye sahip olduğundan bahsedilmiştir. İşte bu çarpanlar kullanılarak yukarıda belirtilen boyut uyumsuzluğu giderilebilir.

Çizelge 2.2 : Kesilmiş MobilNet ağ mimarisi. SK standart konvolüsyonu, DK derinlemesine konvolüsyonu ve NK da noktasal konvolüsyonu belirtmektedir.

Tip / Atlama	Filtre Ebatları	Giriş Ebatları
SK / 2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
DK / 1	$3 \times 3 \times 32$	$112 \times 112 \times 32$
NK / 1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
DK / 2	$3 \times 3 \times 64$	$112 \times 112 \times 64$
NK / 1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
DK / 1	$3 \times 3 \times 128$	$56 \times 56 \times 128$
NK / 1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
DK / 2	$3 \times 3 \times 128$	$56 \times 56 \times 128$
NK / 1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
DK / 1	$3 \times 3 \times 256$	$28 \times 28 \times 256$
NK / 1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
DK / 2	$3 \times 3 \times 256$	$28 \times 28 \times 256$
NK / 1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
		$14 \times 14 \times 512$

Çözünürlük çarpanı her bir katmanın en ve boyunu aynı oranda kontrol eder. SSD'nin giriş katmanı boyutları 300×300 iken MobileNetv1 için bu değer 224×224 'tür. Bu durumda çözünürlük çarpanı $300 / 224$ yani 1.34 seçilirse katmanların en ve boy değerlerinin SSD ile uyumlu hale geldiği Çizelge 2.3'te görülebilir.

Çizelge 2.3 : Kesilmiş MobilNetv1 ağ mimarisi (Çözünürlük çarpanı = 1.34). SK standart konvolüsyonu, DK derinlemesine konvolüsyonu ve NK da noktasal konvolüsyonu belirtmektedir.

Tip / Atlama	Filtre Ebatları	Giriş Ebatları
SK / 2	$3 \times 3 \times 3 \times 32$	$300 \times 300 \times 3$
DK / 1	$3 \times 3 \times 32$	$150 \times 150 \times 32$
NK / 1	$1 \times 1 \times 32 \times 64$	$150 \times 150 \times 32$
DK / 2	$3 \times 3 \times 64$	$150 \times 150 \times 64$
NK / 1	$1 \times 1 \times 64 \times 128$	$75 \times 75 \times 64$
DK / 1	$3 \times 3 \times 128$	$75 \times 75 \times 128$
NK / 1	$1 \times 1 \times 128 \times 128$	$75 \times 75 \times 128$
DK / 2	$3 \times 3 \times 128$	$75 \times 75 \times 128$
NK / 1	$1 \times 1 \times 128 \times 256$	$38 \times 38 \times 128$
DK / 1	$3 \times 3 \times 256$	$38 \times 38 \times 256$
NK / 1	$1 \times 1 \times 256 \times 256$	$38 \times 38 \times 256$
DK / 2	$3 \times 3 \times 256$	$38 \times 38 \times 256$
NK / 1	$1 \times 1 \times 256 \times 512$	$19 \times 19 \times 256$
		$19 \times 19 \times 512$

Katmanların en ve boyları uyumlu hale getirildikten sonra ise kanal sayıları da uyumlu hale getirilmelidir. Genişlik çarpanı ise bu amaçla kullanılabilir. Genişlik her bir katmanın kanal sayısını aynı oranda kontrol eder. SSD'nin ilk ilave öznetelik katmanın giriş kanal sayısı 1024 iken MobileNetv1'n 13. katmanının çıkış kanal sayısı 512'dir. Bu durumda genişlik çarpanı $1024 / 512$ yani 2 seçilirse kanal sayılarının da uyumlu hale geldiği Çizelge 2.4'te görülebilir.

Çizelge 2.4 : Kesilmiş MobilNetv1 ağ mimarisi (Çözünürlük çarpanı = 1.34, genişlik çarpanı = 2). SK standart konvolüsyonu, DK derinlemesine konvolüsyonu ve NK da noktasal konvolüsyonu belirtmektedir.

Tip / Atlama	Filtre Ebatları	Giriş Ebatları
SK / 2	$3 \times 3 \times 3 \times 64$	$300 \times 300 \times 3$
DK / 1	$3 \times 3 \times 64$	$150 \times 150 \times 64$
NK / 1	$1 \times 1 \times 64 \times 128$	$150 \times 150 \times 64$
DK / 2	$3 \times 3 \times 128$	$150 \times 150 \times 128$
NK / 1	$1 \times 1 \times 128 \times 256$	$75 \times 75 \times 128$
DK / 1	$3 \times 3 \times 256$	$75 \times 75 \times 256$
NK / 1	$1 \times 1 \times 256 \times 256$	$75 \times 75 \times 256$
DK / 2	$3 \times 3 \times 256$	$75 \times 75 \times 256$
NK / 1	$1 \times 1 \times 256 \times 512$	$38 \times 38 \times 256$
DK / 1	$3 \times 3 \times 512$	$38 \times 38 \times 512$
NK / 1	$1 \times 1 \times 512 \times 512$	$38 \times 38 \times 512$
DK / 2	$3 \times 3 \times 512$	$38 \times 38 \times 512$
NK / 1	$1 \times 1 \times 512 \times 1024$	$19 \times 19 \times 512$
		$19 \times 19 \times 1024$

Çizelge 2.5'te SSD ile birleştirilmiş MobileNetv1 mimarisinin katmanları ve detayları görülebilir. Özetlemek gerekirse, MobileNetv1 13. katmanından itibaren kesilmiş ve ilk 13 katmanın ebatları çözünürlük çarpanı ve genişlik çarpanına uygun değerler verilerek SSD'ye uyumlu hale getirilmiştir. Ardından yeni oluşan bu yapıya SSD'nin ilave 4 katmanı da eklenerek işlem tamamlanmış ve 17 katmanlı yeni birleşik yapı elde edilmiştir. Bu yapının birinci katmanı ve son üç katmanı standart konvolüsyon işlemini uygulamaktadır. Kalan katmanlar birbirini izleyen derinlemesine konvolüsyon ve noktasal konvolüsyonlardan oluşmaktadır. Katmanlar arasında yapılan atlama işlemi sayesinde derin sinir ağı boyunca ilerledikçe çözünürlük düşürülürken kanal sayısı arttırılmıştır.

Çizelge 2.5 : SSD ile birleştirilmiş MobileNetv1 mimarisi. SK standart konvolüsyonu, DK derinlemesine konvolüsyonu ve NK da noktasal konvolüsyonu belirtmektedir.

Tip / Atlama	Filtre Ebatları	Giriş Ebatları
SK / 2	3 x 3 x 3 x 64	300 x 300 x 3
DK / 1	3 x 3 x 64	150 x 150 x 64
NK / 1	1 x 1 x 64 x 128	150 x 150 x 64
DK / 2	3 x 3 x 128	150 x 150 x 128
NK / 1	1 x 1 x 128 x 256	75 x 75 x 128
DK / 1	3 x 3 x 256	75 x 75 x 256
NK / 1	1 x 1 x 256 x 256	75 x 75 x 256
DK / 2	3 x 3 x 256	75 x 75 x 256
NK / 1	1 x 1 x 256 x 512	38 x 38 x 256
DK / 1	3 x 3 x 512	38 x 38 x 512
NK / 1	1 x 1 x 512 x 512	38 x 38 x 512
DK / 2	3 x 3 x 512	38 x 38 x 512
NK / 1	1 x 1 x 512 x 1024	19 x 19 x 512
SK / 2	3 x 3 x 1024	10 x 10 x 1024
SK / 2	3 x 3 x 512	5 x 5 x 512
SK / 2	3 x 3 x 256	3 x 3 x 256
		1 x 1 x 256

Çizelge 2.6’da SSD’nin öznetelik çıkarımı için farklı temel CNN mimarilerini kullanan sürümlerinin ihtiyaç duyduğu toplam toplama ve çarpma işlemi sayılarını ve parametre sayılarını görmek mümkündür. Burada nesne tespiti yapmak için ihtiyaç duyulan toplam toplama ve çarpma işlemi sayısı ihtiyaç duyulan işlem gücüyle, ihtiyaç duyulan parametre sayısı ise ihtiyaç duyulan bellek miktarı ile doğru orantılıdır.

Çizelge 2.6 : SSD’nin farklı temel CNN mimarileri kullanılarak hazırlanmış sürümlerinin karşılaştırılması [14].

Model	mAP	Toplama ve Çarpma Sayısı (milyar)	Parametre Sayısı (milyon)
VGG-16	% 21.1	34.9	33.1
Inception	% 22.0	3.8	13.7
MobileNet	% 19.3	1.2	6.8

Görüldüğü üzere SSD’nin MobileNetv1 ile birleşimi ihtiyaç duyulan işlem gücünü ve bellek miktarını ciddi miktarda azalmıştır. Tüm bunların bir sonucu olarak da SSD’yi artık mobil cihazlarda da çalışabilecek kadar hızlı hale getirmiştir.



3. MOBİL KAMERA HAREKETİNİN MODELLENMESİ

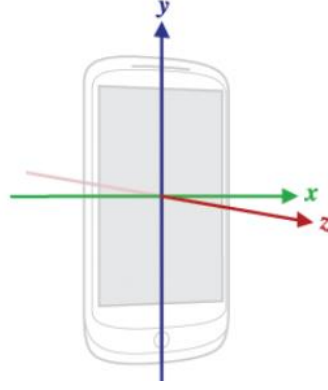
Tez çalışması kapsamında amaçlanan, mobil cihaz sensörlerinden alınan verilerden çıkarılan hareket bilgisi kullanılarak, nesne izleme performansının artırılmasıdır. Bu amaçla öncelikle ivmeölçerden alınan veri üzerinde gürültü giderme gerçekleştirilmiş, ardından ivme verisi kullanılarak mobil cihazın hareketinden kaynaklanan hız bilgisi elde edilmiştir. Cihaz üzerine monte edilmiş mobil kameranın cihaz ile aynı hızda hareket ettiği kabulü ile çıkarılan hız verisi kullanılarak kamera hareketi modellenmiş ve nesne seziciye geri besleme olarak gönderilen hız bilgisi sayesinde nesne izleme performansı artırılmıştır.

Bu bölümde, ivmeölçerden alınan veri kullanılarak afin kamera hareketi parametrelerini nasıl hesaplandığı anlatılacaktır. Altbölüm 3.1’de ivmeölçerden alınan veriden kamera hızının hesaplanması formüle edilmektedir. Altbölüm 3.2’de afin hareket varsayımı altında kamera hareketi parametrelerinin hesaplanması anlatılmıştır. Altbölüm 3.3’de sensörden alınan bilgi ile hesaplanan hareketin doğrulanması yapılarak kestirim sonuçları ile uyumluluğu tartışılmaktadır

3.1 İvmeölçer Verisinden Mobil Kamera Hızının Elde Edilmesi

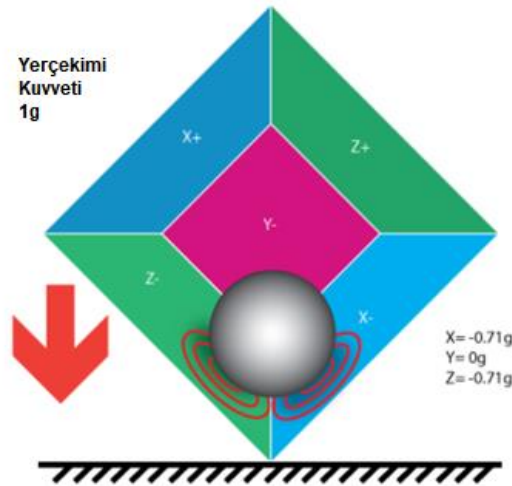
Mobil cihazlarda bulunan ivmeölçer üzerinde bulunduğu cihazın sahip olduğu ivmeyi ölçen bir sensördür. İvmeölçerler genelde çıkış verisi olarak m/s^2 cinsinden veya G cinsinden ivmeyi verirler. Burada G yerçekimi ivmesinin bir ifadesidir ve $1 G$ büyüklüğündeki ivme $9.81 m/s^2$ olarak kabul edilir.

Piyasada çok çeşitli tipte ivmeölçerler olmakla birlikte bu tez çalışmasında kullanılan akıllı telefon da dâhil olmak üzere mobil telefonlar çoğunlukla 3D yani 3 eksenli ivmeölçer kullanırlar. İvmeölçerlerin eksenlerinin doğrultuları karışıklığa yer vermemek adına tüm Android telefonlarda Şekil 3.1’de gösterildiği gibidir.



Şekil 3.1 : Android telefonlardaki ivmeölçerin eksen doğrultuları [27].

Şekil 3.1’de de görüldüğü üzere ivmeölçer her bir eksenindeki ivme bilgisini çıkışta ayrı ayrı verir. Cihaza etki eden gerçek ivme bilgisi ise bu eksenlerdeki ivme vektörlerinin bileşkesi alınarak bulunabilir. Öte yandan ivmeölçerler çalışma prensipleri gereği üzerinde bulundukları cihaz hareket etmiyorken dahi çıkıştaki ivme verisini 0 m/s^2 olarak vermezler. Bunun sebebi cihaza dünyanın yerçekimi ivmesinin (9.81 m/s^2) sürekli olarak etki etmesidir. Diğer taraftan bir ivmeölçer ancak serbest uzayda duruyorken veya dünyamızda cihaz serbest düşüş yapıyorken çıkıştaki ivme verisini 0 m/s^2 olarak verir. Ayrıca yerçekimi ivmesinin ivmeölçerin hangi eksenlerine ne ölçüde etki edeceği de cihazın o sırada nasıl bir pozisyonda durduğu ile alakalıdır. Bunun bir örneği Şekil 3.2’de görülebilir.



Şekil 3.2 : Yerçekimi ivmesinin farklı eksenlere etkisi [28].

Buraya kadar irdelenenlerden de anlaşılacağı gibi bir ivmeölçer üzerinde bulunduğu cihaz duruyorken dahi yerçekimi ivmesi sebebiyle en az bir eksende sıfırdan farklı bir ivme değeri ölçer. Ölçülen bu yerçekimi ivmesi bileşenleri nispeten düşük frekanslı

bir gürültü gibi modellenebilir ve yüksek geçiren karakteristiğine sahip bir filtre ile bastırılır. Ayrıca her algılayıcıda olduğu gibi ivmeölçer de nispeten yüksek frekanslı bir ölçüm gürültüsüne de sahiptir. Bu gürültü ise alçak geçiren karakteristiğine sahip bir filtre ile bastırılır. Görüleceği üzere gerçek ivme verisini elde edebilmek için alçak ve yüksek geçiren karakteristiğe sahip filtreler peşi sıra kullanılmalıdır ve bu durum bir bant geçiren filtrenin kullanılmasına karşı düşmektedir.

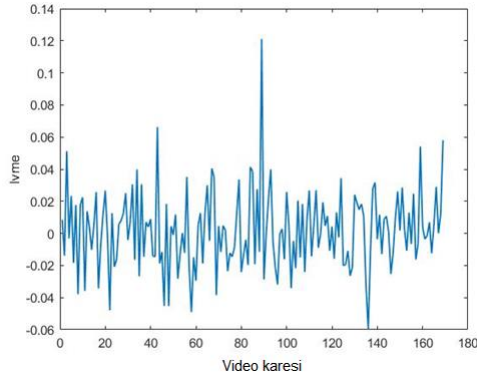
Pratikte bahsedilen bant geçiren filtre karakteristiğinin elde edilebilmesi için öncelikle yerçekimi ivmesinin her bir eksenindeki bileşeni alçak geçiren karakteristiğine sahip bir filtre kullanılarak elde edilir [29]. Denklem 3.1’de kullanılan bu alçak geçiren filtreye ait fark denklemi görülmektedir. Denklemdaki g , n anında bulunmak istenilen yerçekimi bileşenini, α_{meas} ivmeölçer tarafından ölçülen ivme değerini, β ise ağırlıklandırma katsayısını gösterir ve bu katsayı aksi belirtilmedikçe 0.8 olarak kullanılmaktadır [29].

$$g[n] = \beta * g[n - 1] + (1 - \beta) * a_{meas}[n] \quad (3.1)$$

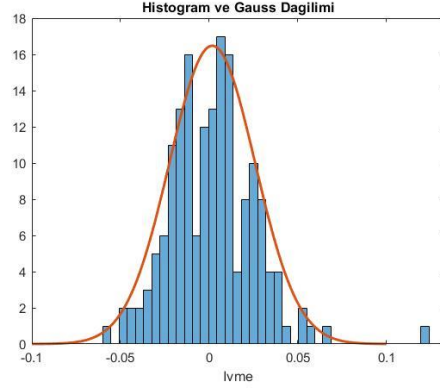
Yerçekimi ivmesinin her bir eksenindeki bileşeninin bulunmasının ardından, yerçekimi ivmesi bileşenleri ölçülen bileşenlerden çıkarılarak gerçek ivme bileşenleri her bir eksen için elde edilir [29]

Tüm bu filtreleme işlemlerinin ardından yerçekimi ivmesine ve ölçme gürültüsüne ait bileşenler büyük ölçüde sönümlenir ancak filtreleme işleminin doğası gereği tamamen yok edilemezler. Şekil 3.3(a)’da sabit duran bir cihazın üzerindeki ivmeölçerin yaptığı ölçümün z – eksenindeki bileşenlerinin filtrelenmesi ile elde edilen gerçek ivme verisi m/s^2 biriminde görülmektedir. Cihaz sabit durduğu için sıfır olarak ölçülmesi gereken ivme verisi, uygulanan filtre gürültüyü ve yerçekimi ivmesi bileşenlerini tamamen sönümleyemediği için, Şekil 3.3(a)’daki gibi sıfıra yakın ancak sıfırdan farklı olarak ölçülmüştür.

Şekil 3.3 (a)’daki ivme verisinin histogramı ve bu histograma uyan Gauss dağılımı ise Şekil 3.3(b)’de verilmiştir. Normalde ivmeölçerin kendisinden kaynaklanan ve Gauss karakteristiği gösteren ölçüm gürültüsünün ortalamasının sıfır olması beklenir, ancak Şekil 3.3(b)’den hesaplanan ortalama 0.0017 ve standart sapma değeri 0.024 tür. Bunun sebebi, tıpkı ölçüm gürültüsü gibi yerçekimi ivmesinin bileşenlerinin de tamamen sönümlenememesi ve gürültü gibi davranmasıdır.



(a)



(b)

Şekil 3.3 : (a) Sabit bir cihaza ait filtrelenmiş ivme verisinin zamanla değişimi.
(b) Aynı ivme verisinin histogramı

Burada dikkat edilmesi gereken bir diğer husus, ölçüm gürültüsünün her zaman aynı karakteristiğe sahip olmasına karşın, yerçekimi ivmesinin bileşenlerinden kaynaklanan DC karakteristikteki gürültünün cihazın pozisyonuna göre sürekli değişiklik göstermesi ve asla tam olarak modellenememesidir. Bu sebeple sabit duran bir cihazın sadece pozisyonu değiştirilerek ölçümler tekrarlanırsa yaklaşık olarak aynı standart sapmaya fakat farklı ortalamaya sahip bir Gauss dağılımı elde edilir. Örneğin, bahsedildiği gibi ölçüm farklı bir pozisyon için tekrarlandığında bu sefer ortalama 0.0009 ve standart sapma 0.020 olarak hesaplanmıştır. Görüldüğü üzere standart sapma yaklaşık olarak aynı kalmışken ortalama %47 oranında azalmıştır. Ortalamadaki bu değişimin etkilerine ileride değinilecektir.

Yapılan tez çalışması kapsamında kamera hareketinin modellenebilmesi için kameranın üzerinde bulunduğu mobil cihazın hızından yararlanılmaktadır. Bu sebeple birimi m/s^2 olan ivme verisi kullanılarak birimi m/s olan hız verisinin elde edilmesi gerekmektedir. Sürekli zaman uzayında ivmenin zamana göre integrali hızı vermektedir. Tez çalışmasında ayrık zaman uzayında çalışılması nedeniyle integral işlemi Denklem 3.2’de görülen fark denkleminde karşı düşer. Burada $v[n]$ ayrık zaman uzayında n anındaki hızı, $a[n]$ ivmeyi ve Δt ise ardışık iki ivme ölçümü arasında geçen zamanı belirtir. Bu işlem aynı zamanda Δt ile ağırlıklandırılmış birikimli toplama işlemine de karşı düşmektedir.

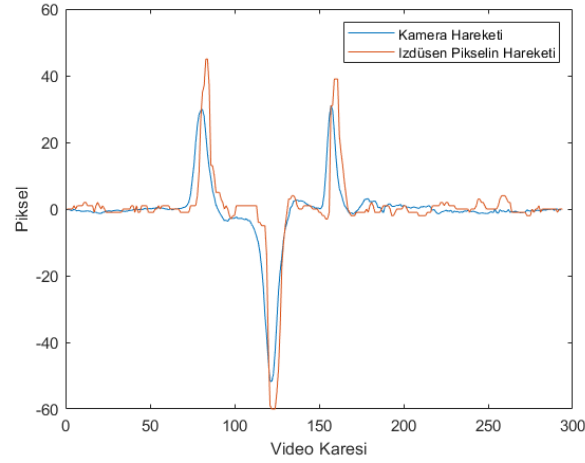
$$v[n] = v[n - 1] + \Delta t * a[n] \quad (3.2)$$

Denklem(3.2) ile hesaplanan m/s birimli hız verisinin tez kapsamında nesne izleme performansını arttırmak amacıyla kullanılması önerildiğinden $piksel/s$ cinsine çevrilebilmesi gerekmektedir. Hız verisine ait bu iki farklı birim arasında kamera parametreleri ve nesnenin kameraya olan uzaklığına göre değişen ve üçgen benzerliği bağıntıları ile hesaplanabilen bir ilişki vardır. Özellikle izlenilecek nesnenin kameraya olan uzaklığının eş zamanlı olarak görüntü işleme teknikleri ile hesaplanması bu tez çalışmasının kapsamının dışındadır. Bu sebeple bu tez çalışması kapsamında üzerinde çalışılan videolarda izlenmesi gereken nesnenin kameraya olan uzaklığının yaklaşık olarak bilindiği ve video boyunca yaklaşık olarak sabit kaldığı varsayılmıştır. Bu varsayım altında, m/s cinsinden hız değeri $piksel/s$ cinsinden hız değerine Denklem 3.3 kullanılarak çevrilebilir. Burada $v_m[n]$ m/s cinsinden hızı, $v_p[n]$ $piksel/s$ cinsinden hızı ve γ kamera parametrelerine ve cismin kameraya olan uzaklığına göre değişen ve üçgen benzerliği bağıntılarıyla hesaplanan ölçekleme katsayısını göstermektedir.

$$v_p[n] = \gamma * v_m[n] \quad (3.3)$$

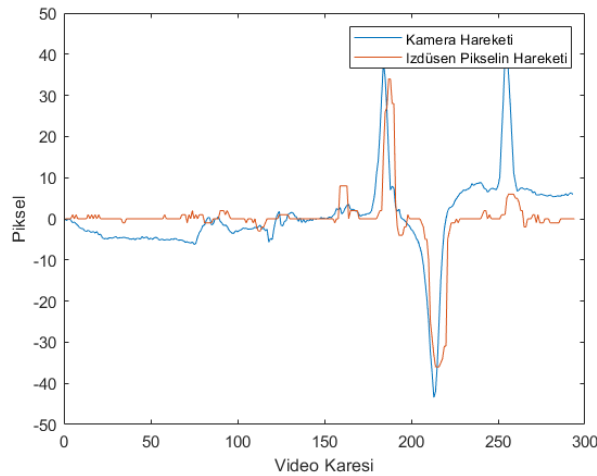
Öte yandan m/s cinsinden hesaplanan hız verisinin $piksel/s$ cinsine dönüştürüldükten sonra doğruluğunu koruduğu kolayca gösterilebilir. Buna göre uzaydaki sabit bir noktanın hareketli kamera ile görüntüsü çekildiğinde izdüşen görüntüye ait pikselin kameranın hareketine ters yönde yer değiştirdiği görülür. Eğer kamera hareketi $piksel/s$ cinsinden doğru bir şekilde hesaplanabiliyor ise, bu pikselin hareketinin $piksel/s$ cinsinden değeri ile kamera hareketinin $piksel/s$ cinsinden değerinin aynı büyüklükte ancak farklı yönde olması beklenir. Şekil 3.4’de örnek bir video için kamera hareketinin $piksel/s$ cinsinden hızı ile izdüşen pikselin ekrandaki yer değiştirmesinin $piksel/s$ cinsinden değişiminin uyumluluğu görülmektedir. Normalde kamera hareketi ile piksel hareketi ters yönlüdür ancak ikisi arasındaki uyumun daha rahat görülebilmesi için pikselin hareketi ters yönlü olarak çizdirilmiştir.

Bu noktada, daha önce detaylarıyla yer verilen yerçekimi ivmesinin yarattığı olumsuzluklardan bahsetmek gerekmektedir. Anlatıldığı üzere yerçekimi ivmesi sürekli olarak ivmeölçere etki etmekte ve bu etki büyük ölçüde filtrelense de tamamen sönümlenmemektedir. Öte yandan ivme verisinin hız verisine dönüştürülmesi sırasında kullanılan ayrık integral işleminin birikimli toplamaya karşı geldiğinden de daha önce bahsedilmiştir.



Şekil 3.4 : Ölçümlerle hesaplanan kamera ve izdüsen bir imge pikselinin hareketlerinin zaman içerisindeki değişimi.

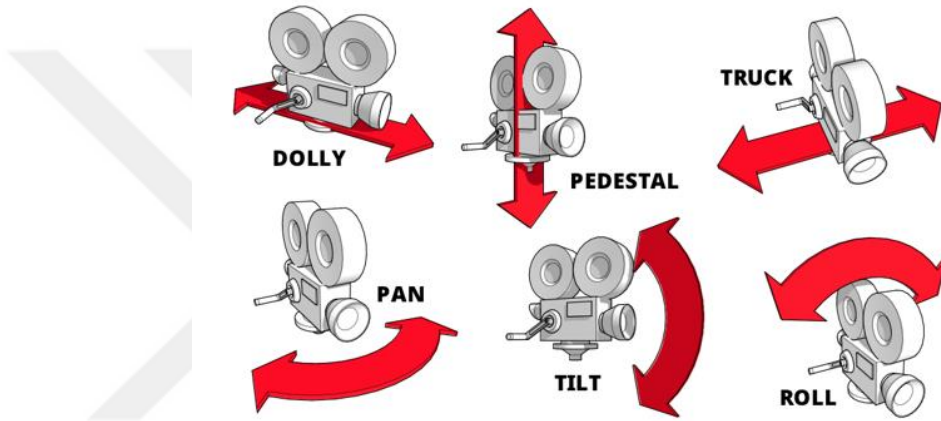
Bu ikisi birlikte düşünüldüğünde yerçekimi ivmesinin yarattığı bozucu etkinin birikimli toplama ile hıza dönüştürülürken gittikçe büyümesi beklenir. Bu olgu Şekil 3.4'teki ölçümlerin farklı eksenler için tekrarlanması ile elde edilen veriler kullanılarak Şekil 3.5'te gösterilmiştir. Beklendiği üzere, videonun ilk karelerinde yer çekiminin yarattığı bozucu etki göz ardı edilebilecek seviyelerdeyken ilerleyen karelere doğru bu etki kayda değer bir hal almıştır. Bu bozucu etki nesne izleme performansının arttırılmasında olumsuz bir etkiye sahiptir. Bu bozucu etkinin Şekil 3.4'de görülmeyip Şekil 3.5'de görülmesinin sebebi ise bu videolar çekilirken akıllı telefonun hangi pozisyonda tutulduğudur. Daha önce yerçekimi ivmesinin akıllı telefonun tutulduğu pozisyona göre bir ya da birden fazla eksende bozucu etkiye neden olduğuna değinilmiştir.



Şekil 3.5 : Yerçekimi ivmesinin ilerleyen video karelerinde yarattığı bozucu etki.

3.2 Afin Kamera Hareketi Parametrelerinin Hesaplanması

Literatürde tanımlı temel kamera hareketleri Şekil 3.6’da görülmektedir. Bu hareketlerden “truck” ve “pedestal” kameranın sağa ve sola t_x ve t_y kadar yer değiştirme hareketi yapması nedeniyle optik ekseninin hareket yönünde değişmesine karşı düşmektedir. “Tilt” ve “pan” sırasıyla kameranın x ve y eksenini çevresinde θ_x ve θ_y kadar dönmesine, “roll” z eksenini çevresinde θ_z kadar dönmesine ve “dolly” ise kameranın nesnelere yaklaşıp uzaklaşarak ρ ölçekli “zoom” hareketi yapmasına karşı düşmektedir.



Şekil 3.6 : Temel kamera hareketleri [30].

Denklem 3.4’te perspektif projeksiyon varsayımı altında, yer değiştirme, “pan”, “tilt”, “dolly”, ve “roll” şeklinde bir dizilişe sahip kamera hareketlerinden kaynaklanan 2 boyutlu afin hareket modellenmektedir. Denklemde video karesindeki bir pikselinin kamera hareketinden önce (x, y) olan konumu, kamera hareketinden sonra (x', y') konumuna gitmektedir. F kamera odak uzaklığını göstermektedir ve sabit kabul edilir.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \rho \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) \\ \sin(\theta_z) & \cos(\theta_z) \end{bmatrix} \begin{bmatrix} x + \theta_y F + t_x \\ y - \theta_x F + t_y \end{bmatrix} \quad (3.4)$$

Kamera hareketlerinin tümünün aynı anda var olması durumunda gerçek kamera hareket parametrelerinin kestirimi 2 boyutlu hareketten 3 boyutlu harekete geçiş yapmayı ve çeşitli varsayımlar altında parametreleri öngörmeyi gerektirir. Ancak nesne tespiti problemi açısından kameranın hareketi ve nesnelerin hareketlerinin ayrı ayrı öngörülmesi gerekmemektedir, bu nedenle tez çalışması kapsamında literatürde

sıklıkla kullanılan varsayımlar altında hareket kestirimi yapılarak derinlikli nesne sezicinin performansının artırılması hedeflenmiştir. Öncelikle literatürde kabul edildiği üzere, x ve y eksenleri doğrultusundaki hareketler birbirinden bağımsız işleme sokulmaktadır. Öte yandan kamera hareketinin düzgün dağıldığı yani video karesindeki tüm piksellere eşit oranda etki ettiği varsayılmaktadır. Bir diğer varsayım, mobil kameranın nesnelerden yeterince uzak olduğu ve böylelikle nesne yüzeylerinin düzlemsel kabul edilebileceği şeklindedir. Özetle tez çalışması kapsamında afin hareket modeli kabul edilmektedir. Bu koşullar altında ivmeölçerden alınan hız bilgisi kullanılarak Altbölüm 3.1’de anlatıldığı şekilde kestirilen t_x ve t_y yer değiştirme parametreleri kullanılarak mobil alet kamera hareketi hesaplanabilmekte, kamera hareketi kompanse edilerek nesnenin bir sonraki video karesinde alacağı konum kestirilebilmektedir. İstenmesi durumunda izlenen nesnenin hareketli olup olmadığı ya da kameranın hareketli olup olmadığı çıkarımları yapılabilmektedir. Böylece her bir video karesi için SSD nesne sezicinin yaptığı konum tahminine alternatif olarak ivmeölçer verisine dayalı konum tahmini de yapılmış olur. Tez çalışmasında MobileNetv1 SSD ve ivmeölçer verisine dayalı nesne kestirim kararları tümleştirilerek nesne izleme başarımının artırılması önerilmektedir ve yapılan gerçekleştirme Bölüm 4’te anlatılmaktadır.

3.3 Hesaplanan Kamera Hareket Parametrelerinin Kestirim ile Doğrulanması

Bu alt başlıkta ivmeölçerden gelen veri yardımıyla hesaplanan kamera hareketi parametrelerinin hareket kestirimi ile hesaplanan kamera hareketi parametreleri ile uyumu irdelenmektedir. Amaçlanan, her iki sonucun da birbirine yakın olduğunun gösterilmesi böylelikle ivmeölçer verisinin doğru bir şekilde kullanıldığının ispatıdır.

Bu tez çalışmasında, gürültü ve bozulmalara karşı daha dayanıklı olması nedeniyle, optik akış hesaplama gibi yoğun hareket kestirim yöntemleri yerine öznitelik tabanlı hareket kestirimi kullanılmıştır. Bu amaçla önce Harris köşe tespit edici [31] kullanılarak ardışık video karelerindeki köşeler tespit edilmiştir ve köşe pikselleri öznitelik noktaları olarak belirlenmiştir. Harris köşe tespit edici resim üzerinde küçük bir pencere gezdirerek her iki ekseninde de en çok gri seviyesi değişimi veren bölgeleri bulmayı hedefler ve bu bölgeleri köşe olarak nitelendirir. Her bir pikseldeki gri seviyesi değişimi Denklem 3.5’te verilen eşitlik ile hesaplanmaktadır. Burada $E(u, v)$, (x, y) pikselindeki gri seviyesi değişimini, $\omega(x, y)$ pencere fonksiyonunu, $I(x, y)$

üzerinde işlem yapılan pikselin gri seviyesini, u ve v ise sırasıyla yatay ve düşey eksenler yönündeki yer değiştirmeyi belirtmektedir.

$$E(u, v) = \sum_{x,y} \omega(x, y) [I(x + u, y + v) - I(x, y)]^2 \quad (3.5)$$

Gri seviyesi değişiminin büyük olduğu piksellerin bulunması için $\sum_{x,y} [I(x + u, y + v) - I(x, y)]^2$ ifadesini en büyükleyen piksellerin belirlenmesi gerekmektedir. Bu amaçla bu ifade Taylor serisine açılır ve sadece ilk 3 terimi incelenerek ifade düzenlenirse, Denklem 3.6'daki ifade elde edilir. Buradaki I_x ve I_y sırasıyla video karesinin gri seviyesinin yatay ve düşey eksenlerindeki birinci dereceden türevlerini belirtmektedir.

$$[I(x + u, y + v) - I(x, y)]^2 \approx [I(x, y) + uI_x + vI_y - I(x, y)]^2 \quad (3.6)$$

Denklem 3.6'da bulunan bu ifade Denklem 3.5'teki ifadede yerine koyulur ve gerekli düzenlemeler yapılırsa Denklem 3.7'deki ifade elde edilir.

$$E(u, v) = \sum_{x,y} \omega(x, y) (u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2) \quad (3.7)$$

Denklem 3.7 matris formunda yeniden düzenlenirse Denklem 3.8'deki gösterim elde edilir.

$$E(u, v) \approx [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad M = \sum_{x,y} \omega(x, y) \begin{bmatrix} I_x & I_x \\ I_x & I_y \\ I_y & I_y \end{bmatrix} \quad (3.8)$$

Denklem 3.8'de $E(u, v)$ artık M matrisi cinsinden yaklaşık olarak ifade edilmiştir ve M matrisi o anki video karesinin her iki eksenindeki birinci dereceden türevlerinden oluşmaktadır. $E(u, v)$ video karesinin her pikseli için ayrı ayrı hesaplandığından hangi piksellerin köşe olarak kabul edileceğine karar vermek için Denklem 3.9'daki eşitlik kullanılarak her bir piksele bir skor atamaya dayanan yöntem önerilmiştir [31]. Burada R ilgili köşe adayına atanan skora karşı gelmektedir ve k ise bir sabit olup serbest parametredir.

$$R = \det(M) - k [iz(M)]^2 \quad (3.9)$$

Denklem 3.9'daki skor atama ifadesi M matrisinin öz değerleri olan λ_1 ve λ_2 cinsinden tekrar düzenlenirse Denklem 3.10'daki ifade elde edilmiş olur.

$$R = \lambda_1 \lambda_2 - k (\lambda_1 + \lambda_2)^2 \quad (3.10)$$

Denklem 3.10 kullanılarak resmin her bir pikseli için bir skor atandıktan sonra skoru belirli bir eşik değerinden büyük olan pikseller için M matrisinin öz değerleri kontrol edilir. Öz değerler birbirine yakın ise piksel köşe olarak, öz değerlerin arasındaki fark çok büyük ise piksel kenar olarak kabul edilir.

Her video karesindeki köşeler bulunduktan sonra bir sonraki video karesinde aynı köşeye karşı düşen piksellerin birbiri ile eşleştirilebilmesi için bu köşelere ait özniteliklerin çıkarılması gerekmektedir. Bu tez çalışmasında FREAK [32] tanımlayıcı kullanılarak her bir köşe noktasının belirli bir büyüklükteki komşuluğu incelenmekte ve köşe noktasının farklı yönlerdeki komşu piksellerle ilişkisini modelleyen bir öznitelik vektörü çıkarılmaktadır.

FREAK tanımlayıcı insan retinasını taklit ederek öncelikle incelenmek istenilen nokta etrafında dairesel ve belirli bir örüntüye sahip ilgi alanı çiftleri oluşturur. Ardından bu ilgi alanı çiftlerini kullanarak incelenmek istenilen noktanın oryantasyonuna dayalı bir öznitelik vektörü hesaplar. Bu tez çalışmasında, her bir köşe noktası için FREAK tanımlayıcının önerdiği 43 adet ilgi alanı kullanılarak 512 bitlik ikili düzene bir sahip öznitelik vektörü çıkarılmaktadır. Denklem 3.11'de formüle edilen F FREAK tanımlayıcı tarafından oluşturulan ikili düzendeki öznitelik vektörünü, N öznitelik vektörünün uzunluğunu, P_α ilgi alanı çiftini, $P_\alpha^{r_i}$ ilgi alanı çiftindeki her bir ilgi alanını ve $I(P_\alpha^i)$ ise ilgi alanının yumuşatılmış gri seviyesini belirtir.

$$F = \sum_{0 \leq \alpha < N} 2^\alpha T(P_\alpha), \quad T(P_\alpha) = \begin{cases} 1, & I(P_\alpha^{r_1}) - I(P_\alpha^{r_2}) > 0 \\ 0, & \text{aksi halde} \end{cases} \quad (3.11)$$

Tez kapsamında kullanılan hareket kestirimi yönteminde Harris köşe sezici ile çıkarılan her bir köşe noktası için FREAK tanımlayıcı vektörleri hesaplanmaktadır. İki ardışık video karesindeki köşe noktaları bu öznitelikler kullanılarak eşlenmektedir. Bu sayede bir köşe noktasının bir sonraki karede nereye gittiği hesaplanabilmektedir. FREAK tanımlayıcı tarafından çıkarılan öznitelikler ikili sistemde oldukları için eşlemede benzerlik ölçütü olarak Hamming uzaklığı kullanılmıştır. İkili düzende

Hamming uzaklığı karşılaştırılmak istenilen iki vektörün karşılıklı bitlerinin kaç tanesinin farklı olduğuna karşı gelir. Şekil 3.7’de iki ardışık video karesi arasında eşleştirilmiş köşeler gösterilmiştir. Bu gösterimde kırmızı renkli nesneler o nesnenin bir önceki video karesindeki konumunu göstermektedir. Kırmızı çemberler birbiri ile eşleşen köşelerin bir önceki video karesindeki konumunu, sarı artı işaretleri köşelerin mevcut video karesindeki konumunu göstermektedir. Öte yandan Şekil 3.7’deki bu gösterimin aynı zamanda iki ardışık video karesi arasındaki öznitelik noktalarında hesaplanan hareket vektörlerine de karşı düştüğünü belirtmek gerekir. Buna göre, kırmızı çemberler ilgili vektörün başını, sarı artı işaretleri ise vektörün ucunu göstermektedir.



Şekil 3.7 : İki ardışık video karesi arasındaki eşleştirilmiş köşeler.

Eşleştirme işleminin ardından, her bir video karesi için eşleştirilen köşelerin iki ardışık video karesi arasındaki yer değiştirmeleri incelenerek, afın hareket varsayımı altında, kamera hareket parametreleri bulunabilir. Bu amaçla öncelikle ardışık video karelerindeki eşleştirilmiş köşe noktalarının koordinatları ortak referansa çekilir ki bu işlem merkezileştirme olarak adlandırılmaktadır. Merkezileştirme işlemi Denklem 3.12’de görüldüğü gibi her köşe noktasının koordinatlarından tüm köşe noktalarının ağırlık merkezinin koordinatlarının çıkarılması ile yapılır. Burada $C_c(x, y)$ merkezleştirilmiş köşe koordinatlarına, $C(x, y)$ köşe koordinatlarına ve $M(x, y)$ de o video karesindeki tüm köşelerin ağırlık merkezinin koordinatlarına karşı gelmektedir.

$$C_c(x, y) = C(x, y) - M(x, y) \quad (3.12)$$

Ardından merkezleştirilen köşelerin koordinatları bir ölçeklendirme çarpanı ile normalize edilerek normalize köşe koordinatları bulunur. Denklem 3.13’te bu ölçek çarpanının nasıl hesaplandığı görülmektedir. Buradaki β ölçek çarpanına, K o video

karesindeki köşe sayısına, $C_c(x, y)$ merkezleştirilmiş köşe koordinatlarına ve $M(x, y)$ ise ağırlık merkezinin koordinatlarına karşı düşmektedir.

$$\beta = \frac{\sqrt{2K}}{\sqrt{\sum \|C_c(x, y) - M(x, y)\|^2}} \quad (3.13)$$

Yukarıda belirtilen normalizasyon işleminin ardından Denklem 3.4'te verilen afin hareket denklemleri en küçük kareler yöntemi kullanılarak çözülebilir. Denklem 3.4'teki ifadenin farklı katsayıları sahip olmaları nedeniyle her bir satır için en küçük kareler yöntemi ile ayrı ayrı çözülmesi gerekmektedir. Bu durumda her iki denklemde de ortak olan ve kameranın z - eksenini etrafındaki dönme açısına karşı düşen q_z açısının en küçük kareler yönteminin doğası gereği her iki çözümde de birbirine yakın ama birbirinden farklı olarak bulunması beklenmektedir. Bunun sebebi en küçük kareler yönteminin hatayı minimize etmesi ancak sıfırlamamasıdır.

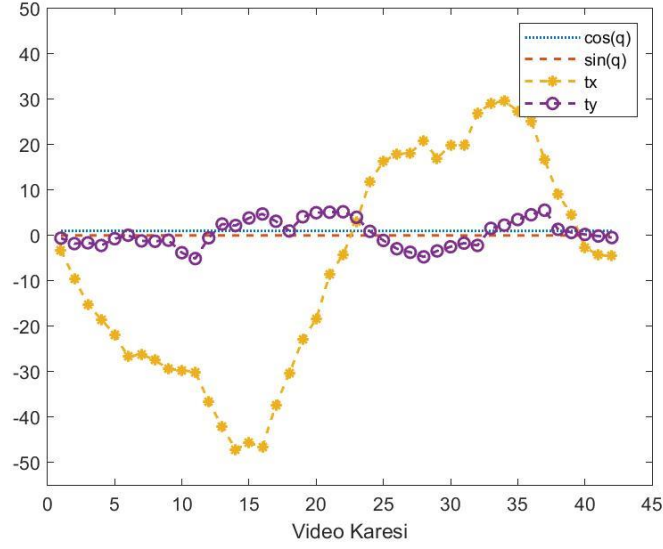
Denklem 3.4'in ilk satırı en küçük kareler yöntemi kullanılarak n tane eşleştirilmiş köşe çifti için Denklem 3.14'teki gibi çözülebilir.

$$\begin{bmatrix} \cos(q) \\ \sin(q) \\ t_x \end{bmatrix} = (X^T X)^{-1} X^T \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \quad (3.14)$$

Denklem 3.4'ün ikinci satırı da benzer şekilde en küçük kareler yöntemi ile çözüldükten sonra, bulunan bu parametreler normalize koordinatlar ile bulunduğu için kullanılmadan önce denormalizasyondan geçirilir.

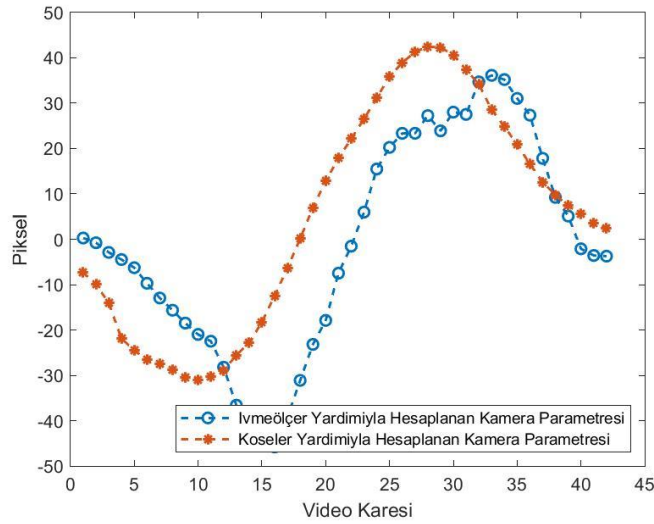
Örnek bir video için yukarıda belirtildiği gibi hesaplanan kamera hareketi parametreleri Şekil 3.8'de gösterilmiştir. Şekil 3.8 incelenirse kamera açısının kosinüsü 1 civarında ve sinüsü ise 0 civarında değerler almaktadır. Bu durum kameranın bu videoda dönme hareketi yapmadığını gösterir. Öte yandan kamera y-ekseni yönünde küçük yer değiştirme hareketleri yaparken x-ekseni yönünde ise her iki yönde ciddi kaymalar yapmıştır.

Yukarıdaki paragraflarda anlatıldığı şekilde hesaplanan kamera hareketinin altbölüm 3.1'de açıklanan ivmeölçerden gelen veri kullanılarak hesaplanan kamera hareketi ile uyumlu olduğunun irdelenmesini yapmak amacıyla her iki hareket birlikte incelenmiştir.



Şekil 3.8 : Örnek bir video için hesaplanan kamera hareketi parametreleri.

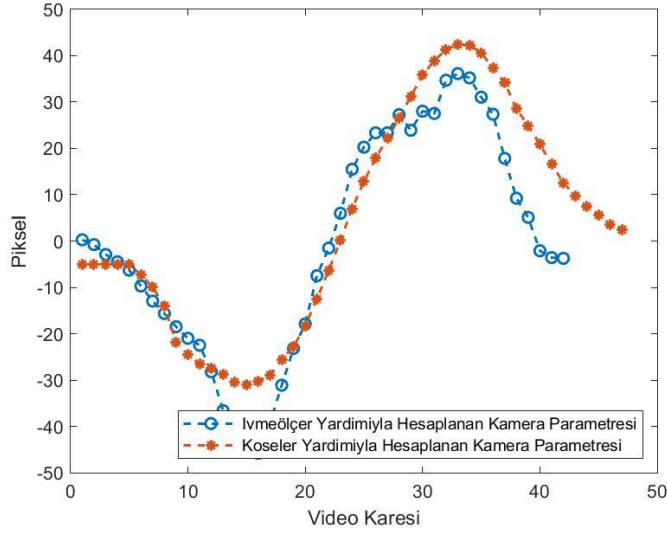
Şekil 3.9’da örnek bir video için eşleşen köşeler kullanılarak hesaplanan kamera hareketi parametreleri ile Denklem 3.6 kullanılarak ivmeölçerden alınan veri yardımıyla hesaplanan kamera parametrelerinin t_x bileşenlerinin video karesi boyunca değişimi görülmektedir.



Şekil 3.9 : Köşe eşleştirme ve ivmeölçer kullanılarak hesaplanan kamera yer değişimi hareketi parametresinin video boyunca değişimi.

Şekil 3.9 incelendiğinde eşleşen köşeler yardımıyla hesaplanan kamera hareketi parametreleri ile ivmeölçerden gelen veri yardımıyla hesaplanan kamera hareketi parametreleri arasında yatay ekseninde bir miktar kayma olduğu görülmektedir. Bu kayma miktarı 5 video karesi kadardır ve bunun zaman cinsinden karşılığı yaklaşık

olarak 250 milisaniye kadardır. Bu kaymanın sebebi, video kaydı sırasında kameradan alınan verinin ivmeölçerden alınan veriye kıyasla daha fazla gecikmeyle alınabilmesidir. İki farklı yöntemle hesaplanan bu parametrelerin uyumunun daha iyi görülebilmesi adına Şekil 3.10'da bu iki parametre eş zamanlı olarak tekrar çizdirilmiştir. Ayrıca bu tez çalışmasının pratik gerçekleştirilmesi sırasında bu gecikmeye dikkat edildiğinin de altını çizmek gerekir.



Şekil 3.10 : İki farklı yöntemle hesaplanan kamera hareketi yer değiştirme parametresinin eş zamanlı çizimi.

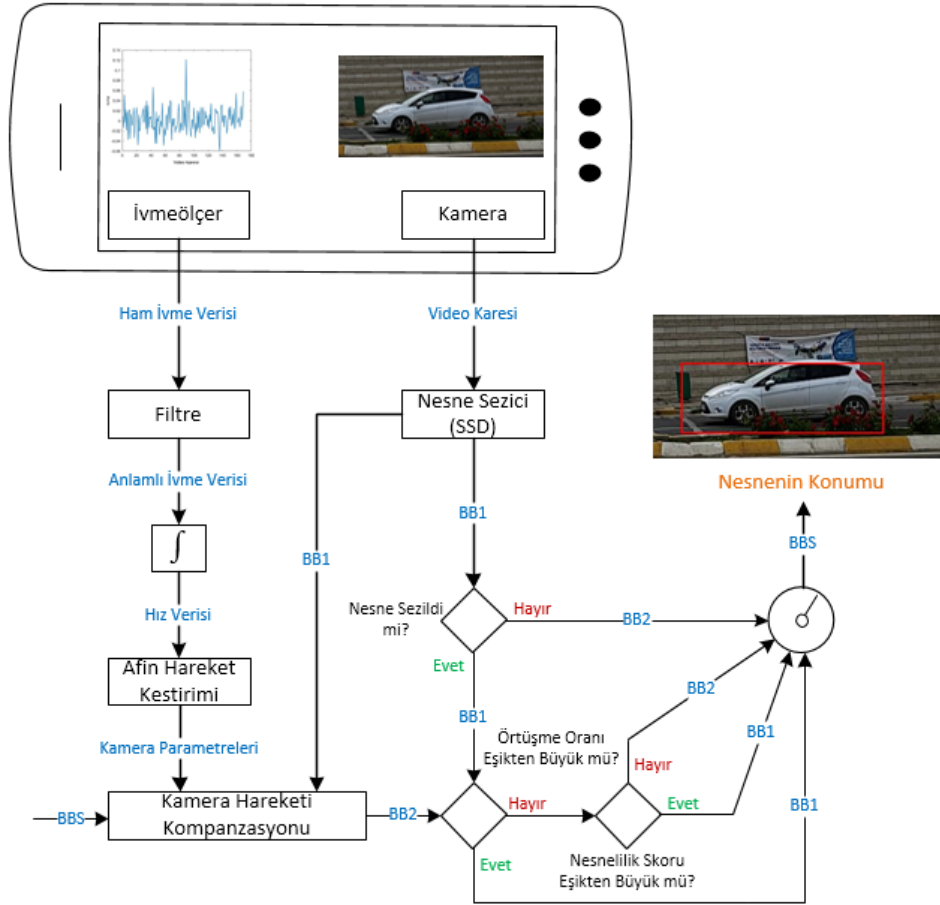
4. İVMEÖLÇER GERİBESLEMESİ İLE SSD NESNE İZLEME

Bir önceki bölümde ivmeölçerden gelen veri kullanılarak kamera hareketi parametrelerinin nasıl modellendiği ve bu parametreler ile nesnenin konumunun nasıl tahmin edildiği anlatılmıştır. Bu bölümde hem ivmeölçere dayalı konum tahmini hem de SSD nesne seziciye dayalı konum tahmini kullanılarak nesne izleme performansının artırılabilceği gösterilmektedir. Alt bölüm 4.1’de nesne izleme performansının nasıl artırıldığına detayları ele alınmaktadır. Alt bölüm 4.2’de bu amaçla geliştirilen Android uygulamasına ilişkin detaylar verilmektedir.

4.1 Kamera Hareketi Kullanılarak Nesne İzleyici Performansının Arttırılması

Daha önce de belirtildiği gibi, izlenen nesnenin konumu hakkında her bir video karesi için birisi ivmeölçerden gelen verilerle, diğeri nesne sezici tarafından hesaplanan iki adet konum tahmini yapılmaktadır. Şekil 4.1’de bu iki adet konum tahminini birleştirerek nesne izleme performansının arttırılması amacıyla gerçekleştirilen algoritma bir akış diyagramı ile açıklanmaktadır. Bu kapsamda birbirine paralel ve eş zamanlı olarak hem 2. Bölümde anlatıldığı gibi MobileNetv1 SSD kullanılarak hem de 3. Bölümde anlatıldığı gibi ivmeölçer verisi kullanılarak iki adet konum tahmini yapılır. Bu iki konum tahminini simgeleyen dikdörtgensel ilgi alanlarının (BB) birbiri ile olan örtüşme oranları (IoU) belirli bir eşik değerinden büyük veya eşit ise MobileNetv1 SSD’ye ve ivmeölçere dayalı konum tahminlerinin birbiri ile uyumlu olduğu kabul edilir ve MobileNetv1 SSD tarafından yapılan tahmin nesnenin gerçek konumu olarak raporlanır. Aksi halde iki konum tahmini arasında bir uyumdan söz edilemez ve bu durumda MobileNetv1 SSD tarafından üretilen nesnelilik skoru karar kriteri olarak kullanılır. Nesnelilik skoru belirli bir eşik değerinden büyük veya eşit ise MobileNetv1 SSD tarafından yapılan tahmin gerçek nesne konumu olarak kabul edilir. Aksi halde MobileNetv1 SSD’nin hatalı sonuç verdiği kabul edilerek ivmeölçere dayanarak yapılan konum tahmini kullanılır. Özel bir durum olarak MobileNetv1 SSD’nin

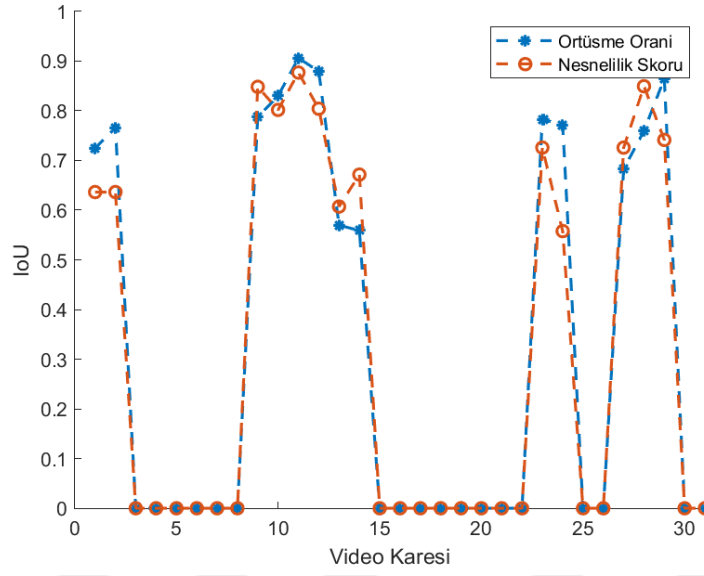
nesneyi tamamen kaybettiği durumda ise nesnenin bilinen son konumu kullanılarak ivmeölçere dayalı konum tahmini nesnenin gerçek konumu olarak kabul edilir.



Şekil 4.1 : Nesne izleme performansının artırılmasına dair akış diyagramı.

Şekil 4.1’de görüldüğü üzere, karar aşamasında kullanılan kriterlerden birisi MobileNetv1 SSD tarafından nesne BB’si için üretilen nesnelilik skorudur. Nesnelilik skoru, MobileNetv1 SSD derin öğrenme ağının nesne BB’si olarak raporlanan BB’de izlenen nesneyi hangi olasılıkla bulduğunu belirleyen ve mAP kestirim sonucunda elde edilen koşullu olasılıktır. Bu nedenle karar tümleştirmede bir karar kriteri olarak kullanılması önerilmektedir. Şekil 4.2’de örnek bir videodaki art arda 31 video karesi SSD tarafından hesaplanan nesnelilik skorunun değişimi, nesnenin sezici tarafından tahmin edilen konumu ile gerçek konumu arasındaki örtüşme oranı (IoU) ile üst üste çizilen eğrilerle raporlanmıştır. Görüldüğü gibi nesnelilik skoru ve nesnenin ne kadar doğru sezildiğini gösteren örtüşme oranı arasında bir uyum mevcuttur. Öte yandan bazı video karelerinde nesne izleyici nesneyi kaybettiği için her iki değer de sıfırdır. Şekil 4.2’deki çizim için nesne sezicinin nesneyi takip edebildiği video karelerindeki

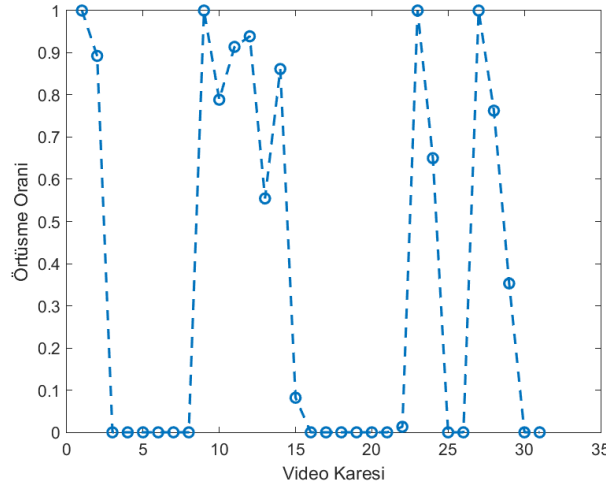
nesnelilik skorunun ortalaması bu örnek video için yaklaşık olarak 0.70 civarındadır. Aynı deney toplam 357 video karesi içeren iki farklı videoda daha tekrarlandığında nesnelilik skoru ortalaması 0.67 olarak hesaplanmıştır. Bu veriler ışığında, Şekil 4.1’de de gösterilen nesnelilik skoru eşik değeri aksi belirtilmedikçe 0.6 olarak alınmıştır.



Şekil 4.2 : Örnek bir videonun her bir karesindeki nesnelilik skoru ve örtüşme oranı.

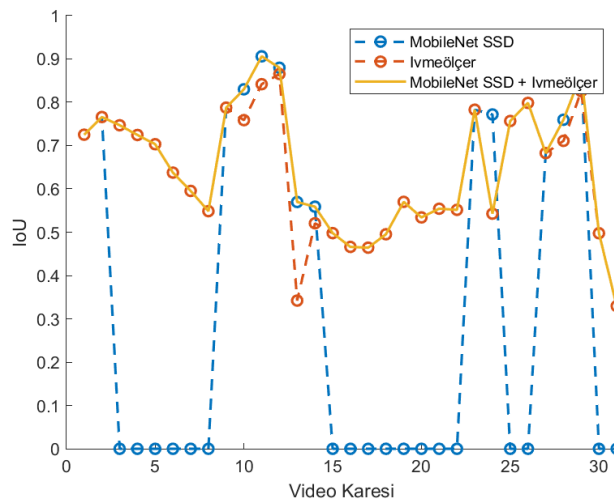
Şekil 4.1’de görüldüğü üzere karar aşamasında kullanılan bir diğer kriter ise MobileNetv1 SSD’ye ve ivmeölçere dayalı olarak yapılan iki konum tahmini arasındaki örtüşme oranıdır. İki tahmin arasındaki örtüşme oranının büyük olması tahminlerin birbiri ile uyumlu olduğunu, küçük olması ise tahminlerin birbiri ile çeliştiğini gösterir. Şekil 4.3’te aynı örnek video için nesne izleyici tarafından hesaplanan nesne konumu ile ivmeölçerden alınan veriye dayanarak hesaplanan nesne konumunun birbiri ile olan örtüşme oranının 31 video karesi boyunca değişimi görülmektedir. Nesnenin SSD MobileNetv1 tarafından kaybedildiği video karelerinde örtüşme oranı hesaplanamayacağı için, bu karelerdeki örtüşme oranı sıfır kabul edilmiştir. Bu örnek video için nesnenin kaybedilmediği video karelerindeki örtüşme oranı 0.72 olarak hesaplanmıştır. Aynı deney toplam 357 video karesi içeren iki farklı videoda daha tekrarlandığında örtüşme oranı ortalaması 0.92 olarak hesaplanmıştır.

Bu veriler ışığında Şekil 4.1’de görülen örtüşme oranı eşik değeri aksi belirtilmedikçe 0.7 olarak alınmıştır.



Şekil 4.3 : Örnek bir videonun her bir karesi için hesaplanan iki konum tahmininin birbiri ile olan örtüşme oranı.

Şekil 4.4’te karar tümleştirme öncesi ve sonrası elde nesne konumlarının nesnenin gerçek konumuyla olan örtüşme oranları, aynı 31 kareli videonun her bir karesi için çizdirilmiştir. Sadece nesne sezicinin kullanıldığı durum maviyle, sadece ivmeölçerden gelen veriye dayalı tahminin kullanıldığı durum kırmızıyla ve her iki tahminin Şekil 4.1’de gösterildiği gibi birleştirildiği durum sarı ile gösterilmiştir. Bu çizime ait detaylı sonuçlara tez çalışmasının 5. Bölümünde yer verilmiştir.



Şekil 4.4 : Önerilen yöntemin nesne izleme performansını gösteren bir örnek.

Buraya kadar yapılan tüm bu hesaplamalar örnek bir videonun ardışık kareleri üzerinde gösterilebilir. Şekil 4.5'te, Şekil 4.4'te kullanılan örnek videoya ait ardışık 4 video karesi görülmektedir. Bu videoda izlenmek istenilen nesne beyaz renkli arabadır. Mavi dikdörtgen nesnenin gerçek konumunu, yeşil dikdörtgen nesne sezici tarafından hesaplanan konum tahmininin ve kırmızı dikdörtgen ivmeölçerden gelen veriye dayanarak karar tümleştirme ile hesaplanan konum tahminini göstermektedir. 12. video karesinde her iki konum tahmini arasındaki örtüşme oranı 0.93 olup eşik değerden büyüktür, bu sebeple MobileNetv1 SSD tarafından yapılan konum tahmini doğru olarak kabul edilmiştir. 13. video karesinde örtüşme oranı 0.55 olup bu değer eşik değerinden küçüktür. Bu durumda karar kriteri olarak nesnelilik skoru kullanılmıştır. Nesnelilik skoruna göre 0.81 olasılıkla araba olup sınır değerinden büyük olduğu için MobileNetv1 SSD tarafından yapılan tahmin doğru kabul edilmiştir. 14. video karesinde örtüşme oranı 0.86 olup sınır değerinden büyüktür ve MobileNetv1 SSD tarafından yapılan konum tahmini doğru kabul edilmiştir. 15. video karesinde ise nesne MobileNetv1 SSD tarafından tamamen kaybedilmiştir. Bu sebeple Şekil 4.1'de de gösterildiği gibi ivmeölçer verisine dayanan konum tahmini doğru kabul edilmiştir.

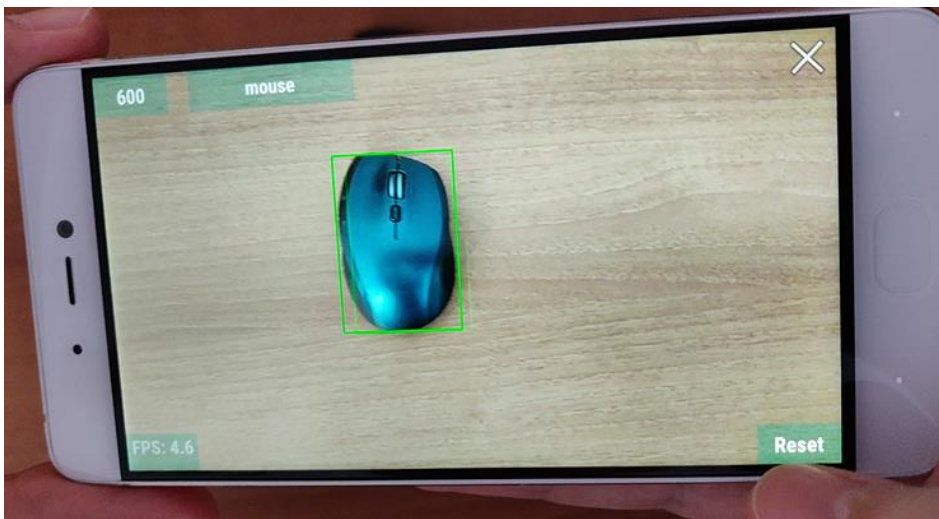


Şekil 4.5 : Örnek bir videonun 12, 13, 14 ve 15. kareleri.

4.2 Önerilen Yöntemin Android Ortamında Gerçeklenmesi

Bu alt başlık altında tez çalışması kapsamında hazırlanan mobil yazılımın detaylarına değinilecektir. Öncelikle, mobil yazılım Android [33] işletim sistemine sahip mobil telefonlar için Google ve JetBrains [34] şirketleri tarafından geliştirilen Android Studio [35] yazılım geliştirme ortamında yazılmıştır. Geliştirilen yazılımda ihtiyaç duyulan yazılım kütüphaneleri için açık kaynak kodlu OpenCV [36] yazılım geliştirme kitinden yararlanılmıştır. Yöntemsel olarak OpenCV'nin 4.0.1 kodlu sürümünün Android işletim sistemi için Java yazılım dili ile hazırlanan 27 kodlu sürümü (Android 8.1 Oreo) [37] kullanılmıştır. Yine geliştirilen yazılımda kullanılmak üzere Caffe [38] ortamı hedeflenerek hazırlanmış ve önceden eğitilmiş bir model dosyası [39] kullanılmıştır. Bu model dosyası tezin hedeflerine uygun olarak, SSD ve MobileNetv1'n birleşiminden elde edilen yapay sinir ağının eğitilmesiyle elde edilmiştir. Bu yapay sinir ağının detaylarına 2. Başlıkta değinilmiştir.

Tez çalışması kapsamında geliştirilen ve gerçek zamanlı çalışan Android uygulamasının APK dosyasının boyutu yaklaşık olarak 99 MB'tır. Bu Android uygulamasına ait örnek bir ekran görüntüsü Şekil 4.8'de görülmektedir. Sol altta gerçek zamanlı olarak kaç video karesinin işlendiği görülmektedir. Sol üstte sırasıyla Denklem 3.5'te kullanılan γ parametresi ve takip edilen nesnenin ait olduğu sınıf görülmektedir. Sağ altta uygulamayı yeniden başlatma butonu ve sağ üstte uygulamadan çıkış butonu görülmektedir.



Şekil 4.6 : Geliştirilen Android uygulamanın çalışırken görüntüsü.

Bu gerçek zamanlı uygulamada Şekil 4.7'deki sözde kodun gerçekleşmesi hedeflenmiştir. Buna göre öncelikle kameradan yeni bir video karesi okunur ve bu video karesi üzerinde Bölüm 2'de anlatıldığı gibi MobileNetv1 SSD ile nesne sezimi yapılarak ilk konum tahmini yapılır. Ardından ivme verisi okunur, bu veri Altbölüm 3.1'de anlatıldığı gibi filtrelenir ve ayrık integrali alınarak hız verisi elde edilir. Elde edilen bu hız verisi kullanılarak Altbölüm 3.2'de anlatıldığı gibi önce kamera hareketi parametreleri hesaplanır, daha sonra kamera hareketi kompanze edilerek ikinci konum tahmini yapılır. Son olarak ise Altbölüm 4.1'de anlatıldığı gibi iki konum tahmini arasında örtüşme oranı ve nesne skoruna göre bir seçim yapılarak nesnenin konumu belirlenir. Ancak bu hedef tam olarak tutturulamamış ve hedeflenene benzer bir gerçek zamanlı uygulama gerçekleşmiştir. Hedeflenen ve gerçekleşen gerçek zamanlı uygulamalar arasındaki temel fark gerçekleşen uygulamada ivmeölçer verisine dayanarak nesne tespitinin sadece MobileNetv1 SSD'nin nesneyi tamamen kaybettiği durumlarda yapılmasıdır.

```
her bir video karesi için
    video_karesi ← Oku(kamera)
    konum_1, nesne_skoru ← MobileNet_SSD(video_karesi)
    ham_ivme ← Oku(ivmeölçer)
    anlamlı_ivme ← Filtrele(ham_ivme)
    hız ← hız + anlamlı_ivme
    kamera_param ← Afin_Parametreleri_Hesapla(hız)
    konum_2 ← Kompanze_Et(kamera_param)
    eğer Örtüşme_Oranı(konum_1, konum_2) > eşik değeri
        nesne_konumu ← konum_1
    değilse
        eğer nesne_skoru > eşik değeri
            nesne_konumu ← konum_1
        değilse
            nesne_konumu ← konum_2
son
```

Şekil 4.7 : Geliştirilmesi hedeflenen Android uygulamasına ait sözde kod.

Tez çalışması sırasında önerilen yöntemin aynı örnek videoda tekrar tekrar denenebilmesi ve geliştirilebilmesi için ayrıca bir çift çevrimdışı çalışan Android uygulaması da geliştirilmiştir. Bu uygulama çiftinden ilki hem .avi formatında video

kaydetmekte hem de ivmeölçerden aldığı veriyi video kareleriyle eş zamanlı bir şekilde .txt formatındaki ayrı bir dosyaya her eksen için kaydetmektedir, zenginleştirilmiş video kaydı yapmamaktadır. Bu uygulamaya ait APK dosyasının boyutu yaklaşık olarak 82 MB'tır. İkinci uygulama, ilk uygulama ile önceden kaydedilmiş videoları kullanarak MobileNetv1 SSD ile nesne sezme yapmaktadır. Bu uygulamaya ait APK dosyasının boyutu yaklaşık olarak 67 MB'tır. Bu iki uygulamadan elde edilen sonuçlar, Matlab ortamında Şekil 4.1'deki gibi birleştirilmiştir. Sonuç olarak tez çalışmasında önerilen yöntem çevrimdışı olarak tam anlamıyla gerçekleşmişken gerçek zamanlı olarak ise kısmen gerçekleştirilebilmiştir.



5. PERFORMANS TEST SONUÇLARI

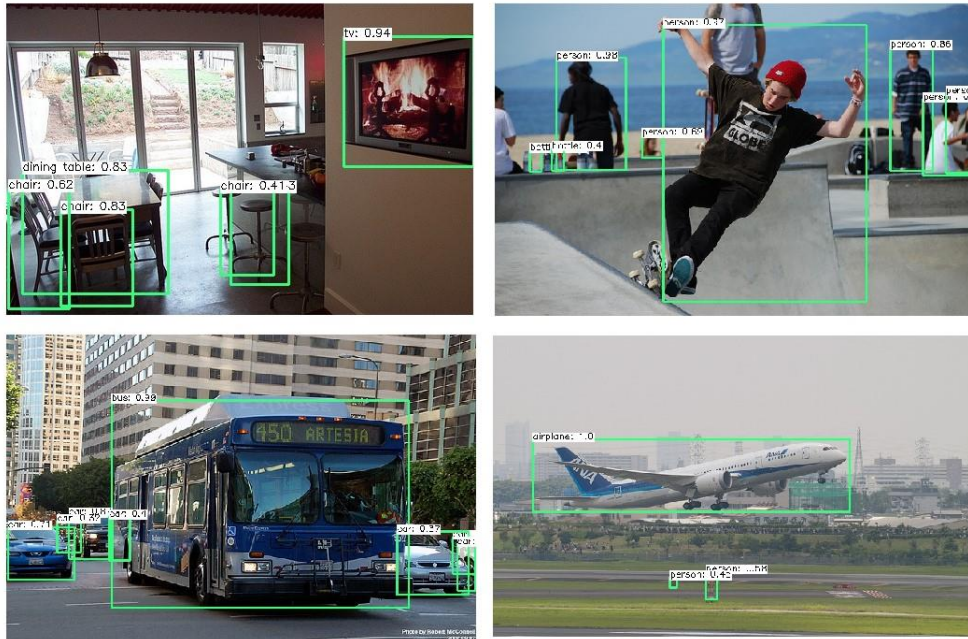
Bir önceki başlık altında bu tez çalışmasında kullanılan yöntemlerin hangi ortamda ve nasıl geliştirildiğinin detayları verilmiştir. Bu başlıkta bu yöntemlerin performans analizlerine yer verilmekte olup bu analizler irdelenecek, iyi ve kötü yanları ortaya koyulmaktadır.

5.1 Eğitilmiş Model Dosyası İle Eğitim Seti Üzerinde Elde Edilen Başarım

Testlerde ilk olarak SSD nesne sezicinin MobileNet mimarisi kullanması sonucu geliştirilen nesne tespit yönteminin öğrendiği nesneleri tespit performansı raporlanarak elde edilebilecek enyüksek doğruluk irdelenmiştir. Bu amaçla nesne sezicinin 80 farklı nesneyi tespit etmek üzere eğitiminde kullanılan MS COCO 2014 [40] veri seti üzerinde performans raporlanmıştır. MS COCO veri setleri 2014'ten bu yana her yıl düzenlenen çeşitli dallardaki MS COCO yarışmalarında kullanılmak üzere günlük hayatın içinde çekilen resimlerden oluşturulmuştur [40]. Bu veri setinde eğitim, doğrulama ve test için ayrı ayrı gruplanmış resimler olup sadece eğitim ve doğrulama gruplarındaki resimler için gerçek nesne pozisyonlarını gösteren BB koordinatları verilmiştir. Bu sebeple, bu bölümde raporlanacak olan tüm performans ölçümleri MS COCO 2014 veri setinin eğitim ve doğrulama gruplarındaki resimlerin birlikte kullanılmasıyla elde edilmiştir. Eğitim setinde 82081 ve doğrulama setinde 40137 olmak üzere toplam 122218 resim bulunmaktadır. Bu resimler toplamda 80 farklı sınıftan nesneler içermektedir. Görsel çıkarım yapılabilmesi amacıyla, Şekil 5.1'de MS COCO veri setinden rasgele seçilen 4 resim üzerinde SSD Mobile nesne tespit sonuçları nesnelilik skorları ve BB'ler verilerek raporlanmaktadır.

Genel sınıflandırma problemlerinde olduğu gibi nesne tespitinde de performans ölçümü amacıyla öncelikle karıştırma matrisi elde edilir ve bu matrsten her bir nesne sınıfı için TP ("true positive"), TN ("true negative"), FP ("false positive") ve FN ("false negative") değerlerine erişilir. TP nesnenin doğru tespit edilmesi, TN nesne olmayan yerde nesne tespit etmeme, FN gerçekte nesne olmadığı halde yapılan hatalı

tespit ve FP ise gerçekte nesne olmayan bir yerin nesne olarak tespit edilmesi yani yanlış alarm durumudur.



Şekil 5.1 : MS COCO 2014 veri setinden rastgele seçilmiş 4 resim üzerindeki nesne tespit sonuçları.

Nesne tespitinin genel sınıflandırma problemlerinden farkı ise bir nesnenin doğru tespit edildiğine nasıl karar verileceğidir. Genel sınıflandırmada nesnenin ait olduğu sınıfı doğru tahmin etmek yeterlidir. Tez kapsamında kullanılan derin öğrenmeye dayalı nesne tespit ediciler sınıf etiketini belli bir olasılıkla belirler ve bu değer nesnelilik skoru (“objectness score”) olarak raporlanır. Tez kapsamında nesne izleme ile ilgilenildiğinden nesnenin ait olduğu sınıfla birlikte resim içindeki konumunun doğru tespit edilmesi de önemlidir. Bu sebeple, nesnenin doğru tespit edildiğine karar verebilmek için nesnenin hem doğru sınıflandırılmış olması hem de konumunun doğru tahmin edilmiş olması gerekmektedir.

Nesne konumu hakkındaki tahminin doğruluğunu sınamak için kullanılabilecek hata ölçümlerinden birisi merkez piksel kayma hatasıdır. Merkez piksel kayma hatası, nesneyi çevreleyen kutunun (BB) merkezinin gerçek konumu ile tahmin edilen konumu arasındaki uzaklığın piksel cinsinden ifadesidir [41]. Burada uzaklık ölçümü için değişik uzaklık çeşitleri kullanılabilir ama bu tez çalışmasında literatüre uygun olarak Öklid uzaklığı kullanılmıştır. Eğer gerçek ve tahmin edilen merkezler

arasındaki uzaklık önceden belirlenmiş bir değerden küçük ise merkez piksel kayma hatası kriterine göre o nesnenin konumu doğru tespit edilmiş olarak işlem görür.

Merkez piksel kayma hatası sadece nesnenin gerçek ve tahmin edilen merkezleri arasındaki mesafeye odaklanmıştır, nesnenin gerçek ve tahmin edilen ebatlarını göz önüne almaz. Bu sebeple pek çok durumda yeterli bir karar kriteri değildir. Bu duruma çözüm olarak IoU yani örtüşme oranı kullanılmaktadır. Nesne tespitinde nesnenin hem gerçek hem de tahmin edilen konumları BB ile ifade edilir. Nesnenin gerçek konumunu belirten BB ile tahmin edilen konumunu belirten BB'nin kesişimlerinin alanının birleşimlerinin alanına oranına örtüşme oranı yani IoU denir [41]. Eğer gerçek ve tahmin edilen dikdörtgenlerin örtüşme oranı önceden belirlenmiş bir değerden büyük ise örtüşme oranı kriterine göre o nesne doğru tespit edilmiş olarak işlem görür. Örtüşme oranı kriterinde nesnenin gerçek ve tahmin edilen ebatları da göz önünde bulundurulduğu için merkez piksel kayma hata kriterine göre daha doğru bir ölçüm aracı olarak kabul edilebilir. Bu tez çalışmasında da aksi belirtilmediği sürece hata kriteri olarak örtüşme oranı raporlanmaktadır.

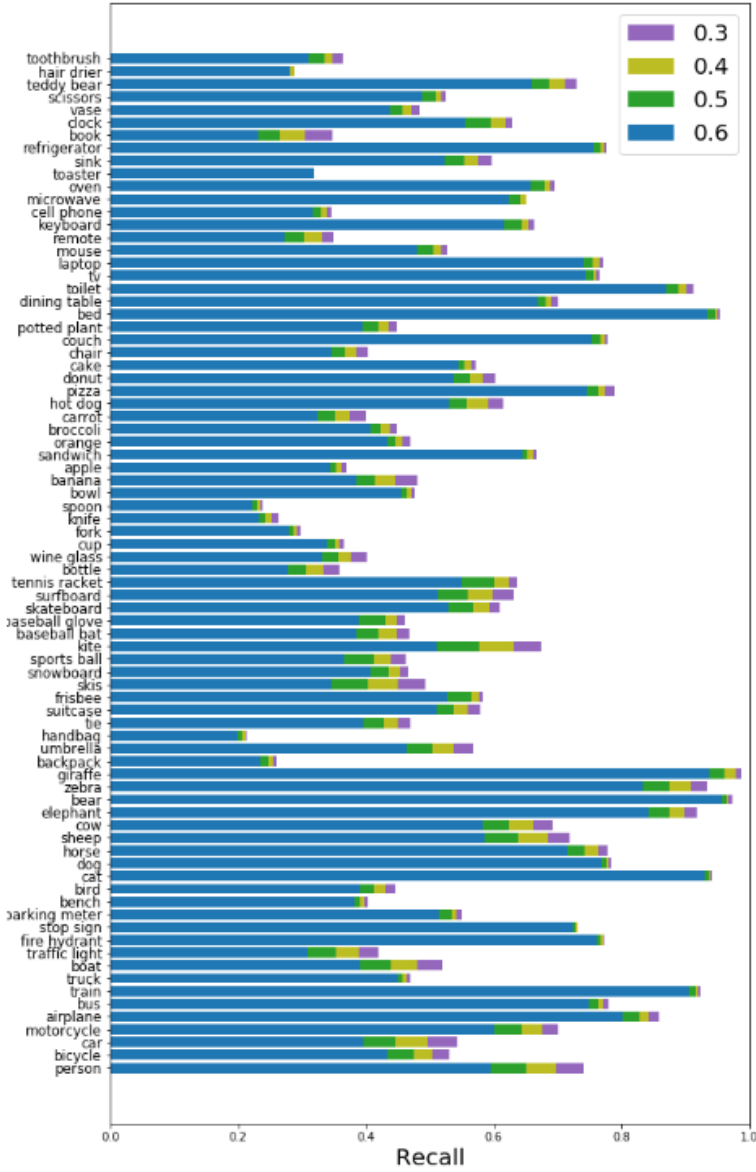
Yukarıda da bahsedildiği gibi bir nesnenin doğru ya da yanlış tespit edildiğine karar vermek için önceden belirlenen bir örtüşme oranı eşik değeri olarak kullanılır. Bu eşik değeri literatürde genel olarak %50 olarak kabul edilmiştir, ancak farklı oranlar da kullanılabilir. Kullanılan eşik değeri başta TP ve FP oranları olmak üzere tüm performans ölçümlerini doğrudan etkilemektedir.

Performans ölçümü için kullanılan bir diğer metrik ise “recall” olarak isimlendirilir. “Recall” nesne tespitinde gerçek nesnelerin hangi oranla doğru tahmin edildiğinin bir ölçütüdür ve Denklem 5.1’de verilen eşitlik ile hesaplanır [42].

$$Recall = \frac{TP}{(TP + FN)} \quad (5.1)$$

Şekil 5.2’de MS COCO 2014 eğitim veri seti için her bir sınıfın “recall” değeri farklı örtüşme oranı eşik değerleri için raporlanmıştır. Grafikte farklı renkler ile o rengin temsil ettiği örtüşme oranı eşik değerine ait “recall” değeri raporlanmıştır. Görüldüğü üzere eşik değeri büyüdükçe “recall” değeri küçülmüştür. Fil, ayı, zebra gibi hayvanları temsil eden sınıflar için yüksek “recall” değerleri ölçülmüşken sırt çantası ve el çantası gibi sınıflarda daha düşük değerler ölçülmüştür. Burada “recall” değerinin farklı sınıflar için çok farklı olabileceği görülmektedir. Yani MobileNetv1 SSD nesne

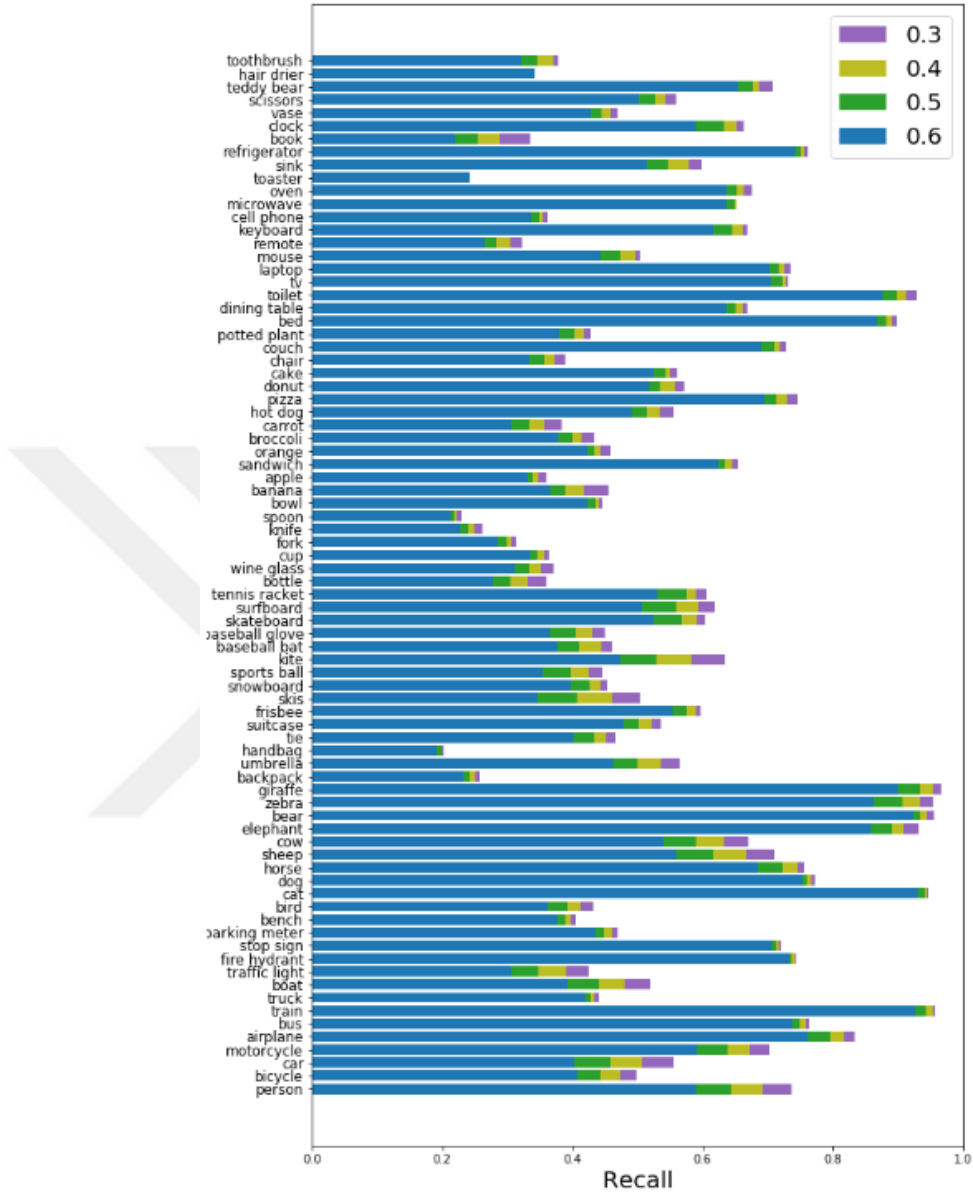
tespit edici eğitildiği her sınıfı farklı oranda öğrenebilmiştir. Bu durum bazı nesne sınıflarının daha zor öğrenilebilmesinden kaynaklanmaktadır.



Şekil 5.2 : MS COCO 2014 eğitim setinde MobileNetv1 SSD’nin sınıflara göre sağladığı “recall” değerleri.

Şekil 5.3’de MS COCO 2014 doğrulama veri seti için her bir sınıfın “recall” değeri farklı örtüşme oranı eşik değerleri için raporlanmıştır. Bu sonuçların eğitim veri setindeki sonuçlara çok yakın olduğu görülmektedir. “recall” ölçütünün nesnelerin tespit edilememe oranlarının bir ölçütü olduğu düşünülürse bu durum nesne tespit edicinin öğrendiği nesneleri eğitim veri setinde ve bambaşka bir set olan doğrulama

veri setinde yaklaşık aynı oranda tespit edemediğini, kaçırma oranının bilinmeyen veri setlerinde artmadığını göstermektedir.

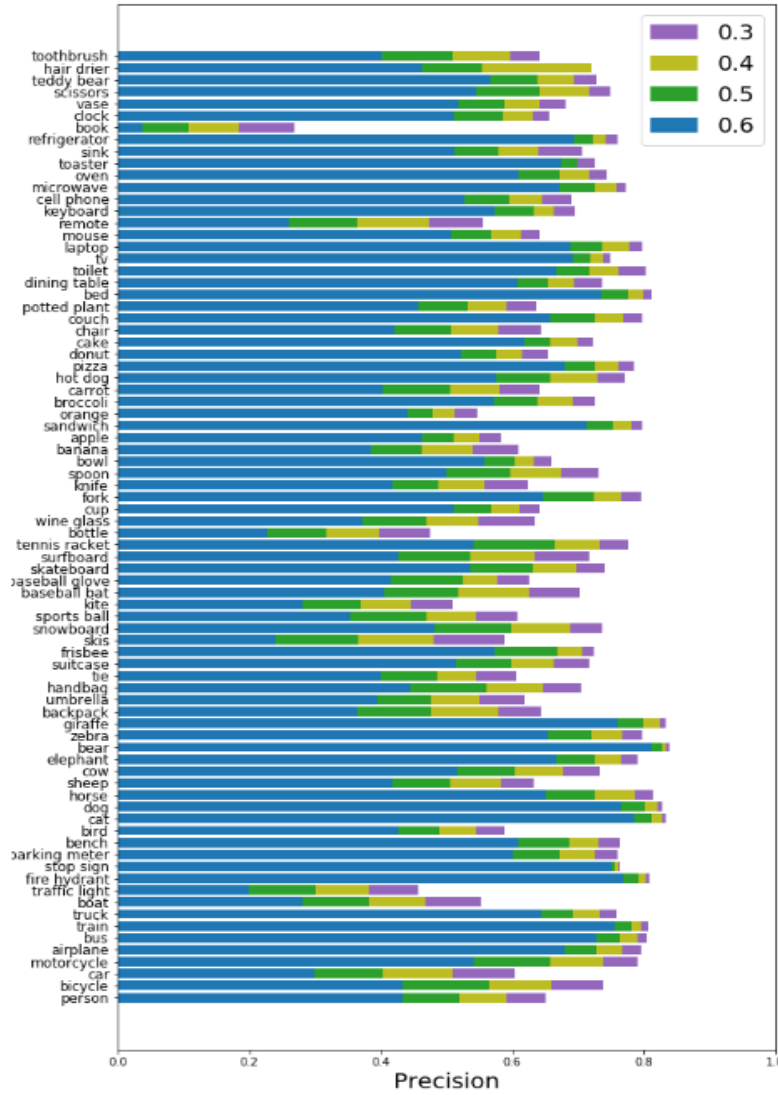


Şekil 5.3 : MS COCO 2014 doğrulama setinde MobileNetv1 SSD’nin sınıflara göre sağladığı “recall” değerleri.

Bir diğer performans metriği olan “precision” ise nesne olarak tahmin edilen BB’lerin hangi oranla gerçekten nesne içerdiğinin ölçütüdür ve Denklem 5.2 ile hesaplanır [42].

$$Precision = \frac{TP}{(TP + FP)} \quad (5.2)$$

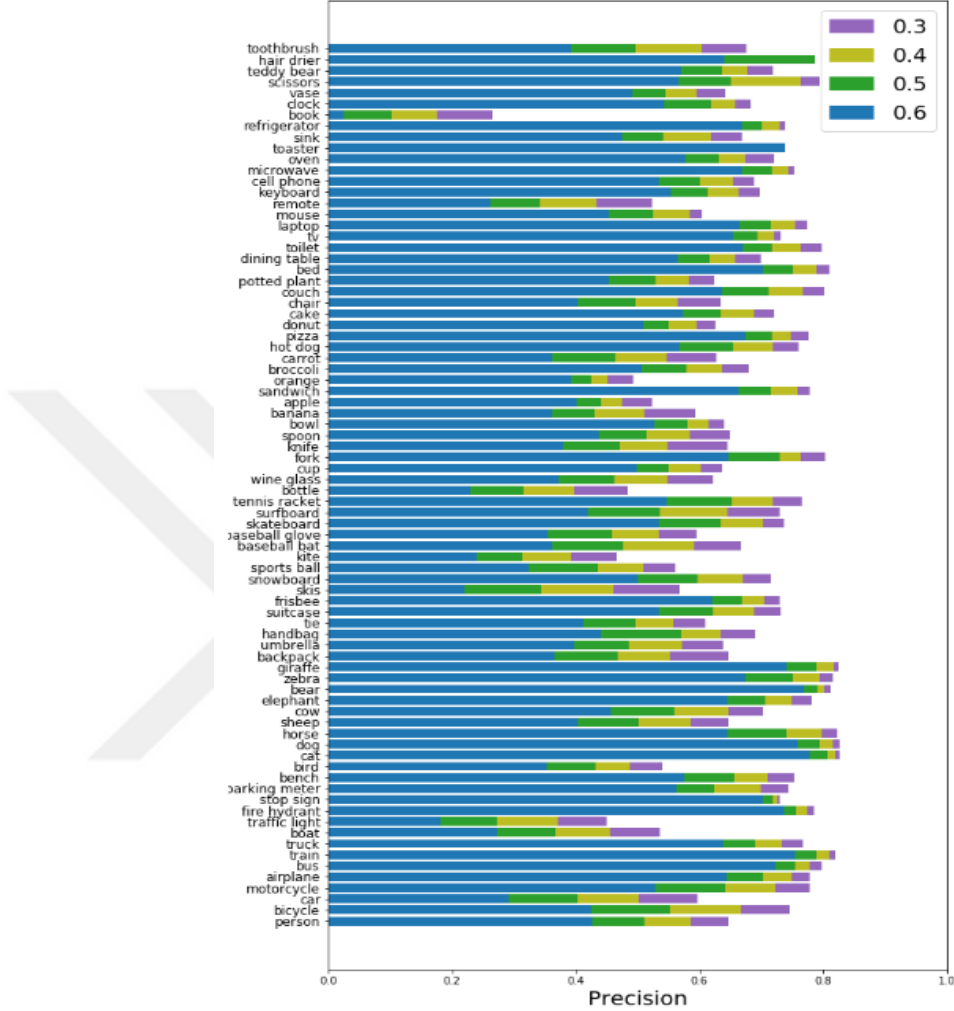
Şekil 5.4’de MS COCO 2014 eğitim veri seti sınıfın “precision” değeri farklı örtüşme oranı eşik değerleri için raporlanmıştır. Grafikte farklı renkler ile o rengin temsil ettiği örtüşme oranı eşik değerine ait “precision” değeri raporlanmıştır. Görüldüğü üzere eşik değeri büyüdükçe “precision” değeri küçülmüştür. En düşük “precision” değerleri itap ve trafik lambası nesne sınıflarında elde edilmiştir. “precision” metriğinin nesneleri başka nesneler ile karıştırma oranının bir ölçütü olduğu düşünülürse, kitap ve trafik lambası başka nesnelerle en çok karıştırılan nesne sınıflarıdır sonucu çıkarılabilir.



Şekil 5.4 : MS COCO 2014 eğitim setinde MobileNetv1 SSD’nin sınıflara göre sağladığı “precision” değerleri.

Şekil 5.5’te MS COCO 2014 doğrulama veri seti için her bir sınıfın “precision” değeri farklı örtüşme oranı eşik değerleri için raporlanmıştır. Bu sonuçların eğitim veri

setindeki sonuçlara çok yakın olduğu görülmektedir. Bir başka deyişle, MobileNetv1 SSD'nin nesneleri karıştırma oranı farklı veri setleri için aynı kalabilmektedir, artmamaktadır.



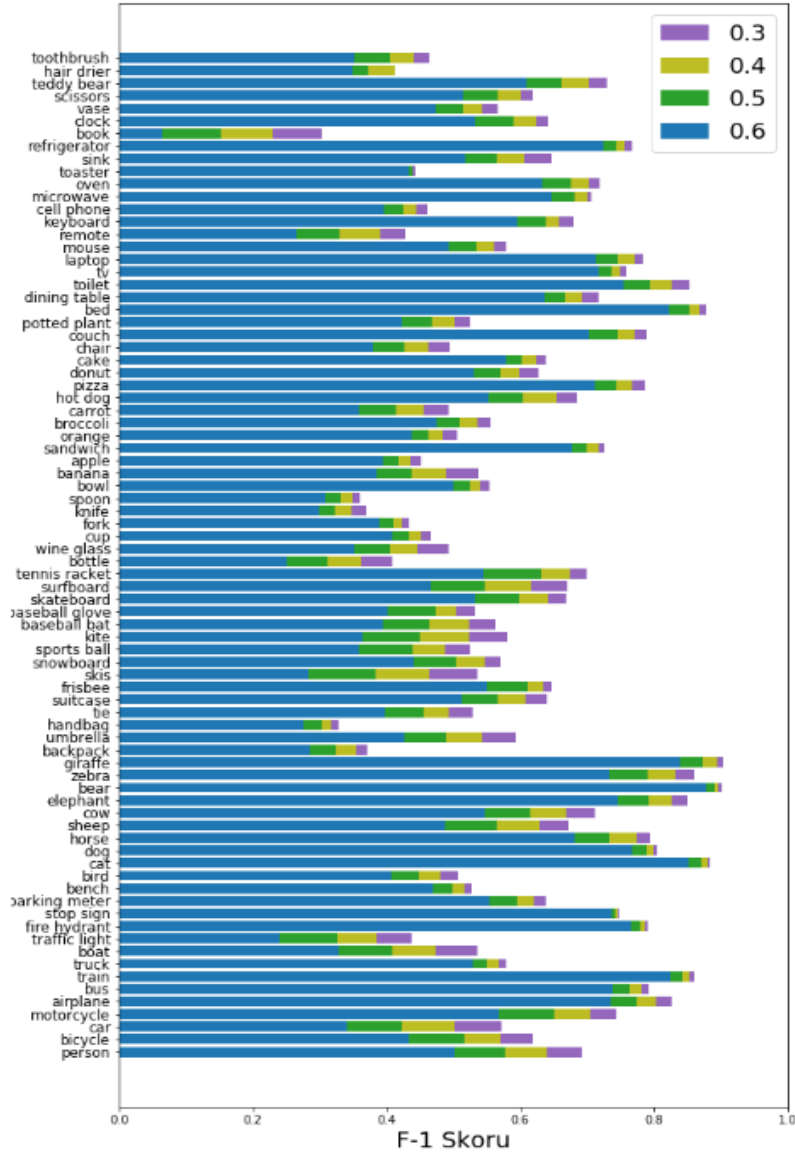
Şekil 5.5 : MS COCO 2014 doğrulama setinde MobileNetv1 SSD'nin sınıflara göre sağladığı “precision” değerleri.

Bu iki metrik kullanılarak elde edilen bir diğer ölçüm metriği F1 skoru olarak adlandırılır ve “precision” ile “recall” parametrelerini harmonik ortalamasına karşı düşer. F1 skoru Denklem 5.3 kullanılarak hesaplanır [43].

$$F1 \text{ Skoru} = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \quad (5.3)$$

Şekil 5.6’da MS COCO 2014 eğitim veri seti için her bir sınıfın F1 skoru farklı örtüşme oranı eşik değerleri için raporlanmıştır. Grafikte farklı renkler ile o rengin temsil ettiği

örtüşme oranı eşik değerine ait F1 skoru değeri raporlanmıştır. Görüldüğü üzere eşik değeri büyüdükçe F1 skoru değeri küçülmüştür. Tespit edilen nesneye ait BB'nin referans olarak verilen BB ile örtüşme oranı arttıkça F1 değerinin küçülmesi beklenen bir sonuçtur ve nesne tespit edicinin esasen nesneyi bulduğu ancak lokalizasyonunu ve boyutlarını belirleyen BB ile yüzde yüz bir örtüşme sağlayamadığını göstermektedir.



Şekil 5.6 : MS COCO 2014 eğitim setinde MobileNetv1 SSD'nin sınıflara göre sağladığı F1 skoru.

Şekil 5.7'de MS COCO 2014 doğrulama veri seti için her bir sınıfın F1 skoru farklı örtüşme oranı eşik değerleri için raporlanmıştır. Bu sonuçların eğitim veri setindeki sonuçlara çok yakın olduğu görülmektedir. F1 skoru nesne tespit edicinin nesneleri

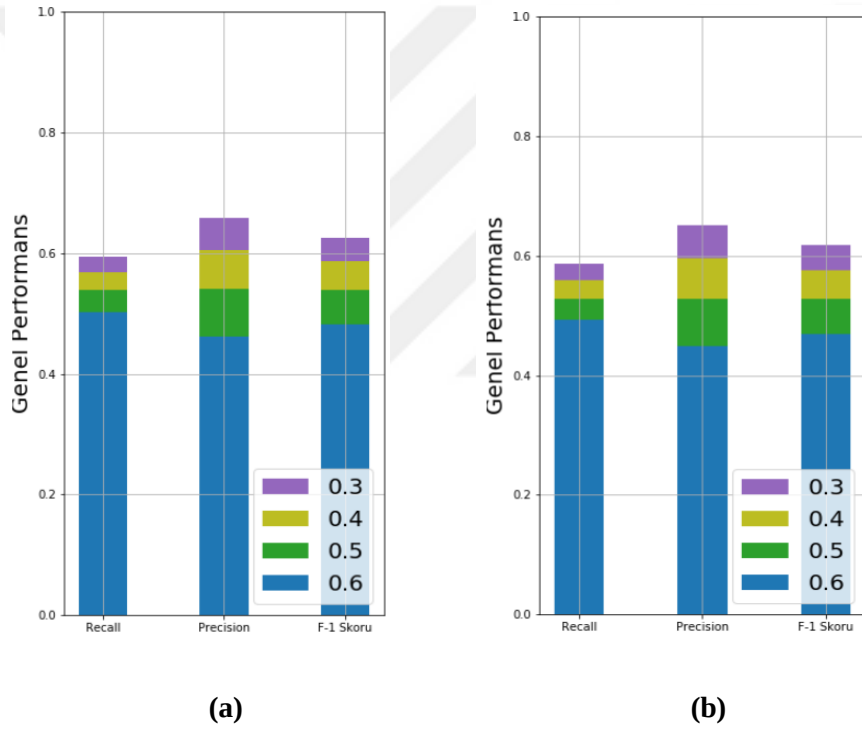
kariřtırma ve bulamama oranlarını birlikte görmemizi saęlayan bir ölçüttür. Bu nedenle F1 skoru ile ölçülen performans, performans “recall” ve “precision” ölçütlerinde olduęu gibi, farklı veri setlerinde aynı seviyede kalabilmektedir, azalmamaktadır řeklinde yorumlanabilir.



řekil 5.7 : MS COCO 2014 doęrulama setinde MobileNetv1 SSD’nin sınıflara göre saęladıęı F1 skoru.

Yukarıdaki paragraflarda nesne sezme performansı her bir nesne sınıfı için ayrı ayrı raporlanmaktadır. Tüm sınıflar için elde edilen genel başarıımı görebilmek amacıyla MS COCO 2014 eęitim seti üzerinde elde edilen ortalama “recall”, “precision” ve F1 skoru metrikleri řekil 5.8(a)’da raporlanmaktadır. Benzer raporlama řekil 5.8(b)’de MS COCO 2014 doęrulama seti için verilmektedir. Sonuçlardan görüldüęü üzere, beklendięi gibi, nesne tespit edildi kabulü yapılan örtüşme oranı (IoU) eřik deęeri

düşükçe performans yükselmektedir. Genel olarak performansın çok yüksek değerlerde çıkmaması, MS COCO veri setinin zor bir set olmasından ve yukarıdaki sınıf bazında raporlanan sonuçlardan görüldüğü üzere, bazı nesne sınıflarında çok düşük öğrenme sağlanabilmesinden kaynaklanmaktadır. Bir diğer çıkarım, eğitim ve doğrulama setlerinde elde edilen performans oldukça yakındır ki bu durum derin öğrenme ağının öğrendiği nesneleri içeren eğitim seti üzerindeki başarımını hiç görmediği görüntüleri içeren doğrulama seti için de koruduğunu doğrulamaktadır. Her üç metrik için raporlanan sonuçların birbirine yakın olması ise, doğru tespit oranı yüksek tutulurken hatalı tespit oranının ve kaçırma oranının çok yükselmediğini göstermektedir.



Şekil 5.8 : MS COCO 2014 veri seti için genelleştirilmiş sonuçlar.

5.2 VOT Videolarında Nesne İzleme Performansı

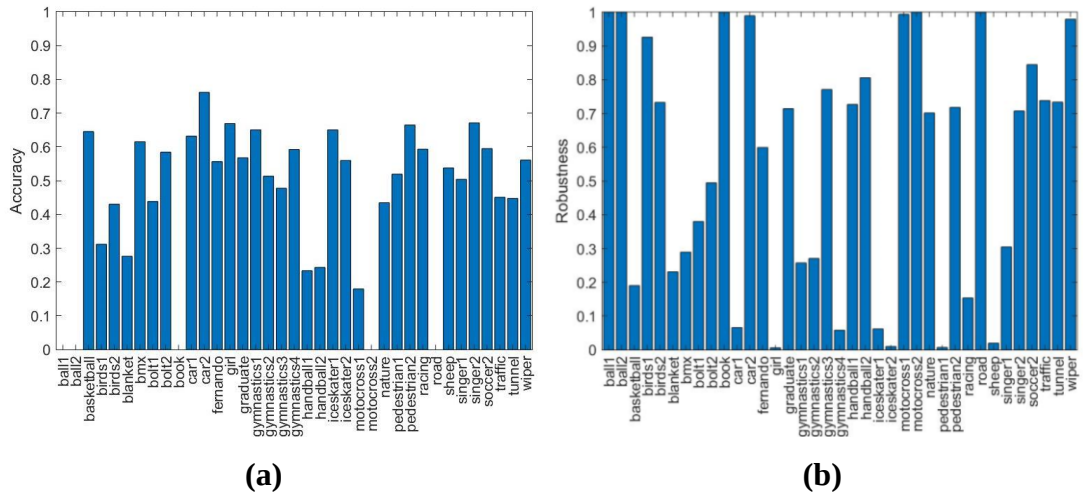
Tez çalışmasında kullanılan MobileNetv1 SSD isimli derinlikli ağ ayırık resimler üzerinde çalışmakta olup herhangi bir hareket bilgisi kullanmamaktadır. Bu nedenle bir önceki bölümde MS COCO resimleri üzerinde performansı irdilemiştir. Tez çalışması kapsamında amaç video nesnelerinin izlenmesidir ve bu izlemede ivmeölçer yardımıyla hareket bilgisi modele katılmıştır. Bu nedenle bu alt bölümde MobileNetv1 SSD'nin VOT 2016 [44] veri tabanı üzerindeki nesne izleme performansı

incelenmiştir. VOT veri setleri 2013’ten bu yana düzenlenen nesne izleme yarışmalarında kullanılmak üzere çeşitli özelliklerdeki videolardan oluşturulmuştur.

Videoda nesne izleme performansının raporlanmasında sıklıkla kullanılan bir metrik doğruluk (“accuracy”) olarak adlandırılmaktadır [41]. Diğer ölçütlerden farklı olarak doğruluk nesne tespit lokalizasyonunun doğruluğunu ölçer ve bu nedenle, Denklem 5.4’te görüldüğü gibi, sadece nesnenin başarıyla yakalandığı video karelerindeki örtüşme oranlarının ortalaması şeklinde hesaplanır. Denklem 5.4’te N_S nesnenin başarıyla izlendiği video karelerinin sayısını ve IoU nesnenin başarıyla izlendiği video karelerindeki örtüşme oranını göstermektedir.

$$Accuracy = \frac{1}{N_S} \sum_{i=1}^{N_S} IoU_i \quad (5.4)$$

Şekil 5.9(a)’da VOT 2016 veri setindeki bazı videolar için doğruluk değerleri görülmektedir. Bu değerlerin hesaplanmasında kullanılan örtüşme oranı eşiği 0.1’dir. Ayrıca bazı videolarda nesnenin hiç bulunamadığı da görülmektedir.



Şekil 5.9 : VOT 2016 veri setindeki bazı videoların (a) doğruluk değerleri, (b) dayanıklılık değerleri.

Nesne izleme performansının ölçümünde kullanılan bir diğer metrik dayanıklılık (“robustness”) olarak isimlendirilmektedir. Dayanıklılık tüm video karelerinin ne kadarlık bir oranında nesnenin hiç yakalanamadığının (“miss detection”) bir ölçüsüdür [41] ve Denklem 4.5 kullanılarak hesaplanır. Denklemde görülen N_F nesnenin hiç yakalanamadığı video karelerinin sayısını, N_T videodaki toplam kare sayısını göstermektedir.

$$Robustness = \frac{N_F}{N_T} \quad (5.5)$$

Şekil 5.9(b)'de VOT 2016 veri setindeki bazı videolar için dayanıklılık değerleri görülmektedir. Bu değerlerin hesaplanmasında kullanılan örtüşme oranı eşiği 0.1'dir. Ayrıca bazı videolarda nesnenin hiç bulunamadığı ve bazı videolarda da neredeyse tüm karelerde nesnenin izlendiği görülmektedir.

Çizelge 5.1 : VOT 2016 veritabanı nesne izleme sonuçları.

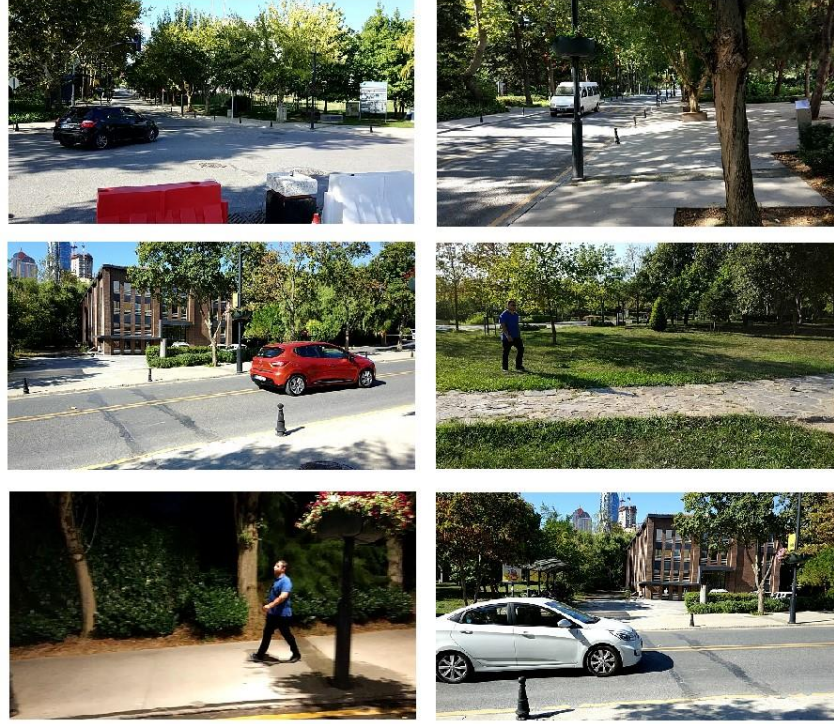
Model	Bilinen Nesne		Bilinmeyen Nesne	
IoU Eşiği	0.1	0.01	0.1	0.01
Doğruluk	0.52	0.46	0.46	0.37
Dayanıklılık	0.55	0.51	0.40	0.21

MobileNetv1 SSD'nin VOT 2016 videoları üzerinde sağladığı ortalama performansı değerlendirebilmek açısından Çizelge 5.1'de iki farklı IoU eşik değerlerinde elde edilen ortalama doğruluk ve ortalama dayanıklılık değerleri raporlanmaktadır. Çizelgede bilinen nesne ile yapılan raporlama izlenen nesnenin araba, insan vb. sınıf etiketinin bilinmesi durumunda elde edilen performansı, bilinmeyen nesne ise bu etiketlere bakılmaksızın elde edilen performansı göstermektedir. Çizelgeden görüldüğü üzere, bilinmeyen nesne için dayanıklılık bayağı yükselmektedir ki bu durum MobileNetv1 SSD derinlikli ağının tespit ettiği nesnelerin sınıf etiketine bakılmaksızın tespit yapıldığında, tespit edilemeyen nesne oranının önemli oranda azaldığını göstermektedir. Çıkarılan bir diğer sonuç ise, Şekil 5.9'da video bazında yapılan raporlamada bazı videolarda %80 doğruluk ve %99 dayanıklılık değerlerine erişilmesine karşın, ortalama performan bunun çok altındadır. Bu durum büyük oranda, derinlikli nesne tespit edicinin hareket bilgisi kullanılmaksızın video karesi bazında işlem yapmasından kaynaklanmaktadır ki tez çalışmasında, videodaki zamansal (temporal) bilginin ivmeölçerden alınan geri besleme ile nesne tespit modeline eklenmesi ve böylelikle performansın artırılması hedeflenmektedir.

5.3 Cep Telefonu İle Çekilen Videolarda Performans

Bu alt bölümde tez çalışması kapsamında önerilen, MobileNetv1 SSD ve ivmeölçer sonuçlarını tümleştiren, yöntemin performansı raporlanmaktadır. Bu amaçla öncelikle ivmeölçer ve video kayıtlarının birlikte yapıldığı bir test veri tabanı oluşturulmuştur.

Şekil 5.10’da Xiaomi Mi5s model bir mobil telefon ile çekilerek oluşturulan veri tabanından örnek video kareleri görülmektedir. Videolarda örtüşme, ıııklılık deęiřiklięi, bulanıklık, bakıř aısı deęiřiklięi, lek deęiřiklięi gibi dnřmlere uęramıř araba, insan, motosiklet gibi eřitli nesneler bulunmaktadır. Mobil kamera ile kaydedilen videoların ierdięi zorluklar ve zellikleri izelge 5.2’de zetlenmektedir.



Şekil 5.10 : Oluřturulan veri setinden alınmıř rnek video kareleri.

izelge 5.2 : Oluřturulan veri tabanındaki videolar ve zellikleri.

Video Klip No	Video Karesi Sayısı	Zorluk Tipi
1	71	Poz, lek, ıııklılık, derinlik deęiřimi
2	110	Poz, lek deęiřimi
3	47	lek, derinlik deęiřimi, rtřme
4	66	Yok
5	45	Yok
6	176	Poz, lek, derinlik deęiřimi
7	63	Dřk ıřık, bulanıklık
8	51	Dřk ıřık, bulanıklık
9	56	Dřk ıřık, bulanıklık
10	81	Poz deęiřimi
11	38	Yok
12	50	Yok

Şekil 5.11’de oluşturulan veri tabanındaki videolarda sadece MobileNetv1 SSD kullanılarak tespit edilen nesnelerin gösterildiği örnek video kareleri görülmektedir. Görüldüğü gibi derinlikli nesne sezici her bir video karesinde öğrenmiş olduğu nesneleri tespit etmesi durumunda nesneye ilişki BB’leri ve herbir BB için tespit edilen nesne sınıfı için hesaplanan nesnelilik skorunu raporlamaktadır. Ancak bazı karelerde örtüşme olması, ışıklılık değişimi, bazı karelerde nesnenin çok küçük olması, çok fazla dönmüş olması veya farklı deformasyonlara uğramış olması nedeniyle nesneler tespit edilememektedir. Önerilen yöntem ile ivmeölçerden geri besleme alınarak nesne tespit performansının yükseltilmesi beklenmektedir ve buna ilişkin sonuçlar aşağıdaki paragraflarda raporlanmaktadır.



Şekil 5.11 : MobileNet SSD ile tespit edilen nesne BB’leri ve nesnelilik skorları.

Veri tabanında toplam 12 video klibi kaydedilmiş olup bunlara ilişkin referans sonuç verisi (ground truth) bulunmamaktadır. Bu nedenle ilk test senaryosu için video kliplerden 31 frame içeren bir video klip için referans işaretleme elle yapılmıştır. Bu video klibe ait 8 video karesi Şekil 5.12’de görülmektedir. Takip edilmeye çalışılan nesne beyaz renkli araba olup video boyunca sola doğru hareket etmekte ve videonun sonunda sahneden çıkmaktadır. Mobil kamera ise önce sola ardından ise sağa doğru

yatay ekseninde hareket etmektedir, bir başka deyişle kamera “pan” hareketi yapmaktadır.



Şekil 5.12 : 31 karelik örnek videodan bazı video kareleri.

Çizelge 5.3'te bu video için nesne izleme performansı farklı metrikler cinsinden raporlanmaktadır. Ortalama örtüşme oranı (Ort IoU), tüm video karelerinde hesaplanan referans ve tahmin edilen BB'ler arasındaki örtüşme oranlarının ortalamasıdır. Buna göre sadece MobileNetv1 SSD'nin kullanıldığı durumdaki

ortalama örtüşme oranı 0.31 olarak hesaplanırken ivmeölçerden alınan geri besleme ile bulunamayan karelerde yakalanan nesne BB leri sayesinde 0.65’lik bir ortalama IoU değerine erişilebilmiştir. Çizelgede raporlanan merkez piksel kayması (Merkez Pik. Kayma) ölçütüne bakıldığında, ivmeölçerin tek başına kullanılması ile tespit edilen nesne BB’leri için 25 piksellik bir ortalama kayma hesaplanmıştır ki bu değer MobileNetv1 SSD nin tek başına kullanılması ve ivmeölçer ile birlikte kullanılması ile elde edilen ortalama 21-22 piksellik kaymadan daha yüksektir. İvmeölçer ile kaydedilen en yüksek kayma 117 pikseldir ancak MobileNetv1 SSD ile birlikte kullanılınca bu hata 59 piksele düşmektedir. Çizelgeden çıkarılacak bir diğer sonuç, MobileNetv1 SSD ile ivmeölçer kestirimlerinin birlikte kullanılması kaçırılan nesnelerin tespitine olanak sağlamıştır ki bunun bir sonucu olarak dayanıklılık ölçütü 0.19’a düşürülebilmektedir. Sonuç olarak, önerilen karar tümleştirme MobileNetv1 SSD derinlikli nesne tespit edicisinin performansını arttırmıştır. Şekil 5.13’de MobileNetv1 SSD ile sezilemeyen 5. ve 26. video karelerindeki araba, ivmeölçer tarafından kestirilen nesne BB’leri (kırmızı) yardımı ile tespit edilebilmiştir ve sonuçlar (mavi) referans BB’ler ile (sarı) uyumludur.

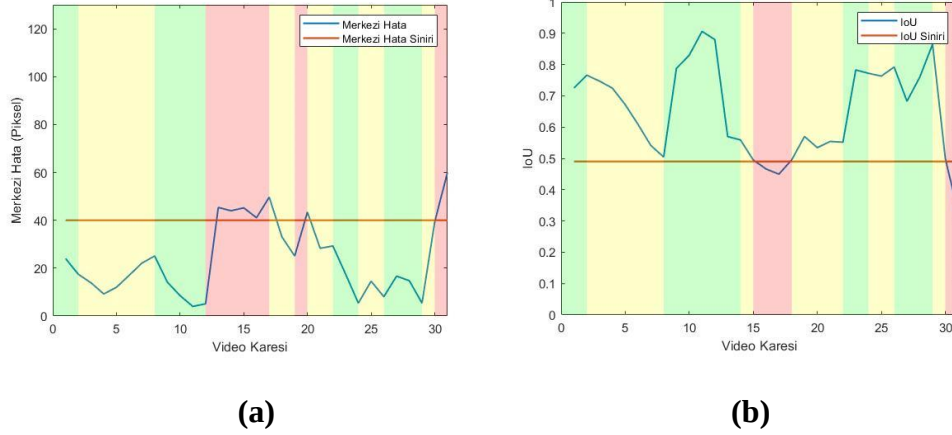
Çizelge 5.3 : 31 karelik örnek videoya ait detaylı sonuçlar.

	MobileNet SSD	İvmeölçer	MobileNet SSD + İvmeölçer
Ort IoU	0.31	0.63	0.65
MinIoU	0.0	0.32	0.33
Maks IoU	0.90	0.87	0.91
Ort Merkez Pik. Kayma	21	25	22
Min Merkez Pik. Kayma	4	4	4
Maks Merkez Pik. Kayma	58	117	59
Dayanıklılık	0.59	0.23	0.19



Şekil 5.13 : MobileNet SSD + İvmeölçer ile 5. ve 26. video karelerinde yapılan nesne tespiti.

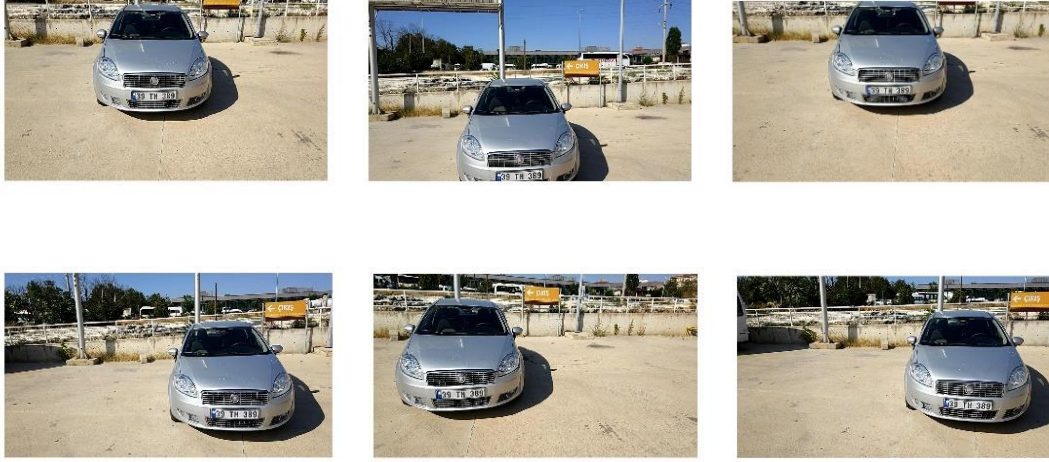
Şekil 5.14’de yukarıda performansı raporlanan video klip için MobileNetv1 SSD ile tespit edilebilme oranı, ivmeölçer verisinin kullanılması sayesinde nesne tespit oranında elde edilen artış, ve nesnenin tespit edilemediği video karesi oranı raporlanmaktadır. Şekil 5.14(a)’da nesne tespit edildi kriteri BB merkez piksel hatasının 40 pikselin altında kalması, Şekil 5.14(b)’de ise referans BB ile en az %50 örtüşme sağlanması olarak alınmıştır. Bu sonuçlara göre Şekil 5.14(a)’da video karelerinin %77.4’ünde doğru nesne tespiti yapılabildiği ve bu oranda ivmeölçerden geri besleme alınmasının katkısı %41.9 gibi önemli bir artıştır. Benzer şekilde Şekil 5.14(b)’de toplam video karelerinin %80.6 sında nesne tespiti başarılı olarak yapılmıştır ve bu oranda ivmeölçerden alınan geri beslemenin katkısı %38.7 gibi önemli bir katkıdır.



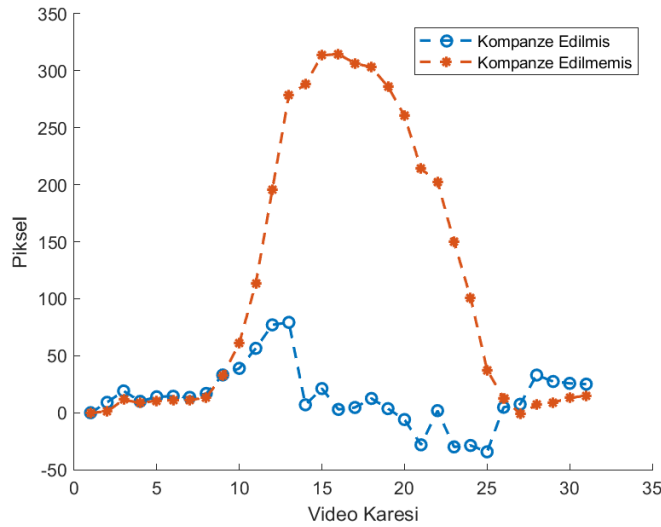
Şekil 5.14 : MobileNet SSD (yeşil), ivmeölçer geri beslemesinin katkısı (sarı), kaçırılanlar (kırmızı) (a) Merkez piksel kayması eşiği 40 piksel. (b) IoU eşiği 0.5.

Performans raporlamada ikinci test senaryosunda, nesnelerin hareketsiz olduğu ancak mobil kameranın hareketli olduğu durumlarda nesne tespit performansının incelenmesi hedeflenmiştir. Şekil 5.15’te çekilen videolardan örnek kareler görülmektedir. Tespit edilecek nesne duran arabadır, mobil kamera hareketi “pan” ve “tilt” tipindedir. Önerilen nesne tespit yöntemi ile öncelikle kamera hareketi tespit edilmiştir. Ardından bu hareket kompanze edilmiştir ve nesne tespiti yapılmıştır. Şekil 5.16’da bu videolardan “tilt” hareketi olan 32 video karesinde nesne tespiti sonucunda elde edilen nesne BB lerinin hareketinin piksel cinsinden değerleri kamera hareketi kompanze edilmeden önce (kırmızı) ve sonra (mavi) raporlanmaktadır. Bu sonuçlarda nesnenin ilk konumu sıfır kabul edilmiş ve buna göre nesnenin kameranın dikey eksenindeki konumu her bir video karesi için gösterilmiştir. Görüldüğü gibi hareket kompanzasyonundan sonra nesne hareketi yaklaşık sıfır piksel olarak

öngörülelebilmektedir ki bu değerler gerçekleşen tespit edicinin yüksek doğrulukla çalışabildiğini göstermektedir.



Şekil 5.15 : Sabit nesnen hareketli kamera durmuna ait video kareleri.



Şekil 5.16 : Örnek bir videoya ait piksel cinsinden kamera hareketi kompanze edilmiş ve edilmemiş nesne hareketi.

Performans raporlamada kullanılan son test senaryosunda, nesnelerin ve mobil kameranın hareketli olduğu videolar üzerinde nesne tespit performansının raporlanması amaçlanmaktadır. Bu amaçla özellikleri Çizelge 5.3'te açıklanan 12 video kullanılmıştır. Videolar kameradan 1280x720 piksel çözünürlükte kaydedilmekle birlikte, hız ve efektif bellek kullanımı açısından MobileNetv1 SSD girişindeki çözünürlük 300x300 piksel olarak kullanılmıştır. Performans Çizelge 5.4'te raporlanmıştır. Buradaki IoU, MobileNetv1 SSD ve ivmeölçere dayanarak

hesaplanan iki farklı BB'nin birbiri ile olan örtüşme oranıdır. IoU ve MobileNetv1 SSD'ye ait nesnelilik skoru yalnızca MobileNetv1 SSD'nin başarıyla çalıştığı video kareleri için raporlanmıştır. Bu veri tabanı için ortalama örtüşme oranı 0.72 ve ortalama nesnelilik skoru 0.81 olarak hesaplanmıştır. Nesnenin kaybedildiği video karelerinin oranının belirten dayanıklılık ise bu veri tabanı için nesnelerin gerçek konumları işaretli olmadığından görsel olarak belirlenmiş ve yaklaşık değerleri raporlanmıştır.

Çizelge 5.4'teki veri tabanının genelini yansıtan ortalama sonuçlar incelenirse tez çalışmasında önerilen yöntemin nesneyi dayanıklılık değerini azalttığı görülebilir. Bu durumdaki en büyük pay önerilen yöntemin MobileNetv1 SSD nesneyi tamamen kaybettiğinde dahi nesneyi izleyebilme yetisine sahip olmasındadır. Yine dikkat edilirse sadece MobileNetv1 SSD'ye dayanarak yapılan nesne izlemenin dayanıklılığı 0.18 ve sadece ivmeölçere dayanarak yapılan nesne izlemenin dayanıklılığı 0.14 olarak belirlenmiştir. Öte yandan bu iki yöntemin Şekil 4.1'deki gibi birleştirilmesiyle elde edilen tümleşik yöntemin dayanıklılığı 0.08'dir. Bu değerler iki yöntemin etkili bir biçimde birleştirildiğinin bir göstergesidir.

Çizelge 5.4 : Oluşturulan veri tabanı üzerinde elde edilen sonuçlar.

Video No	Ortalama IoU	Ortalama Nesnelilik Skoru	MobileNet SSD dayanıklılık	İvmeölçer dayanıklılık	MobileNet SSD + ivmeölçer dayanıklılık
1	0.79	0.78	0.30	0.23	0.11
2	0.83	0.97	0.24	0.19	0.15
3	0.57	0.63	0.55	0.49	0.47
4	0.79	0.91	0.02	0.05	0.02
5	0.81	0.96	0.04	0.02	0.02
6	0.59	0.80	0.11	0.14	0.07
7	0.72	0.74	0.16	0.10	0.00
8	0.73	0.84	0.10	0.10	0.00
9	0.55	0.73	0.39	0.29	0.16
10	0.86	0.72	0.22	0.07	0.01
11	0.51	0.71	0.16	0.11	0.05
12	0.97	0.94	0.00	0.00	0.00
Ortalama	0.72	0.81	0.18	0.14	0.08

Şekil 5.17'de sırasıyla veri tabanındaki 4. videonun 48., 51., 54. kareleri ve 6. videonun 110., 111., 112. kareleri görülmektedir. Bu video kareleri tez çalışmasında

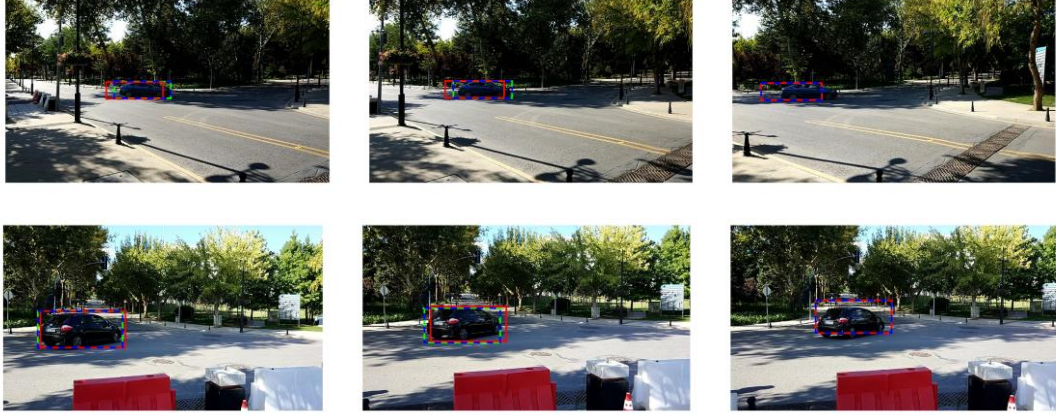
önerilen yöntemin örtüşme (“occlusion”) durumunda nasıl davrandığını göstermektedir. Bu karelerdeki örtüşme izlenmek istenen nesnenin bir kısmının görüntü düzleminin dışına çıkması sonucunda oluşmuştur. Görüldüğü üzere izlenmek istenen nesnenin tamamı görüntü düzleminin içindeyken nesne MobileNet SSD tarafından başarıyla bulunmuştur. Öte yandan 4. videonun 54. karesi ve 6. videonun 112. karesinde görüldüğü gibi nesnenin büyük bir kısmı görüntü düzleminin dışına çıktığında nesne MobileNet SSD tarafından kaybedilmiş ancak ivmeölçer verisi sayesinde bulunmuştur. Buna göre önerilen yöntemin örtüşme durumunda dahi dayanıklılık üzerinde olumlu etkiye sahip olduğu söylenilebilir. Ayrıca, aksi belirtilmedikçe tüm şekillerdeki video karelerinde MobileNet SSD tarafından oluşturulan BB yeşil ile, ivmeölçer verisine dayanarak oluşturulan BB kırmızı ile ve bu ikisi arasından son karar olarak kullanılan BB kesikli mavi ile gösterilmiştir.



Şekil 5.17 : Veritabanındaki videolardan örtüşme durumuna örnek oluşturan video kareleri.

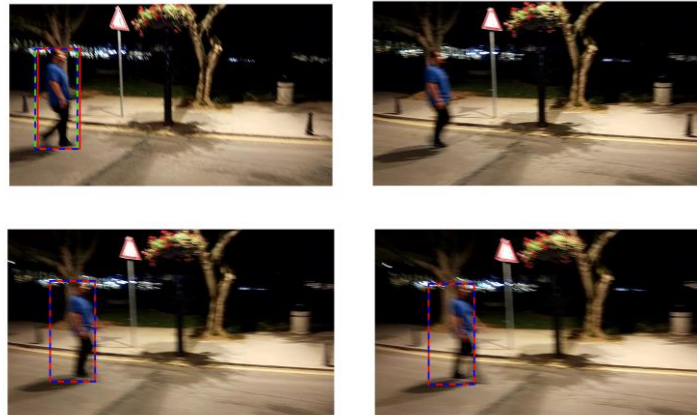
Şekil 5.18’in ilk satırında veri tabanındaki 1. videonun sırasıyla 30., 32., 35. kareleri ve ikinci satırında 2. videonun sırasıyla 67., 69., 71. kareleri görülmektedir. Bu video kareleri tez çalışmasında önerilen tümleşik yöntemin poz değişimi durumunda nasıl davrandığını göstermektedir. Görüldüğü üzere nesne poz değişiminin nispeten az olduğu 1. videonun 32. karesinde ve 2. videonun 69. karesinde MobileNet SSD tarafından başarıyla bulunmuştur. Öte yandan, 1. videonun 35. ve 2. videonun 71. karesinde poz değişiminin etkisi artmıştır ve nesne MobileNet SSD tarafından kaybedilmiştir. Bu video karelerinde nesne ivmeölçer verisine dayanarak bulunmuştur. Bu durum MobileNet SSD’nin tek başına poz değişimine karşı yeterince dayanıklı olmamasına rağmen tez çalışmasında önerilen tümleşik yöntemin ivmeölçer

geri beslemesi sayesinde poz deęişimine karşı dayanıklı olduęu şeklinde yorumlanabilir.



Şekil 5.18 : Veritabanındaki videolardan poz deęişimi durumuna örnek oluşturan video kareleri.

Şekil 5.19’da sırasıyla veri tabanındaki 9. videonun 15., 17., 18. ve 19. kareleri görölmektedir. Bu video kareleri tez çalışmasında önerilen yöntemin bulanıklık (“blur”) durumunda nasıl davrandığını göstermektedir. Görüldüğü üzere bulanıklığın kabul edilebilir seviyede olduęu 15. video karesinde nesne MobileNet SSD tarafından bulunmuştur. Öte yandan bulanıklığın arttığı dięer video karelerinde nesne MobileNet SSD tarafından kaybedilmiştir. Bunlardan 18. ve 19. video karelerinde nesne ivmeölçer verisine dayanarak bulunmuş, 17. video karesinde tamamen kaybedilmiştir. Buna göre tez çalışmasında önerilen yöntemin bulanıklığa karşı kısmen dayanıklı olduęu söylenebilir.



Şekil 5.19 : Veritabanındaki videolardan bulanıklık durumuna örnek oluşturan video kareleri.

Şekil 5.20’de sırasıyla veri tabanındaki 10. videonun 40., 43., 49., 51., 52. ve 53. kareleri görülmektedir. Bu video karelerinde poz değişimi, ölçek değişimi ve örtüşme durumlarının etkileri görülmektedir. Aynı anda üç farklı özel durumun etkilerinin görülmesi nedeniyle bu video karelerinden yalnızca 43. ve 51. video karelerinde nesne MobileNet SSD tarafından bulunabilmiştir. Kalan video karelerinde nesne ivmeölçer verisine dayanarak tespit edilmiştir. Öte yandan, 51. video karesinde nesne MobileNet SSD tarafından doğru bulunmasına rağmen, ivmeölçer verisine dayanarak üretilen BB nesnenin konumunu doğru göstermediği halde son karar olarak raporlanmıştır. Bu durumun sebebi hem örtüşme oranının hem de nesnelilik skorunun belirlenen eşik değerlerinin altında kalmasıdır. Bu durum ivmeölçer verisinin olumsuz etkilerine bir örnek olarak gösterilebilir.



Şekil 5.20 : Veritabanındaki 10. videoya ait 6 video karesi.

Son olarak Çizelge 5.2’de özellikleri ve Çizelge 5.4’te sonuçları verilmiş videolara <https://github.com/msprITU/MobileTracking> adresinden ulaşılabilir.

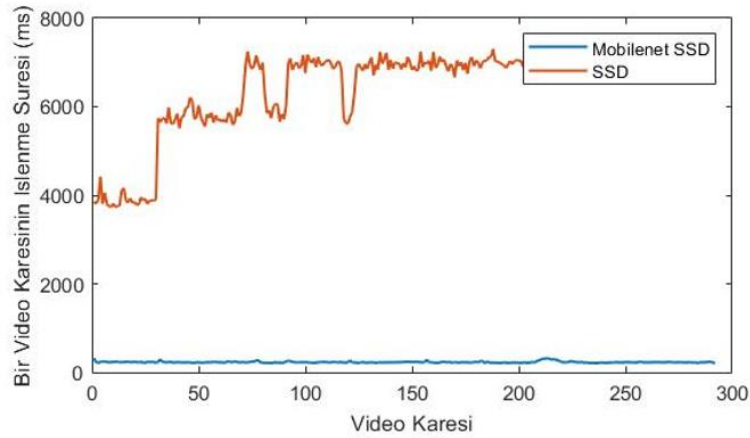
5.4 Gerçeklemenin Nesne İzleme Hızı

Bu tez çalışmasının bu alt başlığında kullanılan nesne tespit yöntemi çalışma hızı açısından incelenecektir. 2. başlıkta bu tez çalışmasında benzerlerinden çok daha hızlı olduğu için MobileNetv1 ile SSD’nin birleşiminden oluşan özel bir nesne tespit yönteminin kullanıldığından bahsedilmiştir. Yine 2. bölümde bu iki mimarinin nasıl birleştirildiği detayları ile anlatılmış olup standart SSD sürümü ile MobileNetv1 SSD sürümünün farklarına değinilmiştir.

Çizelge 2.6’da SSD’nin VGG-16 mimarisi kullanılan standart sürümü ile MobileNetv1 kullanılan özel sürümü ihtiyaç duyduğu toplam çarpma ve toplama

işlemi sayısı ve ihtiyaç duyduğu parametre sayısı bakımından karşılaştırılmıştır. Buna göre VGG-16 mimarisini kullanan standart sürüm MobileNetv1 mimarisini kullanan özel sürüme göre 4.9 kat daha fazla parametreye ihtiyaç duymaktadır. İhtiyaç duyulan parametre sayısı hem çalışma anında ihtiyaç duyulan belleği hem de eğitimin ardından oluşan model dosyasının büyüklüğünü doğrudan etkilemektedir. Bu tez çalışması kapsamında kullanılan standart SSD sürümüne ait model dosyası 103 MB, MobileNetv1 ile hızlandırılmış sürüme ait model dosyası ise 21.2 MB büyüklüğündedir. İki model dosyası arasındaki büyüklük farkı 4.85 kat olup Çizelge 2.6 ile uyumludur.

Öte yandan ise Çizelge 2.6'ya göre VGG-16 mimarisini kullanan standart sürüm MobileNetv1 mimarisini kullanan özel sürüme göre 29.1 kat daha fazla toplam çarpma ve toplama işlemi sayısına ihtiyaç duymaktadır. İhtiyaç duyulan toplam işlem sayısı da çalışma hızını doğrudan etkilemektedir. Şekil 4.21'de örnek bir video için bir Android akıllı telefon üzerinde her iki SSD mimarisinin de çalıştırılması sonucunda elde edilen hız bilgisi gösterilmektedir. Buna göre VGG-16 mimarisini kullanan standart SSD sürümünün bir video karesini işlemesi için gerekli süre ortalama 6420.2 ms olarak ölçülmüştür. Bu değer MobileNetv1 kullanan hızlandırılmış sürüm için ise 239.5 ms olarak ölçülmüştür. İki ortalama işlem süresi arasındaki fark 26.8 kat olup bu değer de Çizelge 2.6 ile uyumludur.



Şekil 5.21 : İki farklı SSD mimarisinin hız yönünden karşılaştırılması.

Son olarak ise Şekil 4.21'deki standart SSD sürümüne ait ölçümlerle ilgili küçük bir ayrıntıya değinmek gerekmektedir. Dikkat edilirse bu sürüme ait işlem süresi zamanla artmış ve belirli bir aralığa oturmuştur. Bunun sebebi ise bu sürümün üzerinde ölçüm alınan Android telefonu (Xiaomi Mi5s) çok zorlamasıdır. Zorlanan telefon zamanla

ısınmıştır ve aşırı ısınmayı engellemek için işlemcisinin saat frekansını düşürmüştür. Bunun sonucu olarak telefonun işlem gücü azalmış ve tek bir video karesini işlemek için gereken süre artmıştır.



6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasının önceki bölümlerinde de bahsedildiği gibi pek çok modern nesne izleme ve tespit yöntemi bilgisayarlara göre nispeten daha düşük işlem gücüne sahip olan mobil cihazlarda çalışmaya uygun olarak tasarlanmamıştır. Bu durum, mobil cihazların bilgisayarlara göre günlük hayatın içinde çok daha fazla yer aldığı düşünüldüğünde, modern nesne izleme ve tespit yöntemlerinin günlük hayatta kendilerine yer bulmalarını zorlaştırmaktadır. Bu sebeple bu tez çalışmasında kullanılacak nesne tespit yönteminin mobil cihazlarda da kabul edilebilir hızlarda çalışması istenmiştir. Bu amaçla nesne sezici olarak MobileNet v1 SSD kullanılmasına karar verilmiş olup bu yöntemin performansı ve çalışma hızı detaylı olarak raporlanmıştır.

Öte yandan mobil cihazlar bilgisayarlarda bulunmayan ve çevresel etkileri ölçmeye yarayan pek çok algılayıcıya sahiptir. Bu tez çalışmasında akıllı telefonlar başta olmak üzere hemen hemen tüm mobil cihazlarda bulunan ivmeölçer kullanılmıştır. Bu doğrultuda, öncelikle ivmeölçerden gelen ivme verisine çeşitli filtreleme işlemleri uygulanarak yerçekimi ivmesi ve ölçüm gürültüsünden kaynaklanan bozucu etkiler giderilmiştir. Ardından elde edilen anlamlı ivme verisi önce hız verisine dönüştürülmüş daha sonra bu hız verisi kamera hareketinin modellenmesinde ve kameranın hareket parametrelerinin bulunmasında kullanılmıştır. Kamera hareket parametrelerinin doğru bir şekilde bulunduğu öznitelik eşleştirme tabanlı bir yöntem kullanılarak gösterilmiştir. Son olarak, modellenen kamera hareketi kullanılarak hareketli kamera durumunda kamera hareketi kompanze edilerek nesnenin konumu hakkında tahminde bulunulmuştur. Bu tahmin nesne sezici tarafından yapılan tahmin ile tümleştirilerek nesne izleme performansında artış sağlanmıştır.

Tez çalışması kapsamında nesne sezici olarak kullanılan MobileNetv1 SSD'nin nesne tespit performansı MS COCO 2014 veri seti üzerinde incelenmiştir. Bu incelemede “recall”, “precision” ve F1 skoru üzerinden yapılmıştır. Buna ek olarak nesne izleme performansı VOT 2016 video veri seti üzerinde “accuracy” ve “robustness” metrikleri ile raporlanmıştır.

Gerçeklemeler için, Android Studio ortamı ve OpenCV yazılım kütüphaneleri kullanılarak çeşitli Android uygulamaları geliştirilmiş olup bu uygulamalar ile video çekimi, videoda nesne izleme ve tespiti gibi işlemler gerçekleştirilmiştir. Bu uygulamalar üzerinde MobileNet v1 SSD'nin standart SSD sürümünden yaklaşık 27 kat hızlı olduğu da gösterilmiştir. Ancak hedeflenen gerçek-zamanlı çalışmaya erişilememiştir. Bunun temel neden “multithread” gerçeklemesinin yazılım ortamında yapılamaması, bu nedenle video karesi ve ivmeölçer verileri okunurken paralel olarak nesne izleme algoritmasının çalıştırılamamış olmasıdır. Ancak hızda istenilen performansa erişememenin önerilen nesne izleyicinin performansına bir etkisi bulunmamaktadır.

VOT başta olmak üzere literatürdeki video veri tabanları ivme verisi içermediği için, tez çalışması kapsamında önerilen tümlşik yöntemin nesne izleme performansını incelemek adına günlük hayatın içinden cep telefonu ile çekilmiş videolardan oluşan özel bir veri seti oluşturulmuştur. Hazırlanan bu veri seti üzerinde, MobileNetv1 SSD nesne izleme performansının ivmeölçerden alınan geri besleme ile “robustness” metriği cinsinden %50'den fazla iyileştirildiği gösterilmiştir. Bunun temel nedeni nesne tespitinde kullanılan derin ağların video karesi bazında çalışması, videodaki hareket bilgisini kullanmamasıdır. İvmeölçer verisi ile nesne seziciye basit bir hareket modeli eklenerek videodaki zamansal bilginin modellenmesi sağlanabilmiş ve MobileNetv1 SSD nesne sezicinin bulamama oranı azaltılmıştır. Oluşturulan yeni veri seti <https://github.com/msprITU/MobileTracking> adresinde araştırmacıların kullanımına açılmıştır.

İleriye yönelik olarak önerilebilecek ilk çalışma, MobileNetv3 omurga ağ mimarisi kullanılarak SSD'nin nesne tespit hız ve doğruluğunun artırılmasıdır. Bir diğer öneri mobil cihazlardaki diğer sensörlerin de kullanılması ile performansın yükseltilmesidir. Tez çalışması kapsamında nesne izleme performansını arttırmak amacıyla ivmeölçer kullanılmıştır. Bu algılayıcıya ilave olarak jiroskop da kullanmanın, kamera hareketi modellenmesinin doğruluğunu arttıracığı için, nesne izleme performansını daha da arttırması beklenmektedir

Gerçeklenen mobil mimari modüler yapıdadır. Derin nesne tespit ağı olarak SSD kullanılmakla birlikte farklı ağlar da, örneğin YOLOv3 miimarisi kullanılarak nesne tespit hız ve doğruluğu arttırılabilir.

KAYNAKLAR

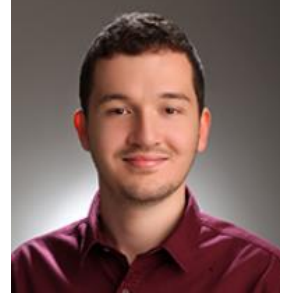
- [1] **Url-1** < <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html>>, erişim tarihi 26.06.2019.
- [2] **Girshick, R., Donahue J., Darrell T. ve Malik J.** (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 580-587). Columbus, Ohio, ABD: 24-27 Haziran.
- [3] **Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. ve Smeulders, A.W.M.** (2012). Selective Search for Object Recognition. *International Journal of Computer Vision*, (pp. 154-171). Santiago, Şili: 10-13 Aralık.
- [4] **Girshick, R.** (2015). Fast R-CNN. *The IEEE International Conference on Computer Vision (ICCV)*, (pp. 1440-1448). Santiago, Şili: 13-16 Aralık.
- [5] **Ren, S., He, K., Girshick, R ve Sun, J.** (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Kanada: 7-12 Aralık.
- [6] **Dai, J., Li, Y., He, K. ve Sun, J.** (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. *Neural Information Processing Systems (NIPS)*. Barcelona, İspanya: 5-11 Aralık.
- [7] **He, K., Gkioxari, G., Doll'ar, P. ve Girshick R.** (2017). Mask R-CNN. *The IEEE International Conference on Computer Vision (ICCV)*, (pp. 2961-2969). Venedik, İtalya: 22-29 Ekim.
- [8] **Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. ve Sun J.** (2017). Light-Head R-CNN: In Defense of Two-Stage Object Detector. *arXiv:1711.07264v2*.
- [9] **Redmon, J., Divvala, D., Girshick, R. ve Farhadi, A.** (2016). You Only Look Once: Unified, Real-Time Object Detection. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 779-788). Las Vegas, ABD: 26 Haziran – 1 Temmuz.
- [10] **Redmon, J. ve Farhadi, A.** (2017). YOLO9000: Better, Faster, Stronger. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 7263-7271). Honolulu, Hawaii: 22-25 Temmuz.
- [11] **Redmon, J. ve Farhadi, A.** (2018). YOLOv3: An Incremental Improvement. *arXiv:1804.02767v1*.
- [12] **Erhan, D., Szegedy, C., Toshev, A. ve Anguelov.** (2014). Scalable Object Detection using Deep Neural Networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2147-2154). Columbus, Ohio, ABD: 24-27 Haziran.

- [13] **Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. ve Berg. A. C.** (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision*, (pp. 21-37). Amsterdam, Hollanda: 8-16 Ekim.
- [14] **Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. ve Adam H.** (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861v1*.
- [15] **Szegedy, C., Ioffe, S. ve Vanhoucke, V.** (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Thirty-First AAAI Conference on Artificial Intelligence*, (pp. 4278-4284). San Francisco, California, ABD: 4-9 Şubat.
- [16] **Chollet, F.** (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 1251-1258). Honolulu, Hawaii: 22-25 Temmuz.
- [17] **Lin, T., Goyal, P., Girshick, R., He, K. ve Doll'ar, P.** (2017). Focal Loss for Dense Object Detection. *The IEEE International Conference on Computer Vision (ICCV)*, (pp. 2980-2988). Venedik, İtalya: 22-29 Ekim.
- [18] **Simonyan, K., ve Zisserman, A.** (2015). Very Deep Convolutional Networks For Large-Scale Image Recognition. *arXiv:1409.1556v6*.
- [19] **He, K., Zhang, X., Ren, S. ve Sun, J.** (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1904-1916.
- [20] **He, K., Zhang, X., Ren, S. ve Sun, J.** (2016). Deep Residual Learning for Image Recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 779-788). Las Vegas, ABD: 26 Haziran – 1 Temmuz.
- [21] **Lin, T., Doll'ar, P., Girshick, R., He, K., Hariharan, B. ve Belongie, S.** (2017). Feature Pyramid Networks for Object Detection. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2117-2125). Honolulu, Hawaii: 22-25 Temmuz.
- [22] **Zhang, X., Zhou, X., Lin, M. ve Sun, J.** (2018). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 6848-6856). Salt Lake City, Utah, ABD: 18-22 Haziran.
- [23] **Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., ve Chen, L.** (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 4510-4520). Salt Lake City, Utah, ABD: 18-22 Haziran.
- [24] **Howard, A., Sandler, M., Grace, C., ...Adam H.** (2019). Searching for MobileNetV3. *arXiv: 1905.02244v4*.
- [25] **Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. ve Murphy, K.** (2017). Speed/accuracy trade-offs for modern convolutional object detectors.

- The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 7310-7311). Honolulu, Hawaii: 22-25 Temmuz.
- [26] **Url-2** < <https://www.youtube.com/watch?v=T7o3xvJLuHk>>, erişim tarihi 26.06.2019.
- [27] **Url-3** < https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords>, erişim tarihi 30.04.2018.
- [28] **Url-4** < http://www.starlino.com/imu_guide.html>, erişim tarihi 26.06.2019.
- [29] **Url-5** < https://developer.android.com/guide/topics/sensors/sensors_motion.html>, erişim tarihi 26.06.2019.
- [30] **Url-6** < <https://i.redd.it/hrr79vpb0m601.png>>, erişim tarihi 06.07.2019.
- [31] **Harris, C. ve Stephens, M.** (1988). A Combined Corner and Edge Detector. *Proceedings of the 4th Alvey Vision Conference*, (pp. 147-151). Manchester, İngiltere: 31 Ağustos – 1 Eylül.
- [32] **Alahi, A., Ortiz, R. ve Vandergheynst, P.** (2012). FREAK: Fast Retina Keypoint. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 510-507). Rhode Island, ABD: 16 – 21 Haziran.
- [33] **Url-7** < <https://www.android.com>>, erişim tarihi 26.06.2019.
- [34] **Url-8** < <https://www.jetbrains.com>>, erişim tarihi 26.06.2019.
- [35] **Url-9** < <https://developer.android.com/studio/>>, erişim tarihi 26.06.2019.
- [36] **Url-10** < <https://opencv.org>>, erişim tarihi 26.06.2019.
- [37] **Url-11** < <https://developer.android.com/about/versions/oreo/android-8.1>>, erişim tarihi 26.06.2019.
- [38] **Url-12** < <http://caffe.berkeleyvision.org>>, erişim tarihi 26.06.2019.
- [39] **Url-13** < <https://github.com/chuanqi305/MobileNet-SSD>>, erişim tarihi 26.06.2019.
- [40] **Lin, T., Maire, M., Belongie, S., Bourdev, I., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. ve Dollar, P.** (2014). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision*, (pp. 740-755). Zürich, İsviçre: 6-12 Eylül.
- [41] **Cehovin, L., Leonardis, A. ve Kristan, M.** (2016). Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 1261-1274.
- [42] **Url-14** < <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>>, erişim tarihi 26.06.2019.
- [43] **Url-15** < <http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUHJlY2lzaW9uX2FuZF9yZWVhbnGw>>, erişim tarihi 06.07.2019.
- [44] **Kristan, M., Leonardis, A., Matas, J., ... Chi. Z.** (2016). The Visual Object Tracking VOT2016 Challenge Results. *European Conference on Computer Vision*, (pp. 777-823). Amsterdam, Hollanda: 8-16 Ekim.



ÖZGEÇMİŞ



Ad-Soyad : Ulaş TOSUN
Doğum Tarihi ve Yeri : 15.04.1993 / Giresun
E-posta : ulastosun28@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** : 2016, İstanbul Teknik Üniversitesi, Elektrik Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü
- 2016, İstanbul Teknik Üniversitesi, Elektrik Elektronik Fakültesi, Kontrol ve Otomasyon Mühendisliği Bölümü

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2016 – 2017 yıllarında Arçelik AŞ’de AR – GE mühendisi olarak çalıştı.