T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAVŞAK VE KARAYOLLAR İÇİN GERÇEK ZAMANLI GÖRÜ TABANLI TRAFİK AKIŞ BİLGİSİ HESAPLAMA SİSTEMLERİNİN GELİŞTİRİLMESİ

DOKTORA TEZİ

Jahongir AZİMJONOV

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. Ahmet ÖZMEN**

Eylül 2021

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAVŞAK VE KARAYOLLAR İÇİN GERÇEK ZAMANLI GÖRÜ TABANLI TRAFİK AKIŞ BİLGİSİ HESAPLAMA SİSTEMLERİNİN GELİŞTİRİLMESİ

DOKTORA TEZİ

Jahongir AZİMJONOV

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 13.09.2021 tarihinde aşağıdaki jüri tarafından oybirliği / ~~oyçokluğu~~ ile kabul edilmiştir.

Jüri Başkanı **Üye** **Üye**

Üye **Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Jahongir AZIMJONOV

27.08.2021

TEŞEKKÜR

Doktora eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Prof. Dr. Ahmet ÖZMEN'e teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

BG	: Bilgisayarla görü
CCTV	: Kapalı devre televizyon kameraları (Closed-circuit television)
CNN	: Evrişimsel sinir ağları (Convolutional neural networks)
DL, ML	: Derin öğrenme, Makine öğrenmesi
GPS	: Küresel konumlama sistemi
GSYİH	: Gayri safi yurt içi hâsıla
ITS	: Akıllı ulaşım sistemleri (Intelligent transportation systems)
LOOP	: Metal kütle dedektörü
LIDAR	: Lazer görüntüleme algılama ve menzil
RADAR	: Birleştirilmiş, son radar görüntüleri
RCNN	: Bölge tabanlı evrişimsel sinir ağı
RFID	: Radyo frekansı ile tanımlama
SSD	: Tek atış dedektörü
YOLOv3	: You only look once v3 nesne tespit algoritması
YZ	: Yapay zekâ

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Araç (Nesne) Tespiti, Araç (Nesne) Takibi, Araç İz Verisi Çıkarma, Trafik Akış Bilgisi Hesaplama.

Bu çalışmada, şehir içi trafik planlamasında yardımcı araç olarak kullanılabilecek, mevcut trafik hakkında istatistiki veri üretebilen bir yazılım sistemi, görüntü işleme yöntemleri kullanılarak geliştirilmiştir. Kavşak/otoyol trafik görüntüleri kameralarla elde edilmiştir. Normalde, şehirlerdeki trafik sıkışıklığının en yoğun olduğu yerler araçların paylaşmak zorunda olduğu kavşak, şehir içi yol veya otoyol bölgeleridir. Bundan ötürü, bu çalışmada örnek kavşak ve otoyollara odaklanılmıştır. Kavşak ve otoyollardan ayrı ayrı elde edilen çevirimiçi trafik görüntü verileri yapay zekâ teknikleriyle işlenerek şehrin seçilmiş bölgelerinin trafiği hakkında anlık bilgiler hesaplanmıştır. Örneğin, şehir içi yol/kavşaklardan geçen araçların sınıflandırılması (otomobil, kamyonet, otobüs gibi) ve her bir araç sınıfına ait sayısal değerlerin bulunması, araçların gittikleri yön ve hız bilgileri hesaplanmıştır. Çalışmada geliştirilen çevirimiçi görüntü işleme yazılım sistemi için derin öğrenme ve makine öğrenmesi teknikleri kullanılmıştır. Veri işleme birimi; nesne tespit etme, nesne takip etme, nesne ilişkilendirme (nesnelerin zamana bağlı izlerinin çıkarılması) ve ilişkilendirilmiş nesne verilerinden hız, sayım (araç türüne göre sayım ve toplam sayım), araçların belirlenen bölgeye giriş ve çıkış noktaları ve bu noktalar arasında geçen süre gibi değerli trafik akış bilgilerini çıkarma aşamalarından oluşmaktadır. Nesne algılama birimi için evrişimsel sinir ağlarına (CNN) dayanan YOLOv3 mimarisi kullanılmıştır. YOLOv3 algoritması, tekrar eğitilerek, çalışmaya özgü ağırlık modelleri oluşturulumuştur. Eğitilmiş yeni ağırlık modeli temel alınarak, kameralarla mevcut saha çalışmalarından elde edilen görüntülerden araç tanıma ve sınıflandırma işlemleri yapılmıştır. Nesne tespit ve sınıflandırma aşamasının çıktıları olan sınırlayıcı kutu bilgileri, nesne takip ve iz çıkarım modülünün girdisi olarak kullanılmıştır. Araçların iz verilerinden trafik akış bilgileri hesaplanmıştır. Geliştirilen bu yazılım sisteminin ürettiği çevirimiçi çıktılardan, şehir trafik analizi yapılabilecek veya elde edilen veriler benzetim yazılım araçları için girdi olarak kullanılabilecektir (örneğin PTV Vissim). Simülasyon araçlarının doğru sonuçlar verebilmesi ancak doğru istatistiki verilerle olabilir, bu bakımdan sağlıklı şehir trafik planlaması için bu çalışma önem arz etmektedir. Proje kapsamında yapılan çalışmalar ve geliştirilen yazılım aracı Sakarya Büyükşehir Belediyesine (SBB) sunulmuştur. Akıllı şehir vizyonu uygulama kentleri arasında seçilen SBB, çalışmaya ilgi göstermiş, trafik ve sinyalizasyon projeleri kapsamında destekleme kararı almıştır. Projenin çıktıları doğrudan şehrin trafik planlamasına katkıda bulunacak olup, dolaylı olarak, şehir yaşam alanı kalitesinin arttırılması, emisyon salınımlarının ve trafik gürültüsünün azaltılmasına katkı sağlayacağı öngörülmektedir. Bu çerçevede Sakarya Büyükşehir Belediyesi ve Sakarya Üniversitesi arasındaki işbirliğinin güncel teknolojiler kullanılarak arttırılması da hedeflenmiştir.

DEVELOPMENT OF REAL-TIME VISION BASED TRAFFIC FLOW INFORMATION ESTIMATION SYSTEMS FOR INTERSECTION AND HIGHWAYS

SUMMARY

Keywords: Vehicle Detection, Vehicle Tracking, Vehicle Trajectory Extraction, Traffic Flow Information Calculation.

In this study, a real-time traffic flow monitoring system that can be used in planning of urban traffic and can generate statistical traffic flow data was developed using image processing methods. Intersection/highway traffic images were obtained with cameras. Normally, the highly congested places in cities are intersections, roads or highway areas where vehicles have to share. Hence, this study focused on intersections/highways. Online traffic images obtained separately from intersections and highways were processed with AI techniques, and instant information about the traffic of selected areas of the city was calculated. For example, the classification of vehicles passing through city roads/intersections (such as cars, trucks, buses) and number (for example, frequency information) of each vehicle class, the direction and speed information of the vehicles were estimated. Deep/machine learning techniques were utilized for the online software system developed in the study. The image processing module; object detection, tracking and association (extraction of time-dependent vehicle trajectory) and traffic flow data such as speed, counts (directional and total vehicle counts), entry and exit points of vehicles to the specified area, and the time period between these points. YOLOv3 architecture based on convolutional neural networks (CNN) was used for the object detection task. By retraining the YOLOV3 algorithm, case-specific weight models were created. Based on the trained new weight models, vehicle detection (localization and classification) tasks were performed from the traffic images obtained from case study intersection and highways. Bounding box information, which is the output of the object detection phase, was used as the input of the object tracking and trajectory extraction modules. Traffic flow information was estimated from the vehicle trajectory data. City traffic can be analyzed from the real-time outputs of this developed software system or the obtained output data can be used as input for the traffic analysis simulation software tools (for example, PTV Vissim). The simulation tools can give accurate results only with the right statistical data, so this study is important for robust city traffic planning. The studies carried out within the scope of the project and the software tool developed were presented to Sakarya Metropolitan Municipality (SBB). SBB, which was selected among the cities to implement the smart city vision, interested in the study and decided to support it within the scope of traffic and signaling projects. The outputs of the project will directly contribute to the traffic planning of the city, and indirectly, it is anticipated that it will contribute to increasing the quality of city living space, reducing emissions and traffic noise. In this context, it is also aimed to increase the cooperation between Sakarya Metropolitan Municipality and Sakarya University by using up-to-date technologies.

BÖLÜM 1. GİRİŞ

1.1. Problem Tanımı

Birleşmiş Milletler kurumu tarafından hazırlanmış yeni bir istatistikte yer alan verilere göre, dünya nüfusunun %55'i (3,5 milyardan fazla insan) şehirlerde veya kentsel alanlarda yaşıyor ve bu sayının 2050 yılına kadar %70'a ulaşacağı tahmin edilmektedir. Nüfus artışının yanı sıra kişi başına düşen Gayri Safi Yıl İçi Hasıladaki (GSYİH) artışın tetiklediği yükselen yaşam standartlarının doğal sonucu olarak; şehir ve kentsel alanlardaki kavşak, yol ve otoyol trafiği daha sıkışık hale gelmesi öngörülmektedir (Milletler, 2019). Trafik sıkışıklığı (yol trafiğinin yavaş ilerlemesi veya durağanlığı, uzun zamanlı sürüş süreleri ve şehir içi yollardaki uzun gecikmelerle karakterize edilen bir trafik akışı durumu) kentsel alanlardaki yaşam kalitesi üzerinde karbon emisyonlarından ekonomik maliyete kadar; şehir içi yollar ve kentsel alan otoyollarında kayıbedilen saatler ve yakıt israfı gibi çok boyutlu olumsuz etkilere sahiptir.

Ek olarak, önde gelen bir gerçek zamanlı trafik bilgisi toplama şirketi olan INRIX tarafından yürütülen yeni bir çalışma, trafik sıkışıklığının; ülkelerin veya yerel yönetimlerin ekonomisi üzerindeki gizli maliyetini ortaya koymaktadır ki, sadece ABD'de, trafik akış yavaşlığı veya trafik sıkışıklığı sorunlarının neden olduğu üretkenlik kaybının toplam maliyeti, 2019 yılı için 87 milyar ABD dolarından fazladır. Bir tek, ABD'nin Boston eyaletinde görülen mali kayıp 4,1 milyar dolardan fazla olduğu hesaplanmıştır. Dünyanın diğer (Tokyo – Japonya, Moskova – Rusya, İstanbul, Ankara ve İzmir – Türkiye, ve Kuala Lumpur – Malezya) gibi bölgeleri de trafik tıkanıklığı nedeniyle büyük miktarda mali kayıplar vermektedir (Inrix, 2019). Bu nedenle, her şehirdeki trafik sıkışıklığının azaltılması yaşam kalitesinin iyileştirilmesi için hayati önem taşımakta olup, kayıpları minimum seviyeye

indirmek bir ihtiyaç haline gelmiş bulunmaktadır. Neyseki, akıllı veya akıllı sürdürülebilir şehir kavramı altında sunulmakta olan akıllı yaklaşım ve uygulamalar, yukarıda sıralanan güncel sorunlara optimal çözümler üretmeyi ve şehir içi veya kentsel alanlardaki yaşam kalitesini (maliyet ve kayıpları en aza indirerek) arttırmayı hedeflemektedir.

1.2. Motivasyon ve Çalışmanın Amacı

Akıllı veya akıllı sürdürülebilir şehir konsepti, günümüzde yoğun nüfuslu kentsel alanlar ve şehirlerin enerji, sağlık, ulaşım alanlarında meydana gelmekte olan gerçek sorunlara bir dizi umut verici çözümler içermektedir (Lombardi ve ark., 2012). Bu çözümlerden biri; akıllı ulaşım sistemleridir (ITS). ITS; şehirleri veya kentsel alanlarda meydana gelen maliyet kaybı veya diğer kayıpları en aza indirmek suretiyle daha akıllı, daha sürdürülebilir mekanlara dönüştürebilecek kapasiteye sahip olduğundan, akıllı sürdürülebilir şehirler kavramının çok önemli bir bileşeni olarak karşımıza çıkmaktadır (Chen ve ark., 2017). Dahası, şehirlerde kurulu sensörlerden ve kameralardan toplanan muazzam miktardaki büyük verileri akıllı araç, algoritma ve metotlar yardımıyla işleyerek üretilen akıllı çözüm ve teknolojilerin etkin bir biçimde kullanılması; şehir ve kentsel alanların güvenliğini artırmaya, ve bu bölgelerdeki karbon emisyonu ve trafik tıkanıklığını azaltmaya veya tamamen ortadan kaldırmaya hizmet edeceği beklenmektedir (Ismagilova ve ark., 2019). Bu tür akıllı ulaşım sistemleri yediden fazla uygulama alanını kapsamaktadır, bu uygulama alanlarının en önemli bir uygulaması ise "Gelişmiş Trafik Yönetim Sistemi" veya "Dinamik Trafik Akış İzleme ve Kontrol Sistemleri (DTAİKS)" olarak bilinmektedir (Sullivan, 2018).

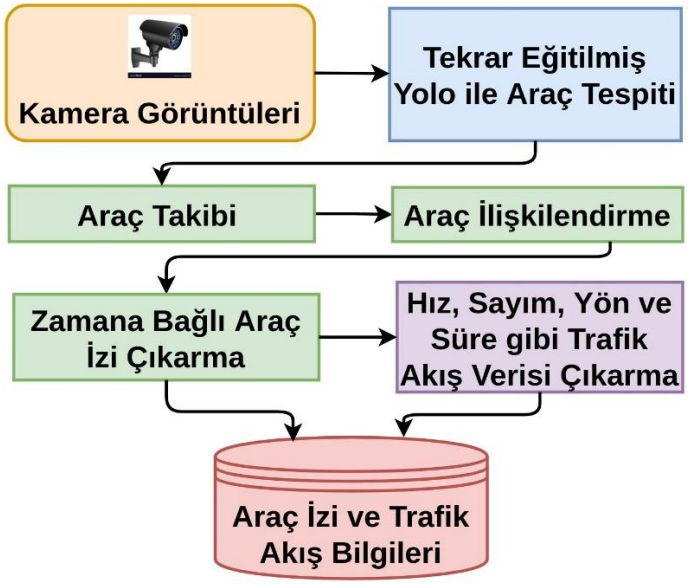
Dinamik trafik akış izleme sistemi; veri toplama (ham trafik verilerini toplama) ve veri işleme (ham trafik verilerinden zengin bağlamsal trafik akışı bilgilerinin çıkarılması) olarak adlandırılan iki temel aşamadan oluşmaktadır. Bu iki adım dinamik trafik akış izlemeyi mümkün kılarak kavşak, şehir içi yol ve otoyollarda ulaşım koşullarını iyileştirmeye hizmet etmektedir. İlk aşama olan trafik verilerini

toplamada, trafik verileri sensör veya kamera(lar) yardımıyla doğru ve titiz bir biçimde elde edilmektedir. İkinci adımda, toplanan veriler gerçek zamanlı yapay zeka (YZ) algoritmaları ile işlenmektedir. Üçüncü aşamada, araç tipleri, yörüngeler, hız, şerit değişiklikleri, sürücü davranışları gibi gerekli tüm gerçek zamanlı trafik akış bilgileri çıkarılmaktadır (Zheng ve ark., 2019).

Geçmişte, otomatik trafik izleme sistemleri için geliştirilmiş çözümler, hem donanım hem de yazılım araçlarının yetersiz kalması nedeniyle, tüm bu bilgileri gerçek zamanlı olarak çıkaramamakta idi. Ancak, yapay zekada kullanılan grafiksel işlem birimleri (binlerce çekirdekten oluşan yüksek görüntü verisi işleme gücüne sahip GPU) gibi donanımlardaki son gelişmeler ve bilgisayarla görme (BG), derin öğrenme (DÖ) ve makine öğrenmesi (MÖ) gibi yeni yapay zeka algoritmaları bu alanda gerçek zamanlı teknoloji ve uygulama geliştirme işlemlerini mümkün kılmış ve kolaylaştırmıştır. Bu alanlardaki gelişme ve iyileşmeler sayesinde araç algılama, araç izleme ve veri ilişkilendirme dahil olmak üzere YZ tekniklerini yararlanarak gerçekleştirilen birçok işlem kolay ve masrafsız yapılabilmektedir. Bunun yanı sıra tüm gerekli ve zengin bağlamsal trafik akışı bilgileri çıkarılabilmektedir (El Mokaddem ve ark., 2019). Bu tür trafik akışı bilgileri, görüntü işlemenin nesne (araç) tespit ve takip etme yaklaşımları kullanılarak ham video görüntülerinden çıkarılmaktadır.

Araç tespiti; bir görüntüdeki veya bir videonun belirli bir karesindeki nesnelerin türünü ve konumunu tahmin ederek belirleme işlemidir. Araç takibi ise tespit edilen araçların çerçeveler aracılığıyla doğru eşleriyle ilişkilendirilmesidir. Bu iki görev, klasik bilgisayar görüşü ve görüntü işleme araştırmasının zorlu görevlerinden biri olmuştur. Çünkü, farklı renkler, şekiller, gündüz-gece aydınlatması, anormal parlaklık, nesnelerin kısmen veya tamamen kapatılması, sensör veya kamera titremesi, yağmur, sis, kar ve rüzgar gibi farklı hava koşulları, ve aşırı yüksek veya düşük kaliteli resimler, araç algılama ve izleme süreçlerini zorlaştırmaktadır. Ve bazen de bu sorunlar nesne algılama ve izlemeyi neredeyse imkansız hale getirmektedir (Datondji ve ark., 2016). Dolayısıyla, sağlıklı ve düzgün bir trafik akış

izleme sisteminin geliştirilebilmesi için sağlam, hızlı, düzenli ve yüksek doğrulukla çalışan YZ tabanlı yenilikçi donanım ve yazılımlara ihtiyaç duyulmaktadır.



Şekil 1.1. Trafik Akış İzleme Sisteminin Genel Mimarisi.

Bu çalışmada, çevirimiçi video işleme sistemlerinin karşı karşıya oldukları yukarıda sıralanan sorunlar için modern veri işleme yöntemlerinden yapay zeka, özellikle yapay zekanın derin öğrenme ve makine öğrenmesi teknikleri kullanılarak görme tabanlı (vision-based) araç tespit, takip, sayım ve hız ölçme çözümleri üretilmiştir. Görüntü verisi işleme birimi; nesne tespit etme, nesne takip etme, nesne ilişkşlendirme (nesnelerin zamana bağlı izlerinin çıkarılması) ve ilşkilendirilmiş nesnelerin iz verilerinden hız, araç türüne göre sayım, toplam araç sayımı ve araçların belirlenen bölgeye giriş ve çıkış noktalarının belirlenmesi ve bu noktalar arasındaki zaman dilimi gibi çok kıymetli trafik akış bilgilerini çıkarma aşamalarından oluşmaktadır, Şekil 1.1. Nesne algılama birimi için evrişimsel sinir ağlarına (CNN: Convolutional Neural Network) dayanan mimarisi kullanılmıştır (Redmon ve ark., 2016). YOLOv3 algoritması, bu çalışma için tekrar eğitilerek, çalışmada kullanılan kavşak ve otoyollara özgü araç algılama ağırlık modelleri oluşturulumuştur. Eğitilmiş yeni ağırlık modeli temel alınarak mevcut saha çalışmalarında elde edilen görüntülerden araç tanıma ve sınıflandırma işlemleri yapılmıştır. Nesne tespit ve sınıflandırma aşamasının çıktıları olan nesnelerin konum

ve büyüklükleri, türleri ve güven değerleri gibi sınırlayıcı kutu bilgileri nesne takip ve iz çıkarım modülünün girdisi olarak kullanılmış ve zamana bağlı olarak çıkartılan nesne iz verilerinden hız, sayım ve araçların belirlenen bölgeye giriş ve çıkış noktaları ve bu noktalar arasındaki zaman süresi gibi trafik akış verileri elde edilmiştir.

1.3. Araştırmanın Kısıtları

Geliştirilmiş trafik akış bilgisi hesaplama yazılımı, gerçek zamanlı trafik görüntü verileri ile direk ilişkili olması nedeniyle avantajlarının yanında bir takım kısıtları da içermektedir. Mesela, araç tespit ve takip etme birimlerinin yüksek işlem gücü gerektirmesi sebebiyle performans, hız ve donanım sıkıntıları ortaya çıkmaktadır. Bu nedenle, bu çalışmada olduğu gibi gerçek zamanlı bir performans elde edilebilmesi için en az 8 GB hafızalı, 16 GB RAM özelliklere sahip bilgisayarın olması gereklidir. Onun dışında, bir trafik akış bilgisi hesaplama sistemini doğrudan etkileyen araç tespit ve takip modülleri; yanlış açılı kamera konumlandırma, nesnelerin üst üste örtüşmesi, parlaklık veya aşırı kötü hava koşulları gibi görüntü işleme yaklaşımlarının sıkıntılarıyla karşı karşıya kalabilmektedir.

BÖLÜM 2. BENZER ÇALIŞMALAR

2.1. Akıllı Ulaşım Sistemleri

Akıllı Ulaşım Sistemleri (ITS); bilgi ve telekomünikasyonun en çağdaş, en yeni teknolojik çözümlerini kullanarak akıllı ulaşım imkanlarını sağlayan yapılardır. Başka bir deyişle, ITS; trafik tıkanıklığını gidererek, veya en aza indirerek, kentsel alanlardaki veya şehirlerdeki yoğun trafiği hafifleten, yol güvenliğini, verimliliği ve konforunun yanı sıra çevrenin korunmasına önemli ölçüde katkıda bulunmak üzere tasarlanmış; insanlar, yol ve araç sistemlerinin bütünleştiği mimarilerdir. Akıllı ulaşım sistemleri, veri toplama ve veri analizi olmak üzere iki ana bölümden oluşmaktadır.

Veri toplama aşaması, yol ve kavşaklardan; endüktif döngü dedektörü (LOOP), video görüntüsü çekme cihazları (cameras), pnömatik borular, küresel konumlandırma sistemi alıcısı (GPS), akustik/ultrasonik sensör, havadan/uydudan görüntüleme cihazları, radyo frekansı tanımlama teknolojisi (RFID) gibi algılayıcılar yardımıyla gerçekleştirilmektedir (Zheng ve ark., 2019). Bu algılayıcıların arasında, çok boyutulu bilgi toplama açısından en avantajlı ve en az maliyetli olanı gece/gündüz görüntülerini çekme kabilyetine sahip olan kameralardır.

Veri analizi aşamasında ise eskilerde olduğu gibi, daha düşük doğruluklu daha yavaş çalışan geleneksel makine öğrenmesi veya görüntü işleme metotlarının yerine daha yüksek doğruluklu daha hızlı algoritma ve metotlar kullanılmaktadır. Dolayısıyla, birçok trafik akış izleme sisteminde görüntü tabanlı algoritma ve tekniklerinin kullanılması daha uygun bulunmuş ve görme tabanlı sistemlerin geliştirlmesi ve uygulanması hız kazanmış bulunmaktadır. Mesala, akıllı ulaştırma sistemleri; Gelişmiş Trafik Yönetim Sistemleri (ATMS), Gelişmiş Toplu Taşıma Sistemleri (APTS), Gelişmiş Gezgin Bilgi Sistemi (ATIS), Gelişmiş Nakliye Fiyatlandırma

Sistemleri (ATPS), Ticari Araç Operasyonları (CVO), Acil Durum Yönetimi (EM), Bakım ve İnşaat Yönetimi (MCM) gibi birçok uygulama alanlarını tek çatı altında toplamakta ve bu sistemlerin tamamında YZ yaklaşımlarını esas alan yenilikçi görüntü işleme algoritmlarını kullanmaktadır (Sullivan, 2018). Aşağıda, bu uygulamalar (ALT sistemler) üzerinde kısaca bahis edilecektir.

Gelişmiş trafik yönetim sistemleri, kentsel alan veya şehirlerde bulunan kavşak veya yollarda meydana gelen trafik tıkanıklığını dinamik olarak izleyip yöneten bir uygulamadır. Bu tür sistemler, akıllı teknoloji ve haberleşme çözümlerini etkin bir biçimde kullanmak suretiyle, mevcut ulaştırma altyapılarının kullanım verimliliğini arttırmayı hedeflemektedir. Bu sistemler hem tekrarlayan (yoğun saat trafiği) hem de tekrar etmeyen (kazalar nedeniyle oluşan tıkanıklık, durmuş araçlar gibi) trafik koşullarını yollara veya kavşaklara yerleştirilmiş çeşitli yol algılayıcıları aracılığıyla izlemekte ve yollarda oluşan tıkanıklığı azaltmak için trafik akışını dinamik olarak kontrol ederek en optimal çözümler üretmektedir (Shahgholian ve Gharavian, 2018).

Gelişmiş Toplu Taşıma Sistemleri (APTS), toplu taşıma sistemleri, işe gidip gelme süresini azaltırken güvenliği, güvenilirliği ve verimliliği artırmayı ve böylece trafik tıkanıklığı ve emisyonları azaltmayı amaçlayan bir teknoloji koleksiyonudur. Yöneticilere gerçek zamanlı konum takibi, kaza/olay bilgisi, sürücü ve araç izleme gibi önemli avantajlar sunma potansiyeline sahiptır (Adeleke ve ark., 2013).

Gelişmiş Yolcu Bilgi Sistemi (ATIS), en yaygın ITS hizmetlerinden biridir ve yolcuların başlangıç noktalarından varış noktalarına ulaşmalarına yardımcı olur. Trafik sıkışıklığı, gecikmeler, kazalar ve hava durumu ile ilgili bilgiler medya, İnternet, görsel mesajlaşma ve kamuya açık anons sistemleri aracılığıyla yayılmaktadır (Kem ve ark., 2017).

Gelişmiş Nakliye Fiyatlandırma Sistemleri (ATPS), belirli saatlerde ücretler uygulayarak tıkanıklığı kontrol etmek için bir yöntem olarak kullanılan ITS'in uygulama alt dallarından biridir. Mesala, elektronik yol fiyatlandırma (ERP) sistemi bu uygulamaların bir örneğidir. Trafik sıkışıklığı olarak belirtilen bölge ve belirli

saatlerde araç kullanıcılarının bu bölge ve zaman diliminde sadece bulunduğu süre için ödeme yapmasına olanak tanıyan bir sistemdir (Andersen ve Sutcliffe, 2000).

Ticari Araç Operasyonları (CVO), ticari araç operasyonlarının verimli yönetimine yardımcı olan ticari araçlar için bir ITS çözümüdür. Ticari araçları yönetmek için dijital radyolar ve akıllı algoritmalarla birlikte GPS konumlarından yararlanmaktadır (Hall ve Intihar, 1997).

Acil Durum Yönetimi (EM), acil tıbbi hizmetler, büyük ve küçük ölçekli acil müdahale, acil durum araçlarının yönlendirilmesi ve yolcuları felaketler hakkında bilgilendiren bir ITS uygulamasıdır. Bu tür sistemler, acil durum araçlarına yerleştirilmiş kamera veya RFID gibi sensörleri kullanarak gerçek zamanlı yol ve trafik durum bilgilerinin aktarımında da bulunabilmektedir (Martinez ve ark., 2010).

Bakım ve İnşaat Yönetimi (MCM), bir bölgedeki yolları korumak ve inşaatı yönetmek için kullanılan ITS'in bir alt dal uygulamasıdır. Buna, kar temizleme veya yol onarımları birer örnekler olarak verilebilir (Reddy ve ark., 2018).

Akıllı ulaşım sistemleri üzerinde 1990'lı yıllardan beri etkin olarak araştırma ve geliştirme çalışmaları yapılmaktadır. Fakat, yetersiz kalan bilgisayar donanımları veya hesaplamalı zeka algoritmalarından dolayı günümüzde olduğu kadar yaygın kullanılamıyordu. Ancak, son yıllarda bilgisayar ve hesaplama cihazları üretimi ve yapay zeka algoritmalarındaki yenilikçi gelişmeler, ve bu donanım ve yazılım teknolojilerinin fiyatlarındaki kayda değer düşüş nedeniyle bu yeni teknoloji ve hesaplamalı zeka çözümlerinin akıllı ulaştırma sistemlerindeki kullanımı çok hızlı yaygınlaşmakta ve trafik izleme ve yönetme işlemleri yüksek doğrulukla yapılabilmektedir. Mesala, Çin'in Hangzhou trafik yönetim birimi, şehrin aşırı tıkanık olan trafik akışını, yenilikçi yapay zeka algoritmalarını kullanarak geliştirdiği dinamik ve gerçek zamanlı trafik akış izleme ve yönetme sistemleri sayesinde %11 oranında hızlandırmış olup, dünyanın trafiği en tıkanık olan şehirleri listesinde 5. sıradan 57. sıraya geriletebilmiştir. Tıpki Çin'de olduğu gibi, trafik akışı kameralarla izlenerek, çekilen görüntü verileri gerçek zamanlı yapay zeka sistemleriyle aktif bir

şekilde analiz edilip, araçların hareket yönü, hızı ve sayımı doğru bir biçimde yapıldığında, şehir içi trafik sıkışıklığını azaltabilme, veya tamamen ortadan kaldırabilme kapasitesine sahip, faydalı bir trafik akış izleme ve yönetim modeli meydana gelecek ve trafik akışı önemli ölçüde kontrol altına alınmasının mümkün olabileceği öngörülmektedir. Bunun için güçlü donanım ve araç hareketlerini anında tespit edip, davranışlarını tahlil edebilen yazılımlara ihtiyaç duyulmaktadır. Sıradaki alt bölümde, bu ihtiyaç duyulan sistemler; dinamkik ve gerçek zamanlı trafik akış izleme ve yönetme sistemlerinden ayrıntılı olarak bahis edilecektir.

2.2. Trafik Akış İzleme Sistemleri ve Bileşenleri

Trafik akış izleme sistemleri, şehir veya kentsel alanlardaki şehir içi yollar, otoyollar veya kavşaklara yerleştirilmiş çeşitli algılayıcılar, yüksek başarımlı bilgisayar donanımları ve en yeni yazılım araçlarını kullanarak bu bölgelerden trafik akış verilerini gerçek zamanlı veya çevirimdişi olarak toplamaya yardımcı olan sistemlerdir. Mezkur sistemler, akıllı şehirlerdeki araç yörüngelerini belirleme, izleme ve ölçmeyi amaçlamaktadır. Kritik uygulamaları arasında araç hırsızlığı önleme, araç yerelleştirme ve trafik sıkışıklığı çözümü yer almaktadır (Ni, 2016). Bu işlemleri geçekleştirebilecek özellik ve yeterliliğe sahip, sağlıklı bir gerçek zamanlı trafik izleme ve yönetim sistemine ihtiyaç vardır. Bu tür veri toplama ve izleme sistemleri hem yollardan hem de kavşaklardan doğru zaman aralıklarında, titizlikle, doğru verileri toplayarak seçilen bölgeleri gerçek zamanlı veya çevirimdişi olarak izleme imkanı sağlamaktadır (Azimjonov ve ark., 2021).

Kavşaklara kıyasla çeşitli sensörler kullanarak yollardan veri toplamak çok daha kolaydır (Javed ve ark., 2019). Ancak, çok şeritli şehir içi otoyollar ve çok yollu kavşaklardan çevrimiçi veya çevirimdışı trafik verilerini toplamak ve işlemek çok zordur. Ancak, son zamanlarda kullanımı yaygınlaşmakta ve fiyatları giderek uygun hale gelmekte olan, trafik akış sistemlerinin bileşenleri; yüksek başarımlı veri işleme donanımları, geniş görüş açılı görüntü verisi sağlayan kamera sensörleri, görüntü işleme ve yapay zeka alanındaki son iyileşmeler gibi bilgi işlem cihazlarının

geliştirilmesi ve uygun lokasyonlara konuşlandırılması bu alandaki araştırmalara hız kazandırmış durumdadır.

Gerçek zamanlı veri toplama için doğru sensör cihazlarının seçilmesi, çevrimiçi trafik izleme için önemli bir görevdir. Piyasada, Bluetooth, düşük enerjili radyo sensörler, WiFi teknolojileri, GPS/GSM/GPRS, location proofs, Yol Yan Üniteleri (RSU'lar), kamera, RADAR, LIDAR, LOOP, RFID, CCTV ve kızılötesi veya termal kameralar gibi ham trafik verilerini toplamak için çeşitli tipte sensörler bulunmaktadır (Higuchi ve ark., 2015; Khan ve ark., 2014; Lee ve ark., 2014; Rouf ve ark., 2010; Thiagarajan ve ark., 2010; Wu ve ark., 2020; Zhao ve ark., 2019a). Fakat, kamera dışındaki sensörlerin; yüksek kurulum maliyeti, kurulum sırasında trafik kesintisi, bakım ve onarım gibi dört ana dezavantajı vardır (Ahmad ve ark., 2013; Yang ve Duan, 2020; Geetha ve Cicilia, 2017; Guerrero-Ibanez ve ark., 2018; Zhou ve ark., 2017). Dolayısıyla, kamera algılayıcıları hem düşük maliyet bakımından, hem kurulum maliyeti ve esnekliği, bakım ve onarım kolaylığı ve en önemlisi, geniş açılı çok zengin görüntü verilerini sağlaması açısından trafik akış veri toplama ve izleme sistemlerinin olmazsa olmaz bir bileşeni konumundadır. Kamera görüntülerine dayanan trafik akış izleme sistemleri; "görü tabanlı trafik akış veri izleme sistemleri" olarak adlandırılmaktadır. Bu tez çalışmasında da gerçek zamanlı görü tabanlı bir trafik akış izleme sistemi geliştirlimiştir.

(a) (b) (c)

Şekil 2.1. Doğru ve yanlış kamera sensör yerleştirme örnekleri; a) doğru, b) yanlış, c) tamamen yanlış kurulum.

Görü tabanlı trafik akış izleme sistemleri, genel olarak, kamera sensörleri, nesne tespit etme, nesne takip etme ve nesnelerin zamana bağlı izlerini çıkararak izleme gibi bileşenlerden oluşmaktadır. Bu sebeple, görme tabanlı sistemler, doğru kamera algılayıcısının seçilmesi ve seçilen kameraların kavşak ve otoyolların en doğru ve

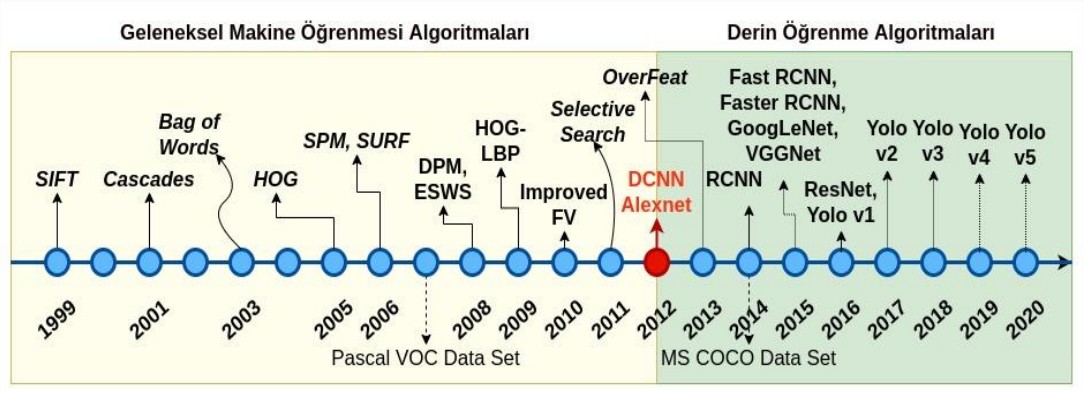
uygun konumuna yerleştirilmesi, bu tür sistemlerin yüksek doğrulukla çalışması bakımından çok önemlidir. Kameraların, genellikle, CCTV veya termal modelleri tercih edilmektedir ve birçok çalışmada bu tür kameraların algılayıcı bakımından yeterli olduğu da bildirilmiştir.

Diğer yandan, kavşak veya otoyollarda temiz veriler toplamak için sensörleri en uygun konumlara yerleştirmek te büyük önem kesbetmektedir. Yanlış sensör (kamera) yerleştirmenin tıkanmaya ve yanılsamaya neden olduğunu bildiren çok sayıda araştırma çalışması vardır (Zhang ve ark., 2011). Yanlış yerleştirilmiş kameralar, kavşaklarda trafik izleme için hayati öneme sahip geniş bir görüş alanı sağlayamaz (Kanhere ve ark., 2005; Lee ve ark., 2014; Thiagarajan ve ark., 2010). Bu çalışmada, yanlış kamera yerleştirmenin etkilerini göstermek için bazı deneyler yapılmıştır, Şekil 2.1. Resimde göründüğü gibi a) durumda doğru yerleştirilmiş kamerayla kavşağın tüm bölgeleri geniş görüş açısıyla çekilmektedir. b) resimde ise kamera kısa yüksekliği yanlış dikey açıyla yerleştirildiği için büyük araçlar küçük araçları kısmi veya tam kapatarak nesne tespit algoritmasının doğruluğunu düşürmüş oluyor. c) resimdeki kavşak yolu drone ile çok yüksekten çekilmiş ve nesne tespit algoritmsı nesneleri algılayamıyor, algıladığı nenseleri de doğru bir biçimde sınıflara ayrıştıramamaktadır. Bu durum bütün sistemin genel doğruluğu olumsuz etkilemekte, hatta sistem tamamen başarısız olmaktadır. Bir sonraki alt bölüm, kameralardan gelen akış verilerini gerçek zamanlı olarak işleyerek nesneleri (araçları) tespit etmek ve istatistiksel bilgiler oluşturmak için onları takip etmektir.

2.3. Nesne Tanıma Algoritma ve Yöntemleri

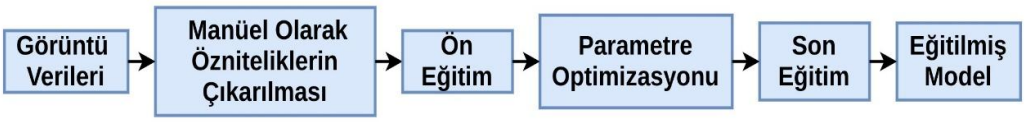
Nesne algılama, hedef nesnelerin türünü tanımak ve bunları bir video karesindeki konumunu belirlemek için kullanılan bir tekniktir. Nesne algılama algoritmaları, genellikle geleneksel makine öğrenimi ve derin öğrenme yöntemleri olarak iki guruba ayrılır, Şekil 2.2. Destek Vektör Makineleri (SVM), K-En Yakın Komşu (KNN), Ölçeği Değişmez Özellik Dönüşümü (SIFT) ve Hızlandırılmış Güçlü Özellikler (SURF) dâhil olmak üzere geleneksel makine öğrenmesi yöntemleri, manuel ve el yapımı öznitelik vektörlerini kullanmaktadır. Dolaysıyla, bu yöntemler,

temsili özelliklerin seçiminde derin bir anlayış ve uzmanlık gerektirmektedir (Chauhan ve Singh, 2018).

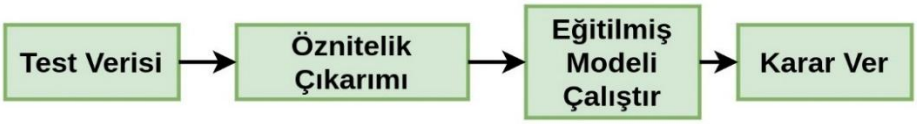


Şekil 2.2. Geleneksel makine öğrenmesi ve derin öğrenme yaklaşımlarına dayalı nesne tespit algoritmaları.

Öte yandan, derin öğrenme yaklaşımları herhangi bir uzmanlık veya bir görüntüdeki nesnelerin içeriklerinin derinlemesine anlaşılmasını gerektirmez. Çünkü bu tür metotlar derin sinir ağlarını kullanarak gizli ve derin özellikleri otomatik olarak çıkarırlar. Derin sinir ağları, doğrusal olmayan aktivasyon işlevleri içeren onlarca gizli katman kullanır, bu nedenle yapay sinir ağları yaklaşım ve teknikleri, orijinal görüntülerden özellik vektörini çıkarma ve karar vermeyi öğrenmede geleneksel makine öğrenmesi algoritmalarına göre daha iyi konumdadır (Zhao ve ark., 2019b). Aşağıda makine öğrenmesi ve derin öğrenme metotları detaylı olarak incelenmiştir.



Şekil 2.3. Geleneksel makine öğrenmesi algoritmalarının eğitim süreci.



Şekil 2.4. Geleneksel makine öğrenmesi algoritmalarının test süreci.

Makine öğrenmesi, Arthur Samuel'e (1959) göre "bilgisayarların açıkça programlanmadan öğrenme yeteneği" olarak tanımlanan bir bilgisayar görüşü alanıdır. Yapay zeka alanının hesaplamalı öğrenme teorisi, ham verilerden

öğrenebilen, sistemi eğitebilen ve bu eğitim verilerine dayanarak tahminlerde bulunabilen algoritmaların analizini ve oluşturulmasını araştıran makine öğrenmesi alt alanının geliştirilmesine olanak sağlamıştır (Kohavi ve Provost, 1998). Geleneksel makine öğrenimi algoritmaları, Şekil 2.3.'te gösterildiği gibi gözetimli bir model geliştirmek için eğitim setiyle sistemin öğrenmesine dayanan otomatik öğrenme yaklaşımlarıdır. Bu önceden eğitilmiş model, Şekil 2.4.'te gösterilen test veri kümesini sınıflandırmak veya tanımak için kullanılır. Denetimli ve denetimsiz öğrenme için çeşitli geleneksel makine öğrenme algoritmalarının sınıflandırılması Şekil 2.5.'te gösterilmiştir.

Şekil 2.5. Geleneksel makine öğrenmesi algoritmaları.

Destek Vektör Makineleri (SVM). SVM'ler en etkili sınıflandırma yöntemi olduğu iddia edilen nesne ayrıştırma algoritmalarıdır (Cortes ve Vapnik, 1995). SVM, ayırıcı bir hiper düzlem aracılığıyla iyi tanımlanmış bir ayırt edici sınıflandırıcı türüdür. Başlangıçta, SVM modeli, yalnızca ikili sınıflandırma işlemleri için kullanılmıştır, ancak daha sonra SVM algoritmasının türevleri çok sınıflı ayrıştırma problemlerinde de yaygın olarak kullanılmıştır (Crammer ve Singer, 2002; Lee ve ark., 2004). Girdi verileri sınıflara daha doğru bir biçimde ayrıştırabilmek amaçlı bazı ek kısıtlamalar ve parametreler eklenmiştir. Sistemin performansı özelliklerin seçimine bağlı olduğundan, optimize edilmiş özellik seçimi için bir karma model PSO-SVM'ler gibi

birçok farklı SVM tabanlı alt sınıflandırma yöntemleri de geliştirilmiştir (Lin ve ark., 2008).

Ayırım analizi, iki veya daha fazla nesnenin, o nesneyi tanımlamak için kullanılan ölçülmüş özelliklere göre gruplandığı veya kümelendiği nesnelerin sınıflandırılmasına dayanmaktadır. Doğrusal diskriminatif analiz (LDA) genel olarak boyut azaltma için kullanılırken, ikinci dereceden ayırıcı analiz (QDA) sınıfların ayrılması için ikinci dereceden yüzey kullanır (Sotiris, 2007).

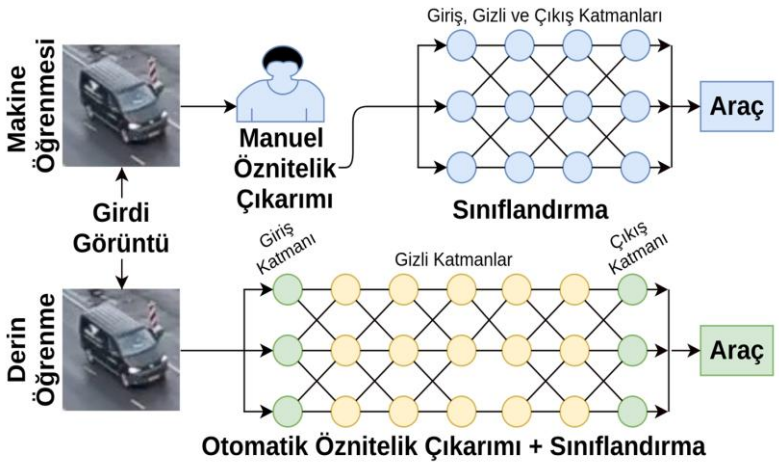
Naive Bayes sınıflandırıcıları, Naive Bayes bağımsızlık modeli (Kuncheva, 2006) tarafından verilen, yalnızca bir gözlemlenmemiş (ana) düğüm ve aralarında bağımsızlık varsayımına sahip birkaç gözlemlenmiş (çocuk) düğüm içeren yönlendirilmiş döngüsel olmayan grafiklerin kullanıldığı Bayes ağlarıdır. Naive Bayes'i basitleştiren SVM ve Sinir Ağlarında gerektiği gibi serbest parametreleri ayarlamaya gerek yoktur, çünkü Naive Bayes'de kullanılan akıllı olasılık fonksiyonları, ayrıştırma uygulanmalarının sınıflandırma özelliklerine katkı sağladığı söylenmektedir (Islam ve ark., 2007).

Karar Ağacı yaklaşımı, tüm veri kümesinin etiketlendiği başka bir denetimli öğrenme algoritmasıdır. Karar ağaçları, sınıfları parametre değerlerine göre sıralayarak sınıflandırma işlemleri için kullanılır. ID3, C4.5, CHAID ve CART, karar ağacına ait algoritmalardan bazılarıdır. Bu yaklaşımın en büyük avantajı, hem sayısal hem de kategorik nitelikleri ele alabilmesidir. Bu yöntem küçük veri kümeleri için uygundur ancak büyük veri kümeleri için gecikmeye neden olur. Ancak karar ağacının oluşturulması, hesaplama için gereken süre daha az olmasına rağmen önemli bir zaman gerektirir (Rokach ve Maimon, 2005).

K-Means Kümeleme, test veri kümesinin etiketlenmediği denetimsiz öğrenme yaklaşımının bir örneğidir. Hiyerarşik kümeleme, iki tür kümeleme tekniği kullanan hiyerarşi oluşturmaya dayanır: Toplayıcı ve Bölünen. Aglomerative Clustering, aşağıdan yukarıya şekilde büyük bir küme oluşturmak için eşleştirilir. Bölücü Kümeleme, onları yukarıdan aşağı bir şekilde küçük kümelere ayıran büyük bir

kümenin kırılmasını kullanır. Bölümleme kümeleme, veri kümelerinin, her birinin küme ortalaması şeklinde karakterize edildiği eşit veya eşit olmayan kümelere bölünmesini kullanan bir tekniktir. K-ortalamalarında kümelemede veri kümesi, her birinin kendi küme ortalaması aracılığıyla temsil edildiği K-küçük kümeler gurubuna ayrılmaktadır (Kanungo ve ark., 2002). *Temel Bileşen Analizi* PCA, yüksek boyutlu verileri düşük boyutlu verilere dönüştürerek boyut azaltma için kullanılır. Öğrenme algoritmalarının m girişi, m çıkışı ve n gizli katmanı varsa, m>n. Bununla birlikte, PCA'nın kapasitesi yalnızca bir alanın (m) başka bir alana (n) doğrusal dönüşümü ile sınırlıdır (Andrecut, 2009).

Yapay Sinir Ağı (YSA) olarak da adlandırılan sinir ağı, nöronların biyolojik teorisinden esinlenmektedir. Girdi, gizli ve çıktı katmanlarından oluşur. Gizli katmanlar, alınan girdiyi işlemek ve bunları çıktı katmanlarına göndermek için kullanılır. Sinir ağının temel mimarisi Şekil 2.6.'da gösterilmektedir. NN yaklaşımı, önceden var olan yöntemlerden daha fazla doğruluk ve verimlilik sağlar. NN'nin çeşitli yaklaşımları geri yayılma algoritması (BPNN), Radyal temel fonksiyonu (RBF), Tamamlayıcı NN (CMPNN) ve Olasılıksal yaklaşımdır. BPNN en yaygın kullanılan NN yaklaşımıdır, ancak büyük veri seti kullanıldığında eğitim süreci diğerlerine nazaran daha yavaştır (Sotiris, 2007).



Şekil 2.6. Makine öğrenmesi ve derin öğrenme yaklaşımlarının akış diagramı.

Yukarıda kısaca tanımları verilen ve diğer geleneksel makine öğrenimi yöntemlerini kullanan nesne algılama ve sınıflandırma yaklaşımları, tamamen manuel ve el yapımı nesne özniteliklerine dayanmaktadır. Öznitelik vektörü, bir görüntüdeki piksellerin görünümünden veya hareketinden çıkarılan sayısal veriler veya yapılardır (Sivaraman ve Trivedi, 2013). Araçları, bu tür manuel, görünüm ve şekil özelliklerine göre tanıyan algılama algoritmaları, esasen üç aşamalı bir yol izler. İlk olarak, arka planı kaldırarak arka plan modelleme veya arka plan çıkarma işlemi gerçekleştirilir. İkinci aşamada, arka plan kaldırıldıktan sonra kalan lekeler ve onların konumları tespit edilir. Daha sonra, istenen (kullanıcı tanımlı) öznitelikler; algılanan leke ve bu lekelerin konumları baz alınarak çıkarılır (Fernandez-Sanjurjo ve ark., 2019; Mandellos ve ark., 2011). Üçüncü aşamada, tespit edilen lekeler ve çıkarılan özellikler, araç türlerini belirlemek için sınıflandırma algoritmalarına beslenir. Bu algoritmalar, sabit arka plana ve iyi hava koşullarına sahip çevrimdışı videolar için iyi çalışmaktadır, ama, hızlı arka plan değişikliklerine, farklı hava koşullarına, renk ve şekil sorunlarına karşı çok zayıftırlar (El Mokaddem ve ark., 2019).

Dinamik arkaplan modellemesi (Zhu ve ark., 2006), optik akış (Franke ve ark., 2005), doluluk ızgarası (Badino ve ark., 2012) ve izleme stikselleri (Erbs ve ark., 2011) gibi hareket tabanlı algoritmalar, geçmişte; görünüm, şekil ve uyumsuzluk (disparity) tabanlı algoritmaların karşı karşıya olduğu, yukarıda sıralanan sorunları ortadan kaldırmak amaçlı geliştirilmiştir (Sivaraman ve Trivedi, 2013). Harekete dayalı algılama, hareketli lekelere veya piksellere odaklanır ve farklı hava koşullarına karşı dayanıklıdır. Lakin, çeşitli kısmi veya tam tıkanmalar, yavaşlık, bulanık görünümler, parlaklık ve arka plan değişiklikleri gibi sorunlar nedeniyle, her tür aracı mükemmel bir şekilde tanımlamak için sağlam bir özellik tanımlayıcısını manuel olarak tasarlamak zordur (Zhao ve ark., 2019b). Ek olarak, geleneksel makine öğrenmesi algoritmalarının sınıflandırma kabilyeti daha yüksek olmasına rağmen nesneleri bir video karesinden veya resimden tespit etme zamanı çok yavaştır. Bu nedenle, bu tür sıkıntıları giderebilmek için makine öğrenmesinin yeni bir alt tekniği olan derin öğrenme yöntemleri geliştirilmiştir. Sıradaki parağraflarda derin öğrenme metot ve yaklaşımları detaylı olarak incelenmiştir.

$$y = f(X) = \sum_{i=0}^{n} w_i X_i^{(input)} + b, \quad (2.1)$$

Burada, y - eğitim fonksiyonu, w_i – ağırlık vektörü, $X_i^{(input)}$– girdi veri, b – bias.

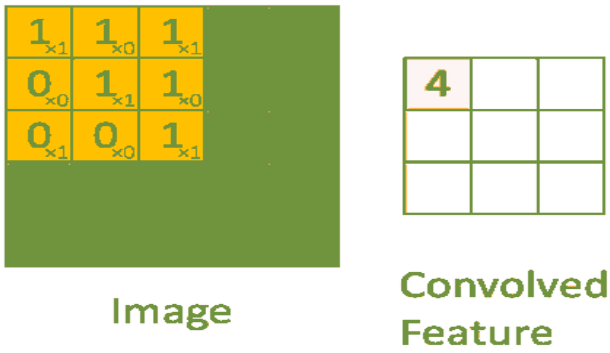
Derin Öğrenme; derin modeller (Denklem 2.1), derin ve gizli yapı ve mimarilere sahip sinir ağları olarak adlandırılmaktadır, Şekil 2.6. (alt görüntü). Bu tür derin sinir ağlarının tarihi 1940'lara kadar uzanabilmektedir (Pitts ve McCulloch, 1947) ve asıl amaç, genel öğrenme problemlerini basit bir şekilde çözmek için insan beyninin çalışma prensiplerini temel alarak benzetmeye çalışmak veya simüle etmektir.

Derin sinir ağları alanı, Rumelhart ve arkadaşlarının geri yayılım algoritmasının önerisiyle 1980'lerde ve 1990'larda yaygınlaşmaya başlamıştır (Rumelhart ve ark., 1986). Ancak, bir derin öğrenme sisteminin eğitilme esnasındaki verilerden ezberleme (overfitting), büyük ölçekli eğitim verilerinin mevcut olmaması veya yetersiz, sınırlı miktardaki eğitim kümeleri, yüksek başarımlı paralel hesaplama yapan cihazların olmayışı veya bu tür bilgisayarların sınırlı hesaplama gücüne sahip olması, ve buna ek olarak diğer makine öğrenimi araçlarına kıyasla derin öğrenme sistemlerinin performans ve doğruluk değerlerinin düşük olması nedeniyle, derin (yapay) sinir ağları 2000'li yıllarının başına dek yaygın kullanılamamıştır.

Fakat, 2006 yılından itibaren, ses işleme, görüntü işleme ve yapay zeka alanlarındaki önemli gelişmeler, bunun yanı sıra, binlerce çekirdekten oluşan, paralel hesaplama yapabilen yüksek başarımlı yeni nesil grafiksel işlem birimlerinin (GPUs) üretilmesi, GPU'lerin fiyatlarındaki ucuzlamalar, bununla birlikte, büyük miktardaki etiketlenmiş örüntüler içeren sağlam veri kümelerinin oluşturulması sebebiyle, derin sinir ağları, bilimsel araştırmacıların dikkatini tekrar kendi üzerine çekmiş ve birçok araştırmacı bu alan üzerinde yenilikçi yaklaşımlar yapmıştır, ve halihazırda da geliştirmeler devam etmektedir. Netice olarak, derin sinir ağlarına dayanan sağlam derin öğrenme mimari ve algoritmları geliştirilmiş olup, bu derin öğrenme yaklaşımları; ses tanıma, doğal dil işleme, görüntü işleme, yüz tanıma, süpheli hareketleri tespit etme, nesne tanıma, nesne tespiti veya nesne takibi gibi birçok

gerçek dünya probleminin yüksek performans ve başarıyla çözümlenmesine olanak tanımaktadır (Zhao ve ark., 2019b).

Derin öğrenme yaklaşımı; giriş, gizili ara katmanlar, ve onların içerdiği yapay sinir düğümleri, güçlü aktivasyon fonksiyonlar, ve son olarak çıkış katmanlarından oluşmaktadır. Bu yaklaşımlar, girdi olarak verilen etiketlenmiş eğitim verilerinden derin ve gizili öznitelikleri otomatik olarak çıkarabilmektedir, ve bu sayede, ağırlık ve bais vektörlerini hesaplayarak, verileri genellemekte ve eğitilmiş hazır ağırlık modelleri oluşturmaktadır. Derin öğrenmemeye dayalı yaklaşımlar, genel olarak, derin sinir ağı (DNN), yapay sinir ağı (ANN), tekrarlayan yapay sinir ağları (RNN), uzun kısa vadeli hafıza ağları (LSTM) ve evrişimsel sinir ağları (CNN) gibi ana mimarilerden oluşmaktadır. DNN, ANN, RNN, LSTM mimarileri çoğunlukla sayısal, metinsel veya ardışık verilerden oluşan veri türleri için tercih edilirken, CNN görüntü veya video-akımlarından oluşan veriler için daha etkin biçimde kullanılmaktadır. Bu çalışmanın konusu video görüntü işleme olduğundan ötürü, aşağıda, CNN tabanlı nesne tespit algoritmlarından ayrıntılı olarak bahsedilecektir (Zhao ve ark., 2019b).



Şekil 2.7. CNN mimarisinin çalışma prensibi (Krizhevsky ve ark., 2017).

CNN; genel olarak, görüntü işleme görevini gerçekleştirme amaçlı geliştirilmiş olup, derin öğrenme alt mimari yaklaşımlarının en temsili modelidir (LeCun ve ark., 2015). CNN; giriş, gizili ve çıkış katmanlarını içeren üç ana adımdan ibarettir. Şekil 2.2.'de, derin öğrenme yaklaşımları başlığı altında bu alt mimarilere birçok örnek verilmiştir. Her bir CNN katmanı, öznitelik haritası olarak bilinmektedir. Giriş katmanının özellik haritası, farklı renk kanalları (ör. RGB) için bir 3B'lu piksel yoğunlukları matrisidir. Herhangi bir dahili katmanın özellik haritası, pikseli belirli

bir özellik olarak görülebilen, böyütü indirgenmiş çok kanallı bir görüntüdür. Her nöron (sinir düğümü), önceki katmandan (alıcı alandan) komşu nöronların küçük bir kısmı ile bağlantılıdır. Filtreleme ve havuzlama gibi özellik haritalarında farklı dönüşüm türleri yürütülebilmektedir. (Krizhevsky ve ark., 2017; Oquab ve ark., 2015; Oquab ve ark., 2014). Filtreleme (evrişim) işlemi, Şekil 2.7., bir filtre matrisini (öğrenilmiş ağırlıkları) alıcı bir nöron alanı değerleriyle sarar ve son çıktıları elde etmek için doğrusal olmayan Sigmoid (Denklem 2.2), ReLU (Denklem 2.3), Tanh (Denklem 2.4) gibi aktivasyon fonksiyonlarını kullanır (Wadley, 1952). Maksimum havuzlama, ortalama havuzlama, L2-havuzlama ve yerel kontrast normalleştirme (Kavukcuoglu ve ark., 2009) gibi havuzlama işlemleri, daha sağlam özellik (öznitelik) tanımları üretmek için alıcı bir alanın çıktılarını tek bir değere indirger. CNN, görüntü, nesne veya leke özniteliğini tek bir skalar veya vektöre indirgeyerek görüntü işleme süresini önemli seviyede düşmesine, algortimanın hızlanmasına, ve bu işlemleri yüksek doğrulukla yapılabilmesine imkan sağlamaktadır. Son on yıllıkta, görüntü işleme, örüntü tanıma, nesne tespiti ve sınıflandırma işlemleri için, CNN tabanlı birçok alt mimariler geliştirilmiştir. Mesala, AlexNet (Krizhevsky ve ark., 2017), Overfeat (Sermanet ve ark., 2014), GoogleNet (Szegedy ve ark., 2015), Visual Geometry Group (VGGNet) (Simonyan ve Zisserman, 2015), Region-based CNN (R-CNN) (Girshick ve ark., 2014), Fast (FR-CNN) (Girshick, 2015), Faster (R-CNN) (Ren ve ark., 2017), Single Shot Detector (SSD) (Liu ve ark., 2016), Residual Nets (ResNets) (He ve ark., 2016), ve YOLO sürüm (1,2,3,4,5) (Redmon ve Farhadi, 2016; Redmon ve Farhadi, 2017; Redmon ve ark., 2018; Bochkovskiy ve ark., 2020; Jocher ve ark., 2021) alt mimarileri, CNN mimarisinden istifade edilerek geliştirilmiş, en yaygın kullanılan ve en başarılı alt mimariler olarak bilinmektedir. Aşağıdaki parağraflarda, bu alt mimarilerden ayrıntılı bir biçimde bahis edilmiştir.

$$f(x) = Sigmoid(x) = \frac{e^x}{e^x+1}, \tag{2.2}$$

$$f(x) = ReLu(x) = max(0, x), \tag{2.3}$$

$$f(x) = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}, \tag{2.4}$$

AlexNet, 60 milyon parametre ve 650.000 nörondan ibaret bir yapay sinir ağıdır. Alexnet, beş evrişimli katmandan oluşur, bunlardan bazılarını maksimum havuz katmanları izler ve son 1000-hücreli softmax (regresyonu) ile tam bağlantılı üç katmandan oluşur. Eğitimi hızlandırmak için, doyurucu olmayan nöronlar ve evrişim işleminin çok verimli bir GPU uygulaması kullanılmıştır. Tam bağlantılı katmanlarda aşırı uyumu (veri ezberlemeyi, overfitting) azaltmak için, yakın zamanda geliştirilmiş, çok etkili olduğu kanıtlanan "bırakma" adı verilen (dropout) bir düzenleme yöntemi kullanılmıştır. ILSVRC-2012 yarışmasında da bu modelin bir varyantına giriş ve ikinci en iyi girişte elde edilen %26,2'lik hata oranına kıyasla %15,3'lük ilk 5 test hata oranı elde edilmiştir.

OverFeat nesne tespiti (sınıflandırması ve konuşlandırması) metodünün ana gayesi, görüntülerdeki nesneleri eşzamanlı olarak sınıflandırmak, konumlandırmak ve tespit etmek için evrişimli bir ağın eğitilerek yeni bir model meydana getirmektir. Bu eğitilmiş CNN alt mimari modeli tüm görevlerin; sınıflandırma, algılama ve konuşlandırma doğruluğunu arttırmıştır. Bu nedenle, çalışma, tek bir ConvNet ile nesne algılama, tanıma ve konumlandırma için yeni bir entegre yaklaşım önermektedir. Başka bir deyişle, bu çalışmada, tahmin edilen sınırlayıcı kutuları biriktirerek konuşlandırma ve tespit için yeni bir yöntem sunulmuştur. Ancak, görüntü işlemnin bu üç görevi her biri yüksek başarımlı işlem gücü gerektiren süreçlerdir, dolaysıyla üç görevin aynı ağ içine yerleştirilmesi ve girdi görüntüde binlerce bölgeyi seçerek her bir seçilen bölgenin tekrar tekrar işlemden geçirilmesi bu yaklaşımın icra süresini çok yükseltmiş, performansını büyük ölçüde düşürerek olumsuz yönde etkilemiştir. Onun dışında, beklenilen yüksek doğruluk elde edilememiştir. Bu metot, 2014 yılının en iyi algoritması olarak kabul edilmiş olsa da, SSD ve YOLOv3 algoritmalarının gün yüzünü görmesiyle bu yaklaşım nesne tespit etme işlemleriyle uğraşan araştırmacıların gündeminden düşmüş, ve artık kullanılmamakatadır.

Başlangıç Ağlar (the inception networks), 2014 yılının, özellikle CNN'ler için sinir ağları alanındaki en büyük atılımlardan biri olarak bilinmektedir. Şimdiye kadar Inception Networks'ün Inception Version 1, 2 ve 3 olarak adlandırılan üç versiyonu

bulunmaktadır. İlk versiyon, görü tabanlı sistemler alanına 2014 yılında girmiştir ve “GoogleNet” adından da anlaşılacağı gibi Google’daki bir ekip tarafından geliştirilmiştir. Bu ağ, ILSVRC'de sınıflandırma ve nesne tespiti için yeni bir son teknoloji ürünü belirlemekten sorumluydu. Başlangıç ağının bu ilk sürümü GoogleNet olarak adlandırılır. Bir ağ birçok derin katmanla kurulursa, aşırı uyum sorunu ile karşı karşıya kalabilir. Bu sorunu çözmek için, konvolüsyonlarla daha derine inen araştırma makalesinde yazarlar, aynı seviyede çalışabilen birden fazla boyuta sahip filtrelere sahip olma fikriyle GoogleNet mimarisini önermiş, ama bu fikirle, ağ aslında derinleşmek yerine genişlemiş, netice itibariyle bu ağ çok yavaş çalışmaktadır, hesaplama maliyeti çok yüksektir. Onun dışında, bu ağlar sadece sınıflandırma yapabilmektedir, girdi görüntüde birden fazla nesne bulunduğu hallerde her bir nesneyi ayrı ayrı konumunu tahmin ederek tespit etme işlevi bulunmamaktadır.

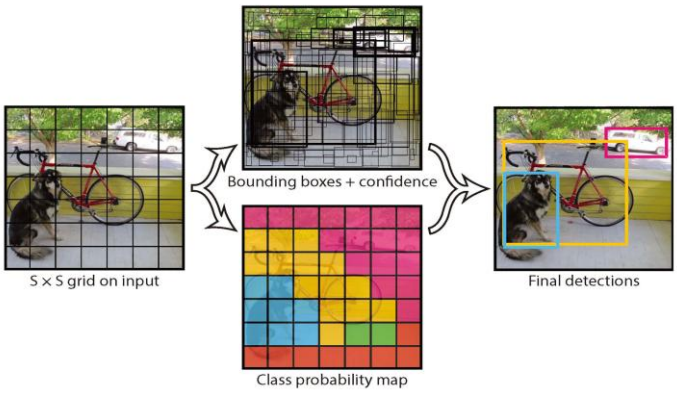
VGG16, çok hiperparametre kullanmadığı için evrişimli sinir ağlarının daha basit, daha hafif bir alt mimarisi olarak bilinir. Her zaman evrişim katmanında 1 adımda 3x3 filtre kullanır ve 2x2 katmanlarını 2 adımda havuzlarken SAME dolgusunu kullanır. Model, 1000 nesne sınıfına ait 14 milyondan fazla etiketlenmiş görüntülerden oluşan ImageNet veri kümesi ile eğitilerek %92.7'lik bir ilk-5 test doğruluğu elde etmiştir. ILSVRC-2014'e sunulan ünlü modellerden biridir. Boyutu büyük olan filtreleri sırasıyla birinci ve ikinci evrişimli katmanda 11 ve 5 çekirdekli filtreleri 3x3 daha küçük çekirdekli filtreler ile değiştirerek AlexNet'e göre iyileştirme yapmıştır. VGG16, NVIDIA Titan Black GPU'ları kullanılarak haftalarca eğitilen bir modeldir, bu model de AlexNet’te olduğu gibi girdi görüntüyü bir bütün olarak işlemden geçiren ve sınıflandırmaya yarayan bir alt mimaridir. Bu yaklaşım da nesne konumunu bulma ve nesneyi tespit etme fonksiyonları içermemektedir.

RCNN, bölge tabanlı evrişimli sinir ağı yaklaşımı, girdi görüntü veya video karesinde mevcut olan nesnelerin konumunu tespit etme ve sınıflandırma algoritmasıdır. Bu algoritma girdi görüntüden önemli olduğu tahmin edilen 2000 bölgeyi rastgele olarak seçmekte ve bu bölgelerde nesne(ler) olup olmadığını kontrol ediyor. Yani, R-CNN'nin asıl amacı bir girdi görüntüsünü değerlendirip, çıktı olarak

bir dizi sınırlayıcı kutu üretmektir. Bu yaklaşımın ürettiği her sınırlayıcı kutu, bir nesne ve ayrıca nesnenin kategorisini (örneğin araba veya yaya) içerir. Daha yakın zamanlarda, R-CNN diğer bilgisayarla görme görevlerini gerçekleştirmek için genişletilmiştir. Aşağıdakiler, R-CNN'nin geliştirilmiş olan bazı versiyonlarını kapsamaktadır: Kasım 2013: R-CNN. Bir girdi görüntü verildiğinde, R-CNN, tespit ve sınıflandırma işlemini, her ROI'nin görüntüdeki bir nesnenin sınırını temsil eden bir dikdörtgen olduğu ilgili bölgeleri (ROI) çıkarmak için Seçmeli Arama (Selective Search) adlı bir mekanizma uygulayarak başlar. Senaryoya bağlı olarak, iki bin kadar ROI olabilir. Bundan sonra, çıktı özelliklerini üretmek için her ROI bir sinir ağına beslenir. Her ROI'nin çıktı özellikleri için, ROI'de ne tür bir nesnenin (varsa) bulunduğunu belirlemek için bir destek vektörü makine sınıflandırıcıları koleksiyonu kullanılır. Nisan 2015: Fast R-CNN. Orijinal R-CNN, iki bin kadar ilgili bölgenin her birinde sinir ağı özelliklerini bağımsız olarak hesaplarken, Hızlı R-CNN sinir ağını tüm görüntü üzerinde sadece bir kez çalıştırır. Ağın sonunda ROIPooling adı verilen, ağın çıkış tensöründen her ROI'yi bölen, onu yeniden şekillendiren ve sınıflandıran yeni bir yöntem vardır. Orijinal R-CNN'de olduğu gibi, Hızlı R-CNN, istenilen bölgeleri belirleyebilmek için Seçmeli Arama'yı kullanır. Haziran 2015: Fatser R-CNN. Hızlı R-CNN, ROI'leri belirlemek amaçlı Seçmeli Arama'yı kullanırken, Faster R-CNN, ROI üretimini sinir ağının kendisine entegre eder. Bu işlemle de daha hızlı nesne tespiti ve sınıflandırması yaptıklarını söylemişler. Mart 2017: Mask R-CNN. R-CNN'nin önceki sürümleri nesne algılamaya odaklanırken, Mask R-CNN örnek segmentasyonu belirlemeye odaklanmaktadır. Maske R-CNN'de ayrıca ROIPooling'i, bir pikselin kesirlerini temsil edebilen ROIAlign adlı yeni bir yöntemle değiştirilmiştir. Haziran 2019: Mesh R-CNN, Mesh RCNN algoritmasına, hızlandırabilmek ve daha gerçekçi nesne tespiti ve sınıflandırması yapabilmek amaçlı, bir 2D görüntüden 3D ağ oluşturma özelliği eklenmiştir. R-CNN bazlı metotların tespit doğruluğu yüksek olmasına rağmen, performansı çok düşüktür, en fazla 3-5 FPS görüntü işleme performansı elde edilebilmektedir. Bu nedenle, R-CNN ve türevleri gerçek zamanlı nesne algılama görevlerinde kullanılamaz haldedir.

SSD, tek bir derin sinir ağı kullanarak görüntülerdeki nesneleri tespit etmek (nesne konumlandırma ve sınıflandırmak) için geliştirilmiş bir yöntemdir. Tek Atış

Dedektörü (SSD) olarak adlandırılan bu yaklaşım, sınırlayıcı kutuların çıktı alanını, özellik harita konumu başına denk gelen farklı en-boy oranları ve ölçekler üzerinden varsayılan kutuların kümesine dönüştürmektedir. Tahmin zamanında ağ, her bir varsayılan kutuda her bir nesne kategorisinin varlığı için puanlar üretir ve sınırlayıcı kutuların nesne şekline daha iyi uyması için ayarlamalar yapar. Ek olarak, bu sinir ağı, çeşitli boyutlardaki nesneleri doğal olarak işlemek için farklı çözünürlüklere sahip birden çok özellik haritasından gelen tahminleri birleştirir. SSD modeli, istenilen bölge oluşturmayı amaçladığı, sonraki piksel veya özniteliği yeniden örnekleme aşamasını tamamen ortadan kaldırdığı ve tüm hesaplamaları tek bir ağda kapsadığı için basitçe nesne önerileri gerektiren yöntemlerle ilişkilidir. Yani SSD metodu bölge tabanlı bir nesne algılama ve tanıma algoritmasıdır. Bu durum, SSD'nin eğitilmesini ve bir algılama bileşeni gerektiren sistemlere entegre edilmesini kolaylaştırır. PASCAL VOC, MS COCO ve ILSVRC veri kümelerindeki deneysel sonuçlar, SSD'nin ek bir nesne önerme adımı kullanan yöntemlerle karşılaştırılabilir doğruluğa sahip olduğunu ve hem eğitim hem de çıkarım için birleşik bir çerçeve sağlarken çok daha hızlı olduğunu doğrulamaktadır. Diğer tek aşamalı yöntemlerle karşılaştırıldığında SSD, daha küçük boyutlu girdi görüntülerden bile çok daha iyi nesne tespit ve sınıflandırıma doğruluğu elde edebilmektedir. SSD, 300 × 300 giriş için bir Nvidia Titan X üzerinde 58 FPS'de VOC2007 testinde %72,1 mAP elde ettiği ve 500 × 500 giriş için SSD, %75,1 mAP elde ederek bir son teknoloji olarak, Daha Hızlı R-CNN modelinden daha iyi performans gösterdiği söylenmiştir. Ancak, gerçek dünya problemleri için, daha az belleğe sahip; mesala, 4, 6, 8 veya 11GB bellek hafızalı grafiksel işlemcilerle denendiği zaman azami 15 fps elde edilebilmiştir.

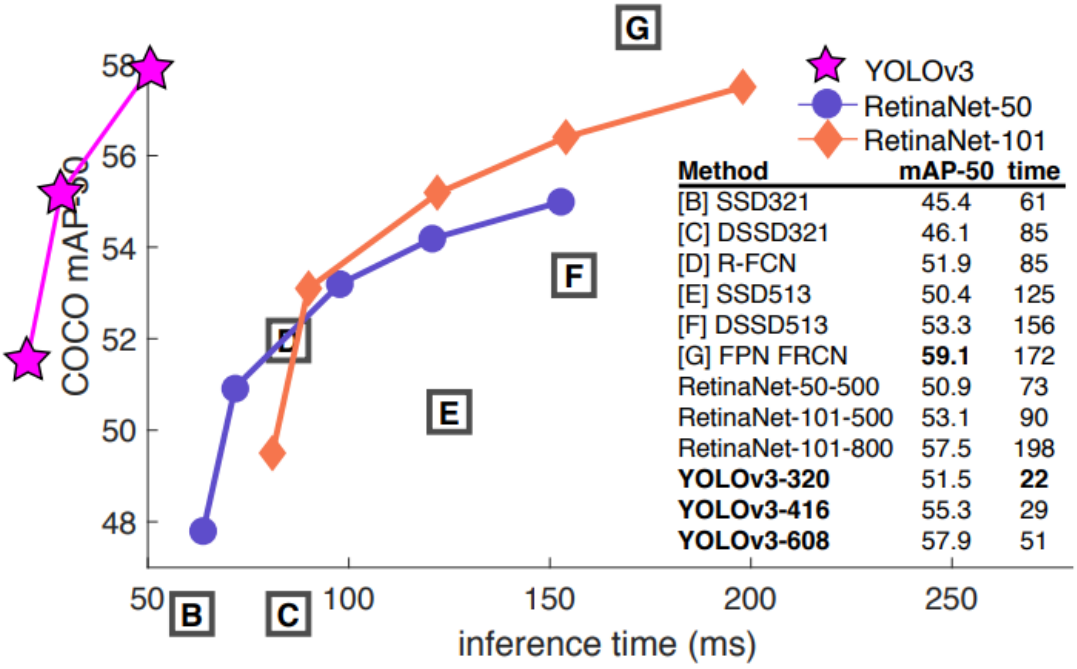


Şekil 2.8. YOLOv3 algoritmasının genel çalışma prensibi (Redmon ve ark., 2018).

YOLOv3, Redmon ve ark. tarafından geliştirilmiş, gerçek zamanlı nesne tespiti ve sınıflandırma yapma kabilyetine sahip olan bir yaklaşımdır. Bu model, birden çok kategori ve sınırlayıcı kutular için güven değerlerini tahmin etmek amaçlı en üst ve önemli özellik haritasının tamamını kullanan yeni bir yaklaşımdır. YOLOv3 algoritmasını altında yatan temel fikr Şekil 2.8.'de gösterilmektedir. YOLOv3, giriş görüntüsünü bir S × S ızgaraya böler ve her bir ızgara hücresi, o ızgara hücresinde ortalanmış nesneyi tahmin etmekten sorumludur. Her ızgara hücresi, B tane sınırlayıcı kutuyu ve bunlara karşılık gelen güven puanlarını tahmin eder. Resmi olarak, güven puanları "Tahmin_Fonksiyonu(Nesne)∗ $IOU_{tahmin}^{doğruluk}$" olarak tanımlanır; bu, nesnelerin (Tahmin_Fonksiyonu(Nesne) ≥ 0) var olma olasılığını gösterir ve öngörü güvenini temsil eder ($IOU_{tahmin}^{doğruluk}$). Aynı zamanda, kutu sayısına bakılmaksızın, “Sınıf” koşullu sınıf olasılıkları (Tahmin_Fonksiyonu (Sınıf$_i$ | Nesne)) da her bir grid hücresinde tahmin edilmektedir. Sadece bir nesneyi içeren ızgara hücresinden gelen katkı hesaplanmaktadır. YOLOv3 algoritmasının beş versiyonu mevcuttur. İlk üçü, C diliyle geliştirilmiş Darknet CNN tabanlı çerçeveyle, kalanları ise Python programla dilinde, Tensorflow ve PyTorch tabanlı çerçeveler ile geliştirilmiştir.

Yukarıda sıralanan derine öğrenme tabanlı CNN mimarisinin alt mimarileri içinde; AlexNet, GoogleNet, VGG gibi alt algoritmalar sadece nesne sınıflandırma veya nesne tanımada kullanılabilmektedir. Onun dışında, bu yaklaşımlar, nesnelerin görüntü veya video karelerindeki konumlarını tespit etme işlevini yapamamaktadır. Yani, bu modeller girdi olarak verilen görüntü veya video karesini bir bütün olarak değerlendirmekte ve tasvirin sınfını bulmaya yardımcı olmaktadır. Böyle bir durumun söz konusu olması, bu algoritmaları nesne tespiti işlemleri için kullanışsız hale getirmiş durumdadır. Ancak, R-CNN, FR-CNN, Faster R-CNN, SSD ve YOLOv3’nun tüm sürümleri girdi olarak verilen görüntüyü parçalara bölerek işliyor, nesnelerin video karesindeki veya görüntüdeki yerini yüksek doğrulukla tespit edebilmekte, ama genel anlamda nesnelerin sınıfını belirlemede biraz düşük performans sergilemektedir.

Bununla birlikte, bu mimarilerden YOLOv3 algoritması dışındaki tüm yaklaşımlar, gerçek zamanlı nesne tespitinde ve nesne algılamanın türevi olan yüz tanıması, şüpheli hareketleri tespit etme, nesne segementasyonu veya akıllı trafik akış izleme ve yönetme sistemlerinde yetersiz kaldığı görülmektedir. Şekil 2.9. ile derin öğrenme tabanlı geliştirilen sistemlerin performans analizi gösterilmiştir. Ancak, gerçek zamanlı trafik akış izleme veya veri toplama sistemleri için, bir bölge seçilip, seçilen bölgeden kameralarla elde edilen görüntülerdeki nesneler iyi bir şekilde etiketlenerek YOLOv3 algoritması tekrar eğitildiğinde yüksek performans ve doğruluklu araç algılama ve ayrıştırma sistemleri gerçekleştirilebilmektedir. Bu çalışma için birçok kavşak ve yoldan toplanan görüntüler etiketlenerek, YOLOv3 algoritması tekrar eğitilmiş ve bu sayede gerçek zamanlı, sağlam, yüksek başarımlı ve doğruluklu araç tespiti ve sınıflandırması yapabilen bir YOLOv3 tabanlı araç algılama modeli oluşturulmuştur. Sonra bu model video görüntüleri için kullanılmış, ürettiği sınırlayıcı kutu, nesne sınıfı ve güven değerlerinden ibaret olan çıktılar; kavşak ve otoyollardan gerçek zamanlı trafik akış toplama sisteminin nesne (araç) takibi modülünün girdisi olarak kullanılmıştır.

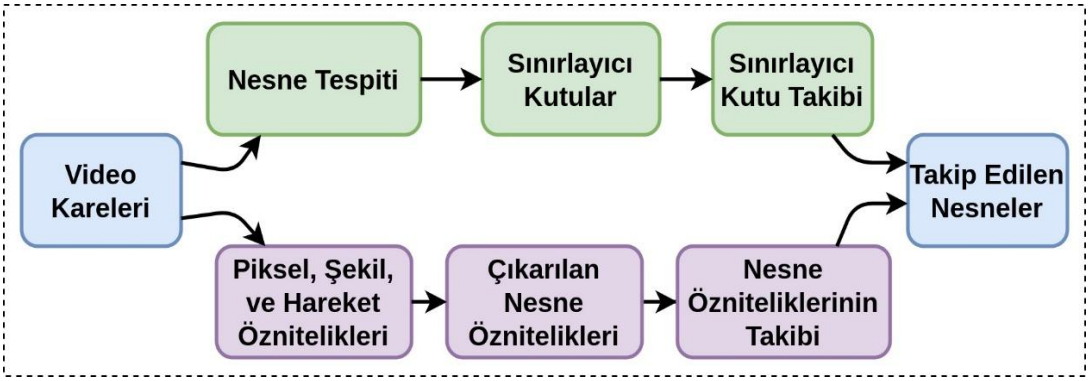


Şekil 2.9. Nesne tespit algoritmalarının doğruluk ve performans karşılaştırması (Redmon ve ark., 2018).

2.4. Nesne Takip Etme Yöntemleri

Araç (Nesne) takibi, çekilen belirli bir video çerçevesinden araçları (nesneleri) tanımanın ve tanımlamanın ötesinde, dinamikleri ve hareket özelliklerini yeniden tanımlamayı, ölçmeyi ve yoldaki araçların yaklaşık konumunu tahmin ederek, her bir araç için zamana bağlı yörüngesinin oluşturulmasını amaçlamaktadır. Kısaca, bu süreç nesne takibi veya nesne izleme olarak bilinmekte olup, bu yaklaşımlar tespit tabanlı ve öznitelik tabanlı nesne takip etme algoritmaları olmak üzere, genel olarak iki ana guruba ayrılmaktadır, Şekil 2.10. Birinci yaklaşım; nesne tespit aracılığyla elde edilen sınırlayıcı kutu bilgilerine dayanan nesne takip etme, ikincisi ise nesnelerin şekil, renk veya hareket özellekilerine göre nesne takip etme yaklaşımıdır.

Bu çalışmada, nesne tespit artı nesne takip yaklaşımı tercih edilmiştir, çünkü bu yöntem diğer, nesne öznetiliklerini esas alarak nesne takibi yapan algoritmalara nazaran daha hızlı, daha performanslı ve daha doğru sonuçlar üretmektedir. Araç izleme yaklaşımlarının başlıca sorunları; ölçüm ve sensör belirsizliği, veri ilişkilendirme ve izleme işleminin doğru bir biçimde yönetilememesi olarak sıralanabilir. Bu alt bölümde, araştırma literatüründe kullanılan yaygın araç takip yöntemlerini detaylandırıyoruz. Tartışma; monoküler ve stereo görüş yaklaşımlarını detaylandıran bölümlere ayırılarak incelenmiştir. Her iki kamera yapılandırmasında, ortak olan tahmin ve filtreleme yöntemleri var olsa da, genellikle tahmin edilen parametreler mevcut ölçümlere göre farklılık gösterir. Birçok monoküler izleme yöntemi, dinamikleri piksel cinsinden ölçüp tahmin ederken, stereo görüş yöntemleri dinamikleri metre cinsinden tahmin eder. Bu bölüme, kavşak ve karayolu araç tespiti ve takibi için monoküler ve stereo görüşü birleştiren çalışmaları tartışarak devam ediyoruz. Ardından, optimize edilmiş sistem mimarisine ve karayolu araç tespiti ve takibinin gerçek zamanlı uygulanmasına odaklanan çalışmaları tartışacağız. Bu bölümü, kavşak ve karayolu araç tespiti ve takibi için vizyonu diğer algılama modaliteleriyle birleştiren çalışmaların tartışılmasıyla sonlandırırız.



Şekil 2.10. Nesne takip etme yaklaşımı. Nesne tespit ile takip (üst taraf) ve özniteliklerle takip etme (alt taraf).

Monoküler Araç Takibi, araçları tipik olarak monoküler bir görüş sistemi kullanılarak, nesneleri görüntü düzleminde algılama ve izleme yöntemidir. Monoküler bir vizyon kullanarak nesne izleme yaklaşımı iki ana amaca hizmet eder; birincisi, görüntü düzleminde hareket ve araç konumunun tahminini kolaylaştırarak normal izleme yapmak. İkincisi ise sahte, yani yanlış pozitifleri filtreleyerek izleme işlemini gerçekleştirmektir (Sivaraman ve Trivedi, 2010). Bu tür nesne izleme; belirli bir çerçevede (Haselhoff ve Kummert, 2009) saptanmamış olan önceden tespit edilen araçların sözde farklı kimliklerini (ID'lerini) korumaya yardımcı olan zamansal tutarlılığı güçlendirme işlemidir. Monoküler görmenin amacı, hareketi ölçmek ve araçların piksel konumundaki ve piksel hızındaki konumunu tahmin etmektir. Piksel bazlı gözlem alanı, aracın görüntü düzlemindeki görünümüne bağlı olarak benzersiz bir şekilde vizyona dayalı izleme yöntemlerine yol açmaktadır. Benzersiz bir vizyona dayalı izleme örneği, şablon eşleştirmedir. Başka bir çalışmada, Haar dalgacık katsayıları ve SVM sınıflandırması kullanılarak görüntü düzleminde araçlar tespit edilmiştir. Araçlar, görünüş özniteliklerine göre benzerlik ölçüsü alınarak çerçeveden çerçeveye takip edilmiştir (Liu ve ark., 2007).

Genellikle, görü tabanlı izleme metotları, nesne izleme işlemini çapraz korelasyon puanlarına dayanarak gerçekleştirir, ancak, performansları yeteri kadar iyi değildir. Ama bu tür nesne takip etme yaklaşımları, özellik tabanlı izleme kullanılarak bir adım daha ileri götürülümüş, doğruluk ve performansta iyileştirmeler elde edilmiştir (Yilmaz ve ark., 2006).

Diğer bir çalışmada, araçlar Haar özniteliklerine benzer özellikler ve AdaBoost kademeli sınıflandırma yöntemi kullanılarak tespit edilmektedir. Aday araçların konumları, görüntü düzleminde Kalman filtresi kullanılarak tahmin edilmiştir. Görüntü düzlemindeki ölçümler, benzer özellik puanları için görüntü yaması üzerinde yerel bir arama ile belirlenmiş ve bu, dedektör belirli bir çerçevede arızalansa bile tahmini bir ölçüme izin vermektedir. Piksellerin görüntü düzleminde yerleşim şeklini baz alarak çalış bu algoritmaların yanı sıra, Optik akış (yani hareket eden bir leke veya nesneyi temsil eden piksellerin ardışık video çerçevelerindeki hareketi), görüntüdeki nesne olduğu tahmin edilen lekelerin yeni konumunu ve ilgi noktalarının yer değiştirmesini doğrudan ölçerek araçları izlemek için de kullanılmıştır (Zhu ve ark., 2005).

Öte yandan, kalssik araç izleme ve Bayes filtreleme teknikleri de, monoküler araç izleme literatüründe yaygın olarak kullanılmış yaklaşımlardır. Bu yaklaşımlarda takip için kullanılan öznitelikler, yani öznitelik durum vektörü; tipik olarak, görüntü düzlemindeki bir dikdörtgeni ve bunların kareler arası piksel hızlarını parametrelendiren piksel koordinatlarından oluşur (Chan ve ark., 2007).

Bu yöntemlerin dışında, Kalman filtresi, görüntü düzleminde tespit edilen araçların hareketini tahmin etmek için çok yaygın olarak kullanılan bir yaklaşımdır (Chang ve Cho, 2008; Arrospide ve ark., 2008; Aytekin ve Altuğ, 2010). Kalman filtresine dayalı araç takip yaklaşımları, araçların sadece koordinatlarını baz alarak araç izleme işlemini gerçekleştirir. Görüntülerdeki çok önemli özellikler olan piksel ve piksellerden oluşan lekeler (tahmini nesneler) Kalman filtresi tabanlı araç takip yaklaşımlarında kullanılamamaktadır. Dolayısıyla, Kalman filtresi tabanlı bu araç takip ayaklaşımlarının yetersiz kaldığı durumları çözebilmek amaçlı, Kalman filtresine çok benzer olan parçacık filtreleme yaklaşımları geliştirilimiştir. Parçacık filtreleme yaklaşımları, görüntü düzleminde araçları monoküler bir şekilde izleyebilmek için yaygın olarak kullanılmıştır (Chan ve ark., 2007; Idler ve ark., 2006; Sivaraman ve Trivedi, 2010; Takeuchi ve ark., 2010; Tehrani ve ark., 2012; Mei ve Ling, 2011).

Monoküler görüş, uzunlamasına mesafe tahmini ve araçların 3 boyutlu bilgilerini öznitelik olarak kullanan çeşitli araç izleme çalışmalarında da denenmiştir. Bu yaklaşımlarda, tipik olarak, zemin düzleminin düz olduğu varsayılmış veya parametreleri, ilgi noktasının tespiti ve RANSAC gibi sağlam bir mod uydurma adımı kullanılarak tahmin edilmiştir (Cui ve ark., 2010; Hilario ve ark., 2006; Fischler ve Bolles, 1981).

Başka bir çalışmada ise hem monoküler görüşe dayalı araç takip yaklaşımı hem de Kalman filtresine dayalı yaklaşım aynı anda kullanılmıştır. Bu yaklaşımlar araçların 3 boyutlu koordinatlarını tahmin etmek için bir dizi kısıtlama ve varsayımsal öznitelikler oluşturmak için monoküler görüş yaklaşımını kullanmış, sonrasında ise bu öznitelikler Kalman filtresine beslenerek daha iyi araç takip sonuçları elde edilelebileceği ileri sürülmüştür (Nuevo ve ark., 2010). Diğer bir çalışmada ise, araçların 3 boyutlu bilgilerinden oluşan öznitelikler, yer düzlemi tahmini kullanılarak çıkarılmış ve araçları izlemek için etkileşimli birden çok model kullanılmıştır, ve her model farklı parametrelere sahip bir Kalman filtresinden oluşturulmuştur. Bazı çalışmalarda üç boyutlu özniteliklerin en baskın olan hareketleri monoküler görme kullanılarak tahmin edilmiş ve hareketli nesneler Kalman filtresi kullanılarak 3 boyutlu olarak izlenmiştir. Bu araç izleme çalışmaları, monoküler ölçümlerden 3 boyutlu araç konumunu ve hız bilgilerini tahmin ederken, başka bir takım çalışmalar, araç takip etmek için RADAR, LIDAR gibi sensörlerden veya stereo görüşlü kameralardan alınan bilgilere göre araç takip etme işlemini gerçekleştirmişler. Aşağıda, bu çalışmalar ayrıntılı olarak açıklanmıştır (Hoffmann, 2006; Yamaguchi ve ark., 2006a; Yamaguchi ve ark., 2006b).

Stereo-Görüşlü Araç Takibi, Stereo görüş sistemlerini kullanan araç takip etme yaklaşımları, yol ve kavşaklarda tespit edilen araçların konumunu ve hızını metre cinsinden ölçmek ve tahmin etmek amaçlı geliştirilmişlerdir. Durum vektörü, genellikle, aracın yanal ve boylamsal konumu, genişliği ve yüksekliğinin yanı sıra hız bilgilerini de içermektedir. Tahmin, genellikle, doğrusal hareket ve Gauss gürültüsü varsayılıp optimal kabul edilen Kalman filtresi kullanılarak gerçekleştirilmiştir (Franke, 2005). Gerçekte, aracın savrulma oranı aracın dönüş

davranışını tanımladığından, araç hareketi doğrusal olmamaktadır. Genişletilmiş Kalman filtresi (EKF), genellikle tahmin için hareket denklemlerini doğrusallaştırarak doğrusal olmayan parametreleri tahmin etme amaçlı da kullanılmıştır (Barth, 2009). Başka bir çalışmada, Partikül filtreleme, EKF'nin doğrusallaştırması yerine önemli örnekleme özniteliklerini yeniden örnekleyerek hem doğrusal hem de doğrusal olmayan hareket parametrelerini ölçmek için bir alternatif olarak kullanılmıştır (Catalin ve Nedevschi, 2008).

Kalman filtresi, monoküler görüşlü araç takip yaklaşımlarında kullanıldığı gibi, stereo görüşlü araç takibi için de yaygın olarak kullanılmıştır. Araçların iki boyutlu görüntü düzlemindeki koordinat özellikleri, Kalman filtresi ile eşitsizlik filtrelemesinden geçirilerek, araç takip işlemi gerçekleştirilmiş ve daha doğru araç takip sonuçları elde edildiği bildirilmiştir (Lim ve ark., 2010). Bazı çalışmalarda, stereo görüntü eşleştirmede, gürültü, genellikle beyaz Gauss gürültüsü olduğu nedeniyle, Gaussion gürültüsü yöntemi kullanılarak modeler oluşturulmuş, oluşturulan modeler ise zamana bağlı olarak Kalman filtresinden geçirilerek daha temiz uyumsuzluk haritaları üretilebilmiştir. Üretilen haritalar kullanılarak araçlar ardışık video çerçevelerinde takip edilmiştir (Franke ve ark., 2005; Broggi ve ark., 2011; Rabe ve ark., 2007).

Bunun dışında, Kalman filtresi, araçların 3B'lu koordinat noktalarını kullanarak tek tek izlemek için de kullanılmıştır. Kalman filtresi, kübik araç modellerine uyan, sabit derinliğe yakın ara dikey elemanlar gibi stikselleri izlemek için de kullanılmıştır Badino ve ark., 2012; Erbs ve ark., 2011; Rabe ve ark., 2007). Bazı çalışmalarda ise, monoküler görüş ve stereo görüşlü metotlar birleştirilerek daha başaralı araç takip sonuçları elde edilmiş. Bu çalışmalarda, araçlar, AdaBoost temelli bir sınıflandırma kullanılarak monoküler düzlemde tespit edilmiş ve stereo alanda Kalman filtresi kullanılarak 3 boyutlu olarak takip edilmiştir (Sivaraman ve Trivedi, 2011; Kowsari ve ark., 2011).

Diğer bir çalışmada ise araçların hem konumları hem de hız bilgileri, genişletilmiş Kalman filtresi kullanılarak tahmin edilmiştir. Bununla birlikte, genişletilmiş Kalman

filtresi, stereo uyumlu kameralar kümesi ile algılanan nesneleri izlemek için de kullanılmıştır. Genişletilmiş Kalman filtresi (EKF), araçların yalpalama oranının yanı sıra konumu ve hızını tahmininde de kullanılıldığı bildirilmiştir. EKF, özellikle araçlarının hareketinin doğrusal olmadığı durumlardaki gözlemsel model miktarlarını hesaba katmak için stereo görüşlü araç izlemede yaygın olarak kullanılmıştır. Başka bir çalışmada ise EKF, araçların yalpalama oranını ve karşılık gelen dönüş davranışını tahmin etmek için kullanıldığı bildirilmiştir. Araç izleme için genişletilmiş Kalman filtresi, bazı çalışmalarda, kamera konumlandırması nedeniyle özellikle uygun olduğu, yol kenarına monte edilmiş stereo donanım, kameranın referans çerçevesine göre izlenen araçların, hareketini, özellikle, doğrusal olmayan hareketini gözlemleyebildiği belirtilmektedir (Barth ve ark., 2008; Grinberg ve ark., 2009; Barth ve ark., 2010; Bota ve Nedeveschi, 2011; Moqqaddem ve ark., 2011).

EKF, araçların 3B'lu konumunu stereo görüntü konumuna ve uyumsuzluğa eşlemenin doğrusal olmadığı durumları modellemek için de kullanılmıştır. Aynı yaklaşım, benzer başka bir çalışmada, bağımsız olarak hareket eden nesnelerin konumu ve bu nesnelerin Kalman filtresi kullanılarak tahmin edilen hareket bilgilerini kullanarak, araçların daha baskın hareketlerini tahmin etmek amaçlı kullanılmıştır (Lategahn ve ark., 2011; Kitt ve ark., 2010; Lim ve ark., 2011).

Kalman filtresi ve EKF yaklaşımlarının yanı sıra, stereo görüntülerden daha sağlıklı araç takibi yapabilmek amaçlı, partikül filtreleme yaklaşımları da oldukça yaygın kullanılmıştır. Partikül filtresinin içerdiği çoklu hipotez yaklaşımları bazı olasılık fonksiyonları ile ağırlıklandırılarak, EKF'nin yaygın olarak kullanıldığı, doğrusal olmayan parametrelerin tahmininde de bir alternatif olarak kullanılmıştır. Bunun dışında, araçların 3B'lu öznitelikleri partikül filtresi ile işlenerek, araçların 3B'lu konumları ve sapma oranları tahmin edilmiştir. Başka bir çalışmada, izlenen araçların hareketi, hareketten önceki gözlem verilerinden öğrenilen tam yörüngeler bir parçacık filtresi ile işlenip, araçların video sahnelerinden anlık olarak kayboluş sorunları çözümlendiği bildirilmiştir. Diğer bir çalışmada ise, karayolu ortamı, iki amaca hizmet eden parçacık filtreleriyle işlenerek modellenmiştir. Yani, karayolu ortamından çekilen iki boyutlu görüntü düzlemi gözeneklere bölünmüş ve

gözeneklerin doluluk oranı, yani bu gözeneklerin herhangi bir nesne içerip içermediği ile ilgili bilgiler edinilmiş. Edinilen bilgiler sayesinde nesneler ve kara yolu ortamındaki engeller tespit edilerek karayolu ortamının modeli çıkarılmıştır.

Araç takip işlemi için, tek yöntem kullanıldığı zaman, kullanılan yöntemin zayıf kaldığı durumlarla ne kadar sık karşılaşılırsa, araç izleme performansı o kadar seviyede olumsuz etkilenmektedir. Bu olumsuzluklarla baş etmek amaçlı, farklı hareket modları verilen bir aracın hareketini tahmin etmek için araç izlemede birden çok modelin etkileşimi kullanılmıştır. Kavşaklar için araç izleme yaklaşımı geliştirilen bir çalışmada, kavşaklarda karşıdan gelen araçların hareketini hız ve yalpalama oranı özellikleri açısından modellemek için dört farklı baskın mod verileri, dört farklı model ile işlenmiş ve daha performanslı araç takip sonuçları elde edildiği bildirilmiştir. Bu yaklaşımdaki ana amaç, hızın sabit mi yoksa hızlandırılmış mı olduğunu ve yalpalama oranının sabit mi yoksa hızlandırılmış olduğunu belirlemekmiş. Bu sorular doğru yanıtlandığı zaman model uyumu daha doğru belirleneceği ile ilgili fikir ileri sürülmüştür. Onun dışında, model uyumu, her bir tahmin edici özniteliğin hata kovaryansı kullanılarak belirlenebildiği de bildirilmiştir. Model uyumu belirlendikten sonra komşu modlar arasında geçiş yapmak için bir durum geçiş olasılığı kullanılmıştır. Bu şekilde geliştirilen nesne takip etme yaklaşımı, etkileşimde bulunan çoklu modeller kullanılarak gerçekleştirilmiştir. Nense takip etme sürecinde kullanılan ölçümler daha hassas yapılamadığından, ve buna ek olarak, tüm hareket parametrelerinin tek bir doğrusal veya doğrusallaştırılmış filtre ile doğru bir şekilde tahmin edilemeyeceği anlaşıldıkça, etkileşimli birden çok modelin bir arada kullanımı muhtemelen daha da artacağı ve yaygınlaşacağı öngörülmüştür (Catalin ve Nedeveschi, 2008; Hermes ve ark., 2010; Danescu ve ark., 2011).

Özet olarak, belirli bir zaman aralığı için veya gerçek zamanlı olarak bir kavşak, şehir içi yol veya otoyol video sahnelerinden nesneleri (araçları) takip ederek araçların yörüngelerinin çıkarılması; araç hızı, hareket yönü, toplam (veya kategorik) sayma ve araçların belirlenen bölgeye giriş ve çıkış noktaları arasında geçen süre gibi oldukça değerli trafik akışı bilgiler elde edilebilmektedir. Bu tür bilgiler ile bir şehir

veya kentsel alanda istenen herhangi bir bölgenin trafik durumu kolayca izlenebilmekte ve yönetilebilmektedir. Yukarıda detaylı olarak bahsi geçmekte olan Optik akış ve Kalman filtreleri dâhil olmak üzere kavşak veya karayollarında araç takip problemini gerçek zamanlı olarak çözmek için literatürde piksel tabanlı ve sınırlayıcı kutu tabanlı nesne izleme çalışmaları bulunmaktadır. Ancak, bu yaklaşımların tamamı, nesne izleme problemini karayollarında veya kavşaklarda gerçek zamanlı olarak çözmede yetersiz kalmaktadır (Rouf ve ark., 2010). Optik akış ve diğer piksel tabanlı yöntemler, doğası gereği çok yavaştır ve gerçek zamanlı uygulamalar için uygun değildir; Kalman filtresi tabanlı nesne takip etme algoritma ve yaklaşımları, nesnelerin minör bir hareketine karşı çok hassas olduğu dolayısıyla, sık sık yanlış tahminde bulunmakta, bunun dışında, belirli bir çerçevede nesne sayıları arttığında, bu takip algoritmalarının icra hızı düşmektedir (Nguyen ve Brilakis., 2018; Liu ve ark., 2019; Song ve ark., 2019). Çünkü Kalman filtresine dayalı nesne takip etme yaklaşımları, araç izleme işlemi sırasında, her örnekleme aralığında, her araç için yeniden hesaplamalar yapar. Bu nedenle, Kalman filtresi tabanlı izleme algoritmasının icra karmaşıklığı $O(n^3)$ seviyelerine ulaşmaktadır. Dahası, YOLOv3 algoritması ve Kalman filtresi tabanlı nesne takip yaklaşımı, YOLOv3'nun kararsız nesne yerelleştirme (iki boyutlu görüntü düzleminde nesnelerin konumlandırılması) doğası nedeniyle, birbirleriyle uyumsuz hale gelebilmektedir. YOLOv3 algoritması, durağan veya hareket eden nesneleri art arada gelen video karelerinde, bulundukları yerden biraz farklı konumlarda algılamaktadır. Kalman filtresine dayalı geliştirilen izleme algoritmaları ise, herhangi bir minör harekete karşı aşırı duyarlıdır, çünkü, bu algoritma, araç hızlarını $v = v_0 + at$, ve koordinatları tahmin etmek için $x = x_0 + v_0t + at^2/2$, $y = y_0 + v_0t + at^2/2$ formülleri, ve bir çok durum bilgilerini kullanmaktadır (Xiao ve ark., 2020). Bu, Kalman filtresine dayalı araç takip algoritmalarını sınırlayıcı kutu noktalarının küçük hareketlerine karşı çok hassas hale getirmektedir. Buna ek olarak, Kalman filtresi, sadece bir önceki çerçeve parametresini kullanarak yeni konum tahmini yapmaktadır. Sadece bir önceki durum parametresini dikkate alarak yeni pozisyonları tahmin etmek, çoğunlukla yeterli olmayıp, bu bilgi eksikliği tahminlerde belirsizliğe neden olabilmektedir (Yang ve ark., 2019; S. ve S., 2018). Bu nedenle, sağlam bir kavşak veya karayolu trafik akışı izleme ve yönetim sistemi geliştirilebilmesi için

YOLOv3'nun dengesiz nesne yerelleştirme doğasına uyumlu, sağlam bir araç takip etme ve veri ilişkilendirme algoritmalarının geliştirilmesine ihtiyaç duyulmaktadır.

2.5. Veri Kümeleri ve Toplama Teknikleri

Bilgisayarla görü yöntemlerine dayalı trafik akış verileri toplama, izleme ve yönetim sistemleri, genel olarak, kavşak veya normal şehir içi yol veya otoyolları kamera sensörleri vasıtasıyla izleyip, elde ettiği görüntüleri nesne tespit ve nesne takip algoritmalarını kullanırak üç seviyeli veri üretmektedir. Birincisi düşük seviyeli veri, yani nesne tespit sonuçları. Nesne tespit sonuçları nesnelerin türünü, konumunu ve nesnesellik güven değerini içeren bilgilerdir. Bu bilgiler, nesne takip etme ve veri ilişkilendirme modülünün girdisi olarak kullanılır ve nesneler video çerçeveleri böyunca vakite bağlı olarak takip edilir. Takip etme aşaması sonucunda, her bir nesneye özgü, yegana kimlik atanır ve bu kimlik sayesinede bir nesnenin video sahnesine giriş ve çıkış noktaları arasındaki izi çıkarılmaktadır. Bu veriler orta seviyeli bilgiler olarak bilinmektedir. Ve son aşamada, hesaplayarak çıkarılan orta seviyeli nesnelerin trafikteki iz verilerinden yüksek seviyeli trafik bilgileri elde edilmektedir. Bu bilgiler; giriş ve çıkış yönlerine, nesnelerin sınıflarına veya toplam araç sayılarına göre trafik sayımı, hız bilgileri, giriş çıkış noktaları arasındaki süre, sürücü davranışları gibi verileri içermektedir. Dolayısıyla, bir traffik akış verisi toplama ve izleme veya yönetim sistemini geliştirebilmek ve bu tür bir sistemin çıktılarını doğruluğunu ve performansını ölçebilmek için üç türlü veri kümesine ihtiyaç duyulmaktadır. Bunlar; nesne tespit, nesne takip ve trafik akış bilgilerinden oluşan veri kümeleridir. Aşağıda bu üç aşama için oluşturulmuş veri toplama teknikleri ve veri kümelerinden bahis edeceğiz.

Nesne tespiti veri toplama teknikleri ve veri kümeleri. Nesne tespiti, sınıflandırması, ve segmentasyonu için kullanılan veri kümeleri; insanlar tarafından tam denetimli bir şekilde veya denetimsiz, ve yarı denetimsiz nesne tanıma metotları yardımıyla etiketlenerk oluşturulmaktadır. Veri kümesi etiketleme bir veya birden fazla insan tarafından ücretli veya birçok nesne etiketleme (object annotation) araçları kullanılarak yapılmaktadır. Günümüzde en çok kullanılmakta olan genel nesne tespit

veri kümeleri sırasıyla; Pascal VOC 2007 ve 2012, ImageNet ve Microsoft COCO'dur (Everingham ve ark., 2015; Deng ve ark., 2009; Lin ve ark., 2014). Bu veri setleri nesneleri tespit etme, tanıma veya segmentatsiyon işlemleri için geniş bir yelpazedeki kullanıcı kitlesine sahiptir. Bu veri setleri, Dünya'nın birçok noktasından binlerece uzman tarafından etiketlenmiş ve denetlenmiştir.

Pascal VOC; Pascal Görsel Nesne Sınıfları (VOC – Visual Object Classes) veri seti iki bileşenden oluşur: (i) kesin referans notu ve standartlaştırılmış değerlendirme yazılımı ile birlikte herkese açık bir görüntü veri seti; ve (ii) yıllık yarışma ve çalıştay. Veri seti, 20 sınıf içerir. Eğitim ve doğrulama olarak ikiye ayrılmış veriler, 27.450 civarında seçilmiş ilgi bölgesi (region of interest, ROI) açıklamalı nesne ve 6.929 segmentasyon içeren 11.530 görüntüye sahiptir. Bu veri kümesi oluşturmada; nesne sınıflandırma, algılama, bölümleme, eylem sınıflandırması ve kişilerin veri kümesini düzenlemedeki insani hata faktörü gibi beş zorluğa değinilmiştir. Veri seti, VOC veri kümeleri üzerinde sunulan algoritmaların performansını analiz etmek için bir dizi yeni değerlendirme yöntemleri denenmiştir: iki algoritmanın performansındaki farklılıkların önemli olup olmadığını belirlemek için bir önyükleme yöntemi; performansın farklı pozitif örnek oranlarına sahip sınıflar arasında karşılaştırılabilmesi için normalleştirilmiş bir ortalama hassasiyet; zor ve kolay görüntülerin tanımlanabilmesi için performansı birden çok algoritmada görselleştirmek için bir kümeleme yöntemi; ve bunların tamamlayıcılığını ve birleşik performansını ölçmek için ise sunulan algoritmalar üzerinde bir ortak sınıflandırıcı kullanılmıştır. Ayrıca çalışmada, diğerlerinin yöntemlerini kullanarak topluluğun zaman içindeki ilerlemesini de analiz edilmiştir. Mesala, (Avrupa Bilgisayar Görüsü Konferansı Bildirileri, 2012) meydana gelen hataların türlerini belirlemek için (Everingham ve ark., 2015).

ImageNet Dataset, İnternetteki görüntü verilerinin patlaması, görüntüleri ve multimedya verilerini etiketleyerek endekslemek, geri getirmek, düzenlemek ve bunlarla etkileşime girmek için daha karmaşık ve sağlam modeller ve algoritmalar geliştirme potansiyelini ortaya çıkarmıştır. Ancak tam olarak ne kadar verinin kullanılabileceği ve düzenlenebileceği kritik bir sorun olmaya devam etmekteydi.

Bundan dolayı, yukarıdaki sorunları, giderebilen veri kümesi oluşturulmuştur. Bu veri kümesi, WordNet yapısının omurgası üzerine inşa edilmiş büyük ölçekli bir görüntü ontolojisi olan "ImageNet" adlı yeni bir veritabanıdır. ImageNet, WordNet'in 80.000 fazla görüntülerinin çoğunu ortalama 500-1000 temiz ve tam çözünürlüklü görüntü ile doldurmayı amaçlamaktadır. Bu, WordNet'in anlamsal hiyerarşisine göre düzenlenmiş on milyonlarca açıklamalı görüntü ile doldurularak sonuçlandırılmıştır. Bu veri kümesi oluşturma çalışması, mevcut durumda ImageNet'in ayrıntılı bir analizini sunmaktadır: 5247 synset ile 12 alt ağaç ve toplamda 3,2 milyon görüntü. ImageNet'in ölçek ve çeşitlilik açısından çok daha büyük olduğunu ve mevcut görüntü veri setlerinden çok daha doğru olduğu gösterilmiştir. Böylesine büyük ölçekli bir veritabanı oluşturmak zorlu bir iştir. Veri toplama şemasını Amazon Mechanical Turk ile anlatmışlar. Son olarak, nesne tanıma, görüntü sınıflandırma ve otomatik nesne kümelemedeki üç basit uygulama aracılığıyla ImageNet'in kullanışlılığını gösterilmiştir. ImageNet'in ölçeğinin, doğruluğunun, çeşitliliğinin ve hiyerarşik yapısının bilgisayar görüşü topluluğundaki ve ötesindeki araştırmacılara benzersiz fırsatlar sunabileceği öngörülmiştir. Sonradan ise 3,2 milyonluk veri seti, binden fazla sınıfa ait olan etiketlenmiş, sırasıyla 14 ve 22 milyona görüntüye ulaşmıştır (Deng ve ark., 2009).

Microsoft COCO (Common Objects in Context), nesne tanıma sorununu daha geniş sahne anlama sorunu bağlamına yerleştirerek nesne tanımada son teknoloji ürününü ilerletme hedefine sahip yeni bir veri kümesidir. Bu veri kümesi, doğal bağlamlarında ortak nesneleri içeren karmaşık günlük sahnelerin görüntülerini toplayarak elde edilmiştir. Nesneler, kesin nesne yerelleştirmesine yardımcı olmak için her örnek için bölümlemeler kullanılarak etiketlenmiş olup, bu veri kümesi, 4 yaşındaki bir çocuk tarafından dahi kolayca tanınabilecek 91 nesne sınıfının fotoğraflarını içermektedir. 328 bin görüntüde toplam 2,5 milyon etiketli örnekle, veri kümesinin oluşturulmasında; kategori algılama, örnek belirleme ve örnek bölümleme için yeni kullanıcı arayüzleri aracılığıyla kapsamlı çalışanların katılımından yararlanmıştır. Veri setinin PASCAL, ImageNet ve SUN ile karşılaştırmalı olarak detaylı bir istatistiksel analizini sunulmuştur. Son olarak, bir Deforme Edilebilir Parça Modeli

kullanılarak sınırlayıcı kutu ve segmentasyon algılama sonuçları için bir temel performans analiz bilgileri de sağlanmıştır (Lin ve ark., 2014).

Bu üç veri kümesi genel olarak yan veya yol kenarından çekilmiş nesne görüntülerini içermekte ve genel amaçlı veri seti olduğu için spesifik yol veya kavşak bölgelerindeki ortalama nesne tespit ve sınıflandırma oranı çok düşük seviyelerdedir. Dolayısıyla, bu veri setleri ile eğitilmiş hergangi bir derin öğrenme mimarisi kavşak veya otoyollardaki trafik akışı izleme sistemleri için direk olarak kullanılamamaktadır. Kullanıldığı taktirde de nesne tespit ve sınıflandırma doğruluğu çok düşük seviyelerde seyretmektedir. Bu durum, trafik akış sistemlerinin veri çıkarım işleminin genel doğruluğunu olumsuz etkilemekte olup istenilen trafik bilgilerini doğru bir biçimde çıkarılamamasına neden olmaktadır. Bu sebeplerden ötürü, bu çalışmanın araç tespiti ve sınıflandırma modülü için uzman ekip tarafından etiketlenerek yeni bir veri seti hazırlanmıştır. Sonra geliştirilen veri setleri YOLOv3 algoritmasına beslenmiş olup, seçilmiş kavşak ve otoyol bölgelerine özgü YOLOv3 tabanlı araç tespit ve sınıflandırma modelleri bir hafta boyunca eğitilerek hazırlanmıştır. Bu hazır veriler ile yüksek doğruluklu nesne tespit ve sınıflandırma sonuçları elde edilmiştir.

Nesne takibi veri toplama teknikleri ve veri kümeleri. Nesne takip etme veya izleme yaklaşımları için kullanılan veri kümeleri de nesne algılama için geliştirilmiş veri kümeleri gibi ya insanlar tarafından tam denetimli olarak, ya denetimsiz öğrenme metodu aracılığıyla, ya da yarı denetimli öğrenme veya nesne tanıma metotları yardımıyla etiketlenerk oluşturulmaktadır. Veri kümesi etiketleme, bir veya birden fazla insan tarafından çeşitli nesne etiketleme (object annotation) araçları kullanılarak yapılmaktadır. Nesne takip etme veri setleri tekli nesne izleme veya çoklu nesne takip etme (Multi-Object Tracking, MOT) olarak ikiye ayırlmaktadır.

Çoklu Nesne İzleme Veri Kümeleri. Çoklu nesne izleme (MOT) görevinde, bilinen bir sınıf kümesinden başlangıçta bilinmeyen sayıda hedef, bir videoda sınırlayıcı kutular olarak izlenmektedir. Özellikle hedefler olay yerine herhangi bir zamanda girip çıkabilir ve uzun süreli tıkanma (occlusion) ve görünüm değişikliklerinden

sonra kurtarılmaktadır. Birçok MOT veri seti sadece sokaklardaki insanların görüntü senaryolarına odaklanmış, örneğin, araca monteli bir kameradan video içeren KITTI izleme veri seti (Geiger ve ark., 2012); veya yayaları çeşitli farklı bakış açılarından gösteren MOTChallenge veri kümeleri (Leal-Taixe ve ark., 2015; Milan ve ark., 2016). UA-DETRAC (Wen ve ark., 2020; Lyu ve ark., 2017) ayrıca sokak sahneleri de içerir ancak yalnızca araçlar için tanımlamalar içermektedir. Başka bir MOT veri seti, çeşitli sahnelerde insan yörüngelerinin tanımlanmasını sağlayan PathTrack'tir (Manen ve ark., 2017). PoseTrack (Andriluka ve ark., 2018), videolardaki birden çok kişi için ortak konumların ek tanımlamalarından oluşmaktadır. Bu veri kümelerinin hiçbiri, tanımlanmış nesneler için bölümleme maskesi sağlamaz ve bu nedenle karmaşık etkileşimleri yeterli ayrıntıda tanımlayamaz. Bu eksiklikler, araç takip etme algoritmaların eğitimi veya testlerinin yapılabilmesi için büyük engel oluşturmaktadırlar. Bu nedenle, kavşak, şehir içi yollar, veya otoyollar için sağlam araç takip etme veri kümeleri geliştirilmesi gerekmekteydi. Bu çalışmamızın özü araç (nesne) takip ve araçların izini çıkarma olduğundan dolayı, kavşak ve otoyollar için araç takip etme işleminin doğruluğunu ölçmek amaçlı seçilen bölgelere özgü araç takip veri kümeleri oluşturulmuştur (Voigtlaender ve ark., 2019).

Trafik akış veri çıkarımı ve tahmin etme teknikleri ve veri kümeleri. Bu aşama için herhangi bir veri setine ulaşamadığımız nedeniyle kavşak ve otoyollar için araçların toplam ve kategorisine göre sayım, giriş çıkış noktaları arasındaki ortalama hız ve süre zarf bilgileri uzman (insan) tarafından yapılmıştır. Ama nesne tespit ve nesne takip kısımlar için genel amaçlı nesne tespiti ve nesne takibine yönelik birçok veri kümesi bulunmaktadır. Ancak, yol veya kavşak trafiğinin ünsürü olan nesneleri yani araçları tespit ve takip etmeye yönelik veri kümeleri mevcut değildir. Bunun dışında, araç sınıflandırma ve sayma, hız ölçme, yön belirleme ve diğer trafik bilgileri çıkarımı için veri kümeleri bulunmamaktadır. Bu sebeplerden ötürü, bu projenın çıktıları olan nesne tespit, nesne takip ve iz çıkarımı ve son olarak trafik akış bilgilerini doğrulayabilmek amaçlı birçok kavşak ve otoyollar incelenerek videolar seçilmesi ve gerekli veri kümeleri hazırlanması gerkemekteydi. Bu projenin her aşaması için önem kesbeden veri setleri uzmanlar (proje ekibi) tarafından oluşturulmuştur.

BÖLÜM 3. METODOLOJİ

3.1. Trafik Akış Verisi Toplama ve İzleme Sisteminin Genel Mimarisi

Bu çalışma kapsamında geliştirilen trafik akış verisi toplama ve izleme sistemi, dört katmandan oluşmaktadır. Birinci katman; araç (nesne) tespiti ve sınıflandırma, ikinci katman tespit edilen araçların sınıf, konum ve güven değerlerinden oluşan araç tespit sonucunu (çok boyutlu diziyi) işleyerek nesneleri ard arda gelen video karelerinde takip etme, üçüncü katman ise araç takip çıktılarını kullanarak veri ilişkilendirme yani araçlara ait hareket izlerini çıkarma, ve en son katmanda ise araçların izlerinden; araçların kategorisine göre sayım, toplam sayım, hız , yön ve araçların, belirlenen kavşak veya otoyol bölgelerine giriş-çıkış noktaları arasındaki geçen süre gibi trafik akış bilgilerinin tahmini olarak hesaplanmasıdır. Birinci katmanda, derin öğrenme yönteminin CNN mimarisine dayanan gerçek zamanlı YOLOv3 nesne tespit algoritması tekrardan eğitilerek araç tespit işlemine uyarlanmıştır.

Şekil 3.1. Sistemin ana akış digramı.

İkinci kısmda, kavşaklar için merkezi nokta tabanlı, karayolu, veya otoyollar için esa sınırlayıcı kutu bilgilerine dayalı araç takip etme algoritmaları geliştirilmiştir. Araçların hareket izini çıkarmak için basit bir arama ve eşleştirme algoritmaları kullanılmıştır. En son katman için sayım, yön belirleme, süre gibi trafik akış bilgileri direk veri tabanından veri sorgulama yöntemi ile çekilmiş ve ortalama hız bilgisi ise hareket izlerinden Denklem 3.1'deki formüle göre hesplanmıştır. Burada, v_{avg} – ortalama hız, Δs – iki çerçeve arasındaki nesnelerin kat ettiği mesafe, Δt ise zaman dilimidir. $x_ı$, $x_{ı+1}$ değerleri ise i ve i+1 inci çerçevelerdeki bir nesneye ait konumlardır. Sistemin genel mimarisi, Şekil 3.1. ile gösterilmektedir.

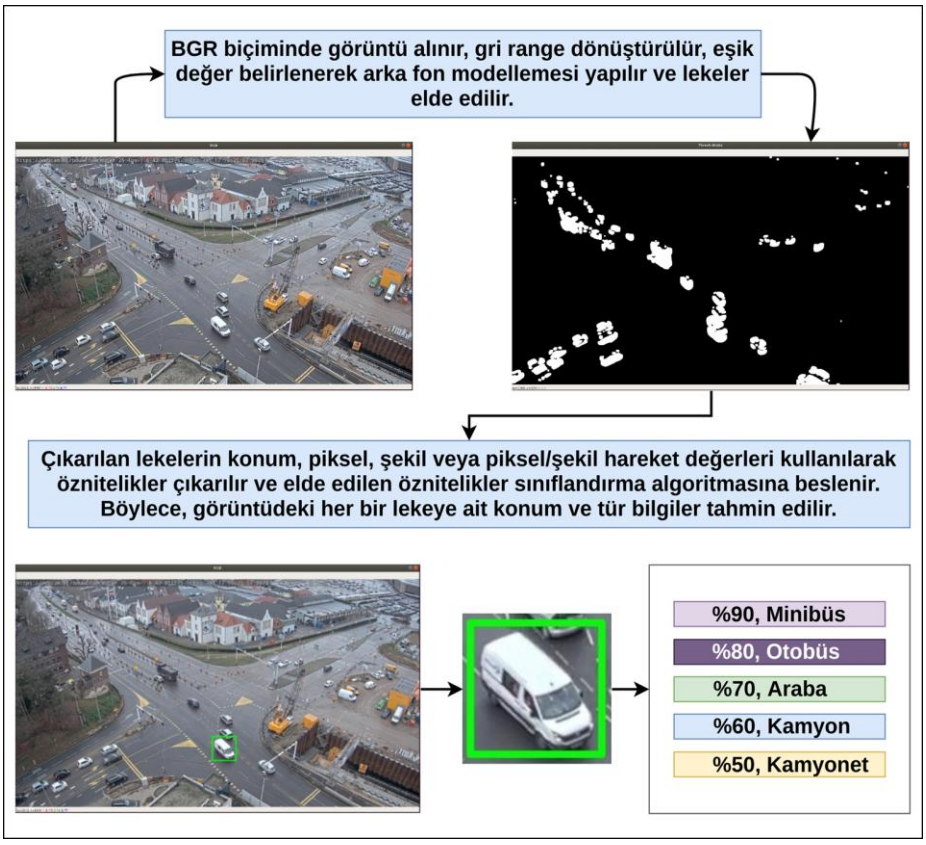
$$v_{avg} = \frac{1}{n}\sum_{i=0}^{n} v_i,\ v = \Delta s/\Delta t,\ \Delta s = x_{i+1} - x_i,\ \Delta t = t_{i+1} - t_i, \quad (3.1)$$

Bu çalışmada, kavşaklar için ayrı otoyollar için ayrı modüller geliştirilmiştir çünkü, kavşak ve yol veya otoyolların yapısı birbirinden farklıdır. Kavşaklarda çoklu giriş-çıkış yönleri mevuttur. Yollarda ise genel olarak geliş ve gidiş şeritleri vardır. Farklı yapıları itibariyle, kavşak ve otoyollar için ayrı ayrı nesne tespit modeller eğitilmiş ve nesne takip etme algoritmaları geliştirilmiştir. Kavşaklar için geliştirilen ensne takip algoritması, nesnelerin merkezi noktalarına odaklanarak araç takibi yapar, otoyollar için geliştirilen sistemde ise, araç takibi, nesnelerin sınırlayıcı kutularının tüm yanları, yani x_merkez, y_merkez, en, ve boy özellikleri gibi bilgiler kullanılarak gerçekleştirilmektedir. Aşağıdaki alt bölümlerde, ilk önce araç tespit modeli, sonra kavşak ve otoyollar için araç takip algoritması, en son olarak ta kavşaklar için ayrı, otoyollar için ayrı trafik akış bilgisi çıkarma sistemleri açıklanmıştır.

3.2. Araç Tanıma, YOLOv3 Algoritması

Nesne tespit etme (tanıma); bir girdi görüntü veya video çerçevesindeki nesnelerin bulunduğu konumu tahmini olarak belirleyip nesnelerin sınıfını ayrıştıran bir yaklaşımıdır. Geleneksel makine öğrenmesi yaklaşımları ile yürütülen bu sürecin işleyiş biçimi Şekil 3.2. ile gösterilmektedir. İlk adımda, bir girdi görüntü veya video çerçevesi alınır, sonra geleneksel makine öğrenmesi algoritmaları yardımıyla arka

fon modellemesi (background subtraction modeling) yapılır. Arka fon modellemesi sonucu elde edilen lekelerden (extracted blobs) manuel olarak tanımlanan öznitelikler çıkartılır ve bu öznitelikler bir sınıflandırma algoritmasına beslenir. Netice olarak, girdi görüntü veya video çerçevedeki nesnlerin konumu ve türü tahmin edilir. Tahmin edilen nesnelerden güven değeri en büyük olan objenin sınıfı sonuç olarak kabul edilir. Ancak, klasik makine öğrenmesine dayalı bu nesne algılama sistemlerinin bir girdi görüntü veya video çerçevesini işleme süresi fazla uzun sürüyor. Buna ek olarak, her bir nesnenin özelliklerini iyi bir biçimde tanımlamak çok zor ve neredeyse imkansız bir görevdir. Bundan ötürü, nesne tanıma sistemleri için daha hızlı olduğu iddia edilen RCNN, FR-CNN, SSD, YOLO (v1, v2, v3, v4, v5) gibi CNN tabanlı derin öğrenme algoritmaları geliştirilmiştir.

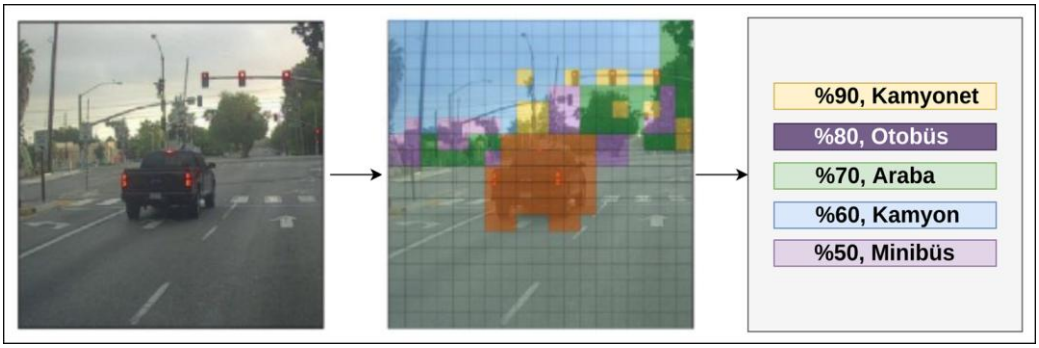


Şekil 3.2. Geleneksel Makine Öğrenmesi Algortimaları ile Nesne Tespiti.

Fakat, YOLOv3'nun tüm versiyonları dışındaki, bu CNN tabanlı mimariler beklenen görüntü işleme hızı olan en az 20 fps'e (frame per second) ulaşamamış durumdadır. Bunun dışında, bu derin öğrenme algoritmalarının YOLOv3 da dahil olmak üzere

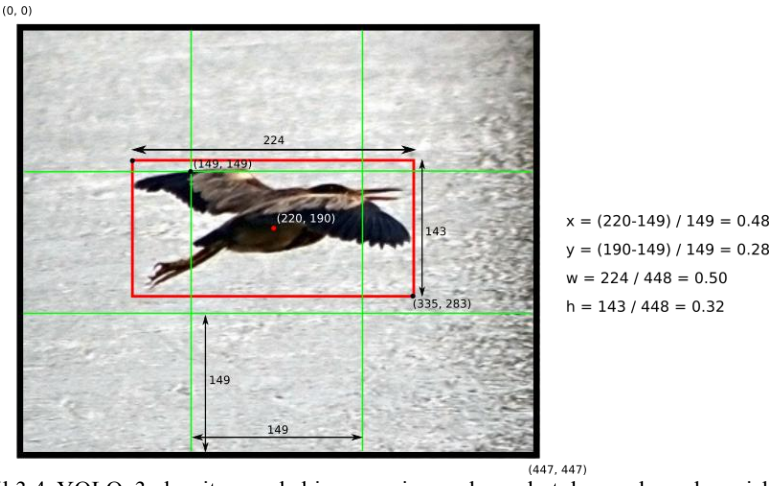
sınıflandırma ve nesne tespit doğruluğu %57'nin altındadır ki bu değer, akıllı ulaşım uygulamaları dahil olmak üzere birçok sistem için yetersiz kalmaktadır. Ancak, YOLOv3 algoritması seçilen bölgeden direk elde edilmiş görüntülerdeki nesneler, iyi bir biçimde etiketlenerek eğitildiği vakit, %90'nın üzerinde ortalama nesne tespit doğruluğunu yakalayabilmektedir. Bu sebeplerden ötürü, mezkur çalışmamızda YOLOv3 nesne tespit algoritması tercih edilmiştir. Yani YOLOv3 algoritmasının hızlı olması ve tekrar eğitildiğinde yüksek nesne tespit ortalama doğruluğuna ulaşabiliyor olması dolayısıyla, YOLOv3 algoritması kullanılmıştır.

YOLOv3 (Sadece Bir Kez Bakarsınız), nesne algılama amaçlı yaygın bir şekilde kullanılmakta olan bir yapay sinir ağı tabanlı yaklaşımdır. Nesne algılama işlemi, bir görüntüdeki nesnelerin bulunduğu konumun belirlenmesinin yanı sıra bu nesnelerin sınıflandırılmasından da sorumlu bir süreçtir. Bunun için R-CNN ve varyasyonları gibi önceki yöntemler, bu görevi birden çok adımda gerçekleştirmek için ardışık modüllerden oluşan yapı kullanmaktadır. Bu durum, her bir bileşenin ayrı ayrı eğitilmesi gerektiğinden algoritmaları yavaş ve optimize edilmesi zor bir hale getirmektedir. Ancak, YOLOv3 nesne tespit yaklaşımı, hepsini tek bir sinir ağıyla yaparak nesne algılama ve sınıflandırma problemini çok hızlı bir biçimde çözmeyi başarmış durumdadır. Bu algoritma ilk defa 2015 yılında, görüntü işleme ve nesne tanıma konferansında sunulmuş olup, makalede, YOLOv3 algortiması şu şekilde tanımlanmıştır: "Nesne algılamayı, doğrudan görüntü piksellerinden sınırlayıcı kutu koordinatlarına ve olası sınıflara dönüştürmek suratıyla tek bir regresyon problemi olarak yeniden çerçevelendiriyoruz". Başka bir deyişle, YOLOv3 algoritması, bir girdi görüntüyü genel olarak 19x19'luk ızgaralara bölerek, bu ızgaralardan nesnelere ait olduğu tahmin edilen sınırlayıcı kutuları belirlemek ve sınıflandırmak üzere yapay sinir ağından sadece bir kere geçirmektedir. Yani, bu algoritma, bir girdi görüntüyü tek seferde işleyerek, nesnelerin konum ve sınıflarını hızlı bir biçimde tahmin edebilmektedir. Daha da basit söylemek gerekirse, girdi olarak bir görüntüyü alırsınız, onu normal bir CNN'e benzeyen bir sinir ağından geçirirsiniz ve çıktıda sınırlayıcı kutular ve sınıf tahminlerinden oluşan bir vektör elde edersiniz. YOLOv3'nun nasıl çalıştığını kavramanın ilk adımı, çıktısını nasıl kodladığını anlamaktan geçer. Birinci adımda,



Şekil 3.3. YOLOv3'da bir girdi görüntünün işlenerek piksellerden nenselerin sınıf ve sınırlayıcı kutu bilgilerini belirleme süreci.

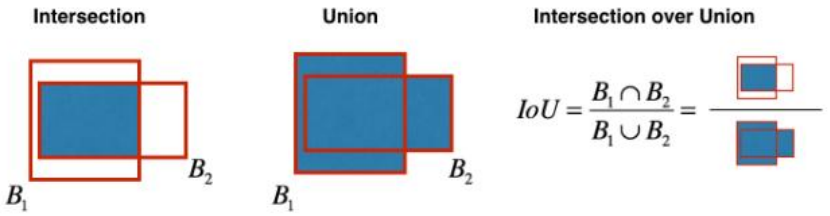
YOLOv3 görüntüyü alıyor ve görüntüyü, 608x608 veya 416x416 veya 320x320 boyutuna dönüştürüyor. Bu çalışmada, 608x608'lik girdi görüntü boyutu tercih edilmiştir. Sonrasında ise, boyutu değiştirilmiş girdi görüntüyü bir SxS hücre ızgarasına bölerek her bir ızgaradan olası nesneleri arıyor, ve nesneselliğe göre sınıfların olasılıksal değerlerini hesaplıyor, Şekil 3.3. Görüntüde bulunan her nesne için, bir ızgara hücresinin "sorumlu" olduğu belirtilmiştir. Bu ızgara, nesnenin merkezi noktalarının "sorumlu" hücre ızgarasının içine düştüğü bölgedir. Her ızgara hücresi, B sayıda sınırlayıcı kutuyu ve C sınıf olasılıklarını tahmin eder. Sınırlayıcı kutu tahmininin 6 bileşeni vardır: (sınıf, güven, x, y, w, h). (x, y) sınırlayıcı kutunun merkezi koordinatları, ızgara hücresinin konumuna



Şekil 3.4. YOLOv3 algoritmasında bir nesnenin sınırlayıcı kutularının hesaplama işlemi.

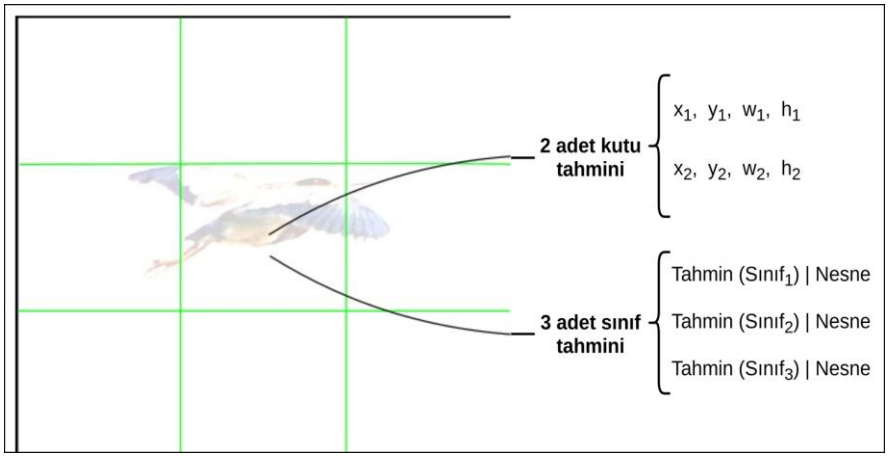
göre kutunun merkezini temsil eder. Kutunun merkezi, ızgara hücresinin içine düşmezse bu hücrenin bundan sorumlu olmadığı anlama gelir. Bu koordinatlar, 0 ile

1 arasına düşecek şekilde normalleştirilir. (w, h) kutu boyutları da görüntü boyutuna göre [0, 1] aralığında normalleştirilir. Şekil 3.4. ile bir örnek gösterilmiştir. Sınırlayıcı kutu tahmininde, güven puanı olan bir bileşen daha vardır. Makaleden: "Resmi olarak güven değerini, Tahmin (Nesne) * IOU (tahmin, gerçek) olarak tanımlıyoruz. O hücrede hiçbir nesne yoksa, güven puanı sıfır olmalıdır. Aksi takdirde, tahmin edilen kutu ile temel gerçek arasındaki güven puanının birleşim üzerinden kesişme noktasına (IOU) eşit olmasını isteriz". Yani, güven değerinin yüksek veya düşük olması; herhangi bir sınıftaki bir nesnenin varlığını veya yokluğunu yansıtmaktadır. IoU (Intersection over Union, Birleşim üzerinden Kesişim) - nesne tespit algoritmalarında kullanılan, algılanan sınırlayıcı kutuların ne kadar doğru olup olmadığını ölçebilmek için kullanılan bir yöntemdir. YOLOv3 genel olarak, IoU değerini %50'nin üzerinde tutmaktadır, Şekil 3.5.



Şekil 3.5. IoU, Birleşim üzerinden Kesişim. En solda, bir kesişim, ortadaki resimde ise bir birleşim gösterilmştir.

Artık kutu tahmininin 6 bileşenini anladığımıza göre, her bir ızgara hücresinin B tane sınırlayıcı kutuyu tahmin ettiğini (dolayısıyla, sınırlayıcı kutu tahminleriyle ilgili toplam SxSxB*5 tane çıktıyı) tartışabiliriz. Ayrıca, sınıf olasılıklarını, Tahmin(Sınıf(i) | Nesne) ifadesiyle tahmin etmek de gereklidir. Bu olasılık, bir nesne içeren ızgara hücresine koşullandırılmıştır. Uygulamada, bu, ızgara hücresinde hiçbir nesne yoksa, kayıp fonksiyonunun onu daha sonra göreceğimiz gibi yanlış bir sınıf tahmini için cezalandırmayacağı anlamına gelir. Ağ, B sayıdaki kutulardan bağımsız olarak, hücre başına yalnızca bir sınıf olasılıkları kümesini tahmin eder. Bu, toplamda SxSxC tane sınıf olasılıkları demektir. Sınıf tahminlerini çıktı vektörüne ekleyerek, çıktı olarak bir SxSx(B*5+C) tensörü elde ederiz. Bu süreç Şekil 3.6. ile gösterilmektedir. Sınırlayıcı kutu, ve sınıf tahmini bu biçimde hesaplanmaktadır YOLOv3'da.



Şekil 3.6. Sınırlayıcı kutu ve sınıf tahmin etme. Izgara (S=3), Sınırlayıcı kutu (B=2), Sınıf (C=3).

YOLOv3'da kullanılan evrişimsel sinir ağı mimarisi. YOLOv3'da özel bir CNN kullanılmamaktadır. Normal, geleneksel CNN mimarisi kullanılıyor. Yani, YOLOv3 nesne algılama mimarisi; giriş katmanı, evrişim katmanları, min, max veya average pooling, ve iki tam bağlı softmax sınıflandırma katmanı gibi bileşenlerden oluşmaktadır. Bu katmanların tamamı YOLOv3'nin küçük mimarisi için 23 katmandan, büyük mimarisi için ise 106 katmandan oluşmaktadır. Şekil 3.7.'de YOLOv3'nun büyük mimarisi gösterilmektedir. Bu mimari, YOLOv3 algoritmasının geliştiricilerinin S = 7, B = 2 ve C = 20 olarak kullandığı Pascal VOC veri setinde kullanılmak üzere tasarlanmıştır. Bu, son özellik haritalarının neden 7x7 olduğunu ve ayrıca çıktının boyutunu (7x7x(2 *5+20)) açıklar. Bu ağın, farklı bir ızgara boyutuyla veya farklı sayıda sınıfla kullanılması, katman boyutlarının ayarlanmasını gerektirebilir. Yazarlar, YOLOv3'nun daha az evrişimli katmana sahip hızlı bir versiyonu olduğundan bahsetmektedir. Ancak, aşağıdaki şekil tam sürümü göstermektedir. 1x1 azaltma katmanları ve 3x3 evrişimli katman dizileri, GoogLeNet (Inception) modelinden esinlenmiştir. Son katman, doğrusal bir etkinleştirme işlevi kullanır. Diğer tüm katmanlar bir LEAKY RELU aktivasyon fonksiyonunu kullanır ($\Phi(x) = x$, $x > 0$ ise; $0.1x$ aksi halde).

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   320 x 320 x   3   ->   320 x 320 x  32  0.177 BFLOPs
    1 conv     64  3 x 3 / 2   320 x 320 x  32   ->   160 x 160 x  64  0.944 BFLOPs
    2 conv     32  1 x 1 / 1   160 x 160 x  64   ->   160 x 160 x  32  0.105 BFLOPs
    3 conv     64  3 x 3 / 1   160 x 160 x  32   ->   160 x 160 x  64  0.944 BFLOPs
    4 res    1                 160 x 160 x  64   ->   160 x 160 x  64
    5 conv    128  3 x 3 / 2   160 x 160 x  64   ->    80 x  80 x 128  0.944 BFLOPs
    6 conv     64  1 x 1 / 1    80 x  80 x 128   ->    80 x  80 x  64  0.105 BFLOPs
    7 conv    128  3 x 3 / 1    80 x  80 x  64   ->    80 x  80 x 128  0.944 BFLOPs
    8 res    5                  80 x  80 x 128   ->    80 x  80 x 128
    9 conv     64  1 x 1 / 1    80 x  80 x 128   ->    80 x  80 x  64  0.105 BFLOPs
   10 conv    128  3 x 3 / 1    80 x  80 x  64   ->    80 x  80 x 128  0.944 BFLOPs
   11 res    8                  80 x  80 x 128   ->    80 x  80 x 128
   12 conv    256  3 x 3 / 2    80 x  80 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   13 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   14 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   15 res   12                  40 x  40 x 256   ->    40 x  40 x 256
   16 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   17 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   18 res   15                  40 x  40 x 256   ->    40 x  40 x 256
   19 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   20 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   21 res   18                  40 x  40 x 256   ->    40 x  40 x 256
   22 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   23 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   24 res   21                  40 x  40 x 256   ->    40 x  40 x 256
   25 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   26 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   27 res   24                  40 x  40 x 256   ->    40 x  40 x 256
   28 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   29 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   30 res   27                  40 x  40 x 256   ->    40 x  40 x 256
   31 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   32 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   33 res   30                  40 x  40 x 256   ->    40 x  40 x 256
   34 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
   35 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
   36 res   33                  40 x  40 x 256   ->    40 x  40 x 256
   37 conv    512  3 x 3 / 2    40 x  40 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   38 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   39 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   40 res   37                  20 x  20 x 512   ->    20 x  20 x 512
   41 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   42 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   43 res   40                  20 x  20 x 512   ->    20 x  20 x 512
   44 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   45 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   46 res   43                  20 x  20 x 512   ->    20 x  20 x 512
   47 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   48 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   49 res   46                  20 x  20 x 512   ->    20 x  20 x 512
   50 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   51 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   52 res   49                  20 x  20 x 512   ->    20 x  20 x 512
   53 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
```

Şekil 3.7. YOLOv3 algoritmasında kullanılan CNN mimarisi.

```
   53 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   54 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   55 res   52                  20 x  20 x 512   ->    20 x  20 x 512
   56 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   57 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   58 res   55                  20 x  20 x 512   ->    20 x  20 x 512
   59 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   60 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   61 res   58                  20 x  20 x 512   ->    20 x  20 x 512
   62 conv   1024  3 x 3 / 2    20 x  20 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   63 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   64 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   65 res   62                  10 x  10 x1024   ->    10 x  10 x1024
   66 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   67 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   68 res   65                  10 x  10 x1024   ->    10 x  10 x1024
   69 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   70 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   71 res   68                  10 x  10 x1024   ->    10 x  10 x1024
   72 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   73 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   74 res   71                  10 x  10 x1024   ->    10 x  10 x1024
   75 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   76 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   77 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   78 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   79 conv    512  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 512  0.105 BFLOPs
   80 conv   1024  3 x 3 / 1    10 x  10 x 512   ->    10 x  10 x1024  0.944 BFLOPs
   81 conv    255  1 x 1 / 1    10 x  10 x1024   ->    10 x  10 x 255  0.052 BFLOPs
   82 yolo
   83 route  79
   84 conv    256  1 x 1 / 1    10 x  10 x 512   ->    10 x  10 x 256  0.026 BFLOPs
   85 upsample            2x    10 x  10 x 256   ->    20 x  20 x 256
   86 route  85 61
   87 conv    256  1 x 1 / 1    20 x  20 x 768   ->    20 x  20 x 256  0.157 BFLOPs
   88 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   89 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   90 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   91 conv    256  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 256  0.105 BFLOPs
   92 conv    512  3 x 3 / 1    20 x  20 x 256   ->    20 x  20 x 512  0.944 BFLOPs
   93 conv    255  1 x 1 / 1    20 x  20 x 512   ->    20 x  20 x 255  0.104 BFLOPs
   94 yolo
   95 route  91
   96 conv    128  1 x 1 / 1    20 x  20 x 256   ->    20 x  20 x 128  0.026 BFLOPs
   97 upsample            2x    20 x  20 x 128   ->    40 x  40 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    40 x  40 x 384   ->    40 x  40 x 128  0.157 BFLOPs
  100 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
  101 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
  102 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
  103 conv    128  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 128  0.105 BFLOPs
  104 conv    256  3 x 3 / 1    40 x  40 x 128   ->    40 x  40 x 256  0.944 BFLOPs
  105 conv    255  1 x 1 / 1    40 x  40 x 256   ->    40 x  40 x 255  0.209 BFLOPs
  106 yolo
Loading weights from ./yolov3.weights...Done!
```

Şekil 3.7. (Devamı).

$$L_{loc} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 + (\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \right]$$

$$L_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} (1_{ij}^{\text{obj}} + \lambda_{\text{noobj}}(1 - 1_{ij}^{\text{obj}}))(C_{ij} - \widehat{C_{ij}})^2 + \sum_{i=0}^{S^2} \sum_{c \in C} 1_i^{\text{obj}} (p_i(c) - \hat{p_i}(c))^2$$

$$L = L_{\text{loc}} + L_{\text{cls}}, \tag{3.2}$$

Burada, 1_i^{obj} i.ızgara hücresinin herhangi bir nesne içerip içermediğini gösteren bir

gösterge işlevi. 1_{ij}^{obj}, i'inci ızgara hücresinin, j'inci sınırlayıcı kutusunun herhangi bir nesne tahmininden "sorumlu" olup olmadığını gösterir. C_{ij}i'inci ızgara hücresinin güven puanı, Tahmin (bir nesne içerirmi?) * IoU (tahmin, gerçek) formülü ile ifade edilir. Bu çalışma için IoU>0.5 olarak ayarlanmıştır. $\widehat{C_{ij}}$öngörülen güven puanı. C ise tüm sınıfların kümesi. $p_i(c)$ i'inci ızgara hücresinin c∈C sınıfından bir nesne içerip içermediğinin koşullu olasılığı. $\widehat{p_i}(c)$öngörülen koşullu sınıf olasılığı.

YOLOv3n'nun eğitim süreci için kullandığı hata fonksiyonu çok karmaşık bir yapıya sahipmiş gibi görünür ilk bakışta, Denklem 3.2. Çünkü, hata fonksiyonu üç büyük hata formülünden oluşur. Başka bir deyişle, YOLOv3'da, bir girdi görüntü, tek bir yapay sinir ağından tek seferde geçirilerek, üç ana bilgi çıkarmak üzere işlenir. Bu bilgiler; girdi görüntüde herhangi bir nesnenin mevcut olup olmadığı bilgisi, varsa o nesne(ler)nin sınıf bilgileri, ve nesneyi en iyi çevreleyen sınırlayıcı kutu(ları)nun konumu ve büyüklük değerlerini içermektedir. Dolayısıyla, bu üç ana bilgi için üç hata fonksiyonu kullanılmıştır. İlk olarak, bir girdi görüntüde nesnenin konumu ve büyüklüğünü kontrol eden hata hesaplama formülünü inceleyelim, Denklem 3.3.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2], \tag{3.3}$$

Bu denklem, tahmin edilen sınırlayıcı kutu konumu (x ve y) ile ilgili kaybı hesaplar. Şimdilik λ için endişelenmeyelim, bunu belirli bir sabit olarak kabul edelim. Fonksiyon, her ızgara hücresinin (i = 0 .. S^2) her bir sınırlayıcı kutu tahmini (j = 0 .. B) üzerinden bir toplam hesaplar. 1^{obj} şu şekilde tanımlanır: Eğer i'inci ızgara hücresinde bir nesne varsa, j'inci sınırlayıcı kutu öngörücüsü bu tahmini doğruluyorsa, 1, aksi takdirde 0'dır. Fakat bu nesneden hangi öngörücünün sorumlu olduğunu nasıl bileceğiz sorusuna YOLOv3 geliştiricileri şu şekilde açıklama getirmişler: "YOLOv3, ızgara hücresi başına birden çok sınırlayıcı kutu öngörür. Eğitim zamanında, her nesneden yalnızca bir sınırlayıcı kutu öngörücüsünün sorumlu olmasını istiyoruz. Temel gerçeğe sahip en yüksek mevcut IoU'ya sahip tahminin bir nesneyi tahmin etmekten "sorumlu" olması için bir tahminci atarız." Denklemdeki diğer terimlerin anlaşılması kolaydır: (x, y) tahmin edilen sınırlayıcı kutu konumudur

ve şapkalı $(\hat{x}, \hat{y})$ eğitim verilerinden alınan gerçek konumdur. Şimdi, sınırlayıcı kutuların eni ve boyu için kullanılan hata fonksiyonunu inceleyelim, Denklem 3.4.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} [(\sqrt{w_i} - \sqrt{\widehat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\widehat{h_i}})^2], \tag{3.4}$$

Bu denklemde gösterilen ifade, tahmin edilen kutu genişliği/yüksekliğiyle ilgili kayıp fonksiyonudur. Denklem, karekök dışında ilkine benzer. Burada neden karekök kullanılmıştır sorusuna, makaleden tekrar alıntı yaparak cevaplayalım: "Hata ölçütümüz, büyük kutulardaki küçük sapmanın küçük kutulardakinden daha az önemli olduğunu yansıtmalıdır. Bunu kısmen ele almak için, doğrudan genişlik ve yükseklik yerine sınırlayıcı kutu genişliğinin ve yüksekliğinin karekökünü tahmin ediyoruz". Yani, küçük ve büyük boyutlu nesnelerin sınırlayıcı kutularını tahmin etmede doğruya en yakın bir şekilde yaklaşabilmek için karekök kullanılmış. Fakat, doğal olarak, aşırı küçük ve büyük nesnelerde bu çözüm işe yaramamaktadır. Üçüncü, son kısıma geçelim, Denklem 3.5.

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} (1_{ij}^{\text{obj}} + \lambda_{\text{noobj}}(1 - 1_{ij}^{\text{obj}}))(C_{ij} - \widehat{C_{ij}})^2 + \sum_{i=0}^{S^2} \sum_{c \in C} 1_i^{\text{obj}} (p_i(c) - \widehat{p_i}(c))^2, \tag{3.5}$$

Burada, her bir sınırlayıcı kutu öngörücüsü için güven puanıyla ilişkili kaybı hesaplıyoruz. C güven puanıdır ve $\hat{C}$, öngörülen sınırlayıcı kutunun temel gerçek ile birleşimidir. 1^{obj}, hücrede bir nesne olduğunda bire, aksi takdirde 0'a eşittir. 1^{noobj} ise tam tersidir. Burada ve ayrıca ilk bölümde görünen λ parametreleri, kayıp fonksiyonlarının parçalarını farklı şekilde ağırlıklandırmak için kullanılır. Model kararlılığını artırmak için bu gereklidir. En yüksek ceza koordinat tahminleri için ($\lambda_{\text{coord}} = 0.5$) ve en düşük ceza ise hiçbir nesne olmadığında ($\lambda_{\text{noobj}} = 0.5$) güven tahminleri içindir. 1^{obj} terimi haricinde, sınıflandırma için normal bir toplam kare hatasına benzer. Bu terim, hücrede herhangi bir nesne bulunmadığında sınıflandırma hatasını cezalandırmak için kullanılır (bu nedenle daha önce koşullu sınıf olasılığı tartışılmıştır). Genel olarak, eğitim süreci şu şekildedir: İlk olarak, 224x224 giriş boyutunu kullanarak ImageNet 1000 sınıfı veri kümesini kullanarak ilk 20 evrişimli

katmanı önceden eğitilir. Ardından, giriş çözünürlüğünü 448x448'e yükseltilir. Toplu iş boyutu 64, momentum 0,9 ve bozulma 0,0005 olarak yaklaşık 135 dönem (epoch) boyunca tüm ağ eğitilir. Öğrenme oranı çizelgesi: ilk adım epochlar için, öğrenme oranı yavaşça 0.001'den 0.01'e yükseltilir. Yaklaşık 75 dönem eğitilen sistemin bu değerleri azaltılır. Rastgele ölçeklendirme, resim döndürme, pozlama ve doygunluk gibi rastgele işlemlerle veri artırmı yapılır. Makalede, önerilen bu metotları kullanarak, hazırlanmış eğitim setleri ile kavşaklar için ayrı, karayolları için ayrı modeller eğitilimiştir. Bu proje için veri kümesi hazırlama ve eğitim süreçleri aşağıda ayrıntılı olarak beyan edilmiştir.

Şekil 3.8. Bir kavşak görüntüsünün etiketlenmiş hali. Üst resim, görüntüyü, alt resim ise etiket bilgilerini içeren metin dosyayı temsil eder.

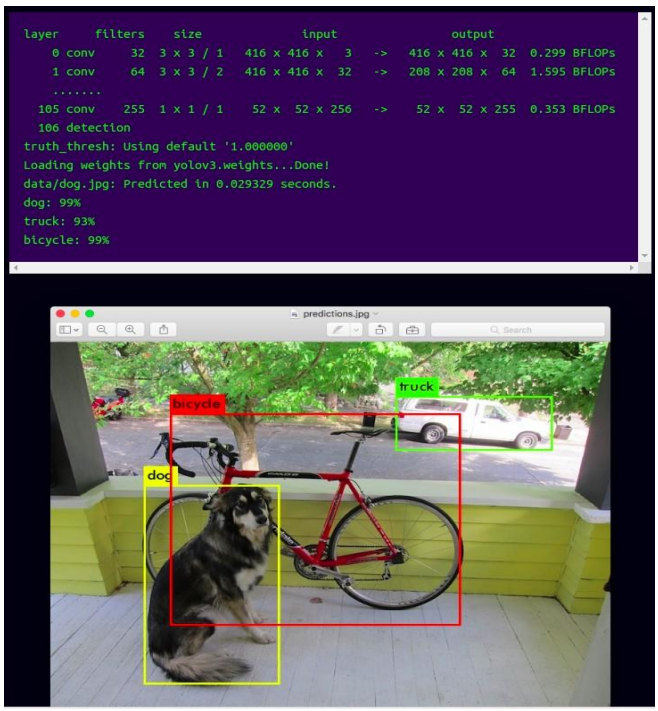
Kavşak ve otoyollar için veri kümesi hazırlama. Kavşaklar için hazırlanan veri kümesi, dünyanın muhtelif bölgelerinde yer alan birçok kavşaktan toplanan video

görüntülerinden derlenmiştir. Toplanan video görüntülerinden, beş kavşağa ait olan kısmı test verisi olarak ayılmıştır, kalanları eğitim için kullanılmıştır. Veri kümesi hazırlamada en önemli görev; görüntülerdeki araçları (nesneleri) iyi bir biçimde etiketlemektir. Bu görevi elle yapmak çok uzun zaman gerektirmektedir. Dolayısıyla, zamandan tassaruf etmek amaçlı transfer learning metodu kullanılmıştır. Bu metoda göre, eğitim için kullanılacak video görüntüler her saniyeden bir çerçeve olarak çıkarılır ve eğitim seti olarak kaydyedilir. Sonra bu çerçevelerin her biri YOLOv3 algoritmasının genel nesne tespit amaçlı eğitilmiş modelinde çalıştırılır ve güven değeri %25'lik nesne algılama eşik değerinin üstünde tespit edilen araçlar otomatik olarak etiketlenir. Otomatik olarak etiketlenmiş her bir görüntü aynı dosya ismi ile .jpg ve görüntü içindeki araça ait sınıf ve sınırlayıcı kutu bilgileri ise .txt olarak kaydedilir. Sonra bu kaydedilen veri kümesi, MİT ünversitetinin araştırmacıları tarafından geliştirilen labelImg açık kaynaklı nesne etiketleme yazılımı ile sıkı bir şekilde incelenip elden geçirilecektir. Aşağıdaki şekilde bir görüntü üzerinde oluşturulan .jpg ve .txt dosyaları gösterilmektedir. Karayollar için hazırlanmış veri seti de aynı kavşaklarda olduğu gibi bir çok otoyol görüntülerinden elde edilen verilerden oluşmaktadır. Kavşaklarda izlenen hazırlama süreci aynı otoyollar içinde geçerlidir. Eğitim setleri hazırlandıktan sonra, eğitim süreci başlatılacaktır. YOLOv3 algoritmasını çalıştırılabilmesi için C programlama dilinde evrişimsel sinir ağı mantığına dayalı geliştirilmiş darknet kütüphanesi kurulmalıdır. Bu kütüphane yüksek hesaplama başarımlı bir ortamda paralel olarak çalıştırılacağından dolayı, işlemin yapılacağı bilgisayarda en az 4GB hafızalı ekran kartına ihtiyaç duyulmaktadır. Bunun dışında, bilgisayara Linux işletim sistemi kurulmuş olması ve Nvidia tabanlı CUDA paralel hesaplama kütüphanesinin >9.0 ve CuDNN kütüphanesinin ise >7.0 versiyonları kurulmuş olması lazımdır.

```
[ ]  1  git clone https://github.com/pjreddie/darknet
     2  cd darknet
     3  make
     4  ./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

Şekil 3.9. YOLOv3 nesne tespit sisteminin kurulum komutları.

Gerekli tüm kütüphaneler kurulduktan sonra darknet kütüphanesi kurulacaktır. Darknet kütüphanesi, https://pjreddie.com/darknet/yolov3/ adresinden Şekil 3.9.'da gösterilen komutları linux terminalinde çalıştırılarak kurulur. Sonrasında ise, kurulumu yapılan Darknet klasörüne YOLOv3'nun genel nesne tespiti için hazır eğitilmiş ağırlık modeli https://pjreddie.com/media/files/yolov3.weights adresinden indirilir. Ve YOLOv3 algoritması test için hazır hale gelir. Herhangi bir girdi görüntü Şekil 3.9.'da gösterilen komutların 4. satırındaki komut ile işlenebilmektedir.



Şekil 3.10. YOLOv3 görüntü işleme çıktısı.

Şekil 3.9.'da gösterilen komutların 4. satırındaki yolov3.cfg dosyası; YOLOv3 yapay sinir ağının konfigürasyon bilgilerini taşıyan dosyadır, yolov3.weights adından da anlaşılacağı üzere ağırlık model dosyasıdır. Sonundaki data/dog.jpg ifadesi bir girdi görüntü ve konumunu belirtmektedir. YOLOv3, genel olarak resimlerde kullanılmaktadır. Video dosyasını direk yüklemek ve işeleme alabilecek API fonksiyonu bulunmamaktadır, bu sebeple bu fonksiyonu kendimiz geliştirdik. Yukarıdaki kodlar eksiksiz bir şekilde çalıştırıldığı vakit, aşağıdaki Şekil 3.10.'daki gibi sonuçlar elde edilebilecektir. Girdi bir video dosyasını işleme fonksiyonu Python dilinde geliştirilmiştir ve aşağıda Şekil 3.11. ile gösterilmektedir. Bu fonksiyonu,

darknet klasöründeki python adlı klasörde bulunan darknet.py adlı YOLOv3'nun ana dosyasına eklenmesi gerkemektedir. Sonrasında direk, "import darknet as dn" deyip, herhangi bir programda "dn.detect_image()" tarzında API olarak kullanılabilmektedir. YOLOv3 tabanlı araç tespit algoritmamız bu şekilde hazırlanmaktadır. Video dosylarını okuma ve girdi olarak verilen video dosyasından çerçeveleri çıkarma gibi bazı ön işleme işlemlerini OpenCV kütüphanesi kullanılarak gerçekleştirilmiştir. Araç algılama modülü, giriş verisi olarak bir video dosyasının yolunu veya canlı video akışının İP adresini veya bir Youtube video dosyasının URL adresini almaktadır. Çıktı olarak, her bir video çerçevesindeki mevcut nesnelerin sınıfını, sınıfa ait olan güven değeri ve konum olarak sınırlayıcı kutu bilgilerinden oluşan iki boyutlu diziyi döndürmektedir. Bu dizi ardışık video çerçevelerindeki nesneleri takip edebilmek amaçlı nesne takip algoritmasının girdileri olarak kullanılmaktadır. Sırada nesne takip algoritmaları nasıl geliştirildiğinden bahis edilecektir.

```
def detect_image(net, meta, image, thresh=.5, hier_thresh=.5, nms=.45):

    im, img = array_to_image(image)
    rgbgr_image(im)
    num = c_int(0)
    pnum = pointer(num)
    predict_image(net, im)
    dets = get_network_boxes(net, im.w, im.h, thresh,
                             hier_thresh, None, 0, pnum)
    num = pnum[0]
    if nms: do_nms_obj(dets, num, meta.classes, nms)

    res = []
    for j in range(num):
        a = dets[j].prob[0:meta.classes]
        if any(a):
            ai = np.array(a).nonzero()[0]
            for i in ai:
                b = dets[j].bbox
                res.append((meta.names[i], dets[j].prob[i], (b.x, b.y, b.w, b.h)))

    res = sorted(res, key=lambda x: -x[1])

    if isinstance(img, bytes):
        free_image(im)
        free_detections(dets, num)

    return res
```

Şekil 3.11. YOLOv3 video çerçevesi işleme fonksiyonu. "darknet.py" ana dosyaya eklenecek fonksiyondur.

3.3. Kalman Filtersi ve Merkezi Nokta Tabanlı Araç Takip Algoritmaları

Araç (nesne) takip etme işlemi, bir aracı, bir video akımının ard arda gelen çerçevelerinde yegane kimlik atayarak izlenmesidir. Yani, bir videonun belli bir çerçevesinde ilk kez görünen arabaya rakam olarak yegana kimlik ataması ve bu arabayı ta video sahnesini terk ettiği çerçeveye kadar aynı kimlikle takip edilmesidir. Bu problem görünüşte çok sade, basitmiş gibi görünsede geçekte zor bir görüntü işleme problemidir. Çünkü, video çekme esnasında kamerların titremesi, yağmur, kar ve sis gibi zorlu hava şartları, nesnelerin birbirini yarı veya tam olarak örtmesi, anlık veya kalıcı parlaklıklar, gece gündüz ışık değişimi problemi gibi bir takım sorunlardan dolayı hem nesne tespit hem de nesne takip etme algoritmaları doğru çalışamamaktadır, bazen ise tamamen başarısız olabilmektedir. Bu tür sorunların hafifletilmesi veya tamamen ortadan kaldırlıması amaçlı Kalman filtresi tabanlı ve merkezi nokta tabanlı nesne takip algoritmaları geliştirilmiştir. Bu algoritmaların ikiside nesne tespit sonuçlarına doğrudan bağlıdırlar. Çünkü, bu algoritmaların girdisi direk YOLOv3 araç tespit algoritmasının çıktılarıdır. Aşağıda, Kalman filtresine dayalı nesne takip ve merkezi nokta tabanlı nesne takip algoritmaları detaylı olarak anlatışmıştır.

3.3.1. Kalman filtresine dayalı araç takip yaklaşımı

Bu bölümde, Kalman Filtresine dayalı araç takip yaklaşımı için bir sistem modelini açıklıyoruz. Kalman Filtresi, genel olarak bir önceki veya mevcut durumu girdi olarak alır ve sistemin belirsiz ölçümleri ve mevcut durumunu göz önünde bulundurarak bir sonraki adımı tahmin eder. Daha sonra, yeni adım için yapılan tahmini, o adım için aldığı girdi bilgiler ile karşılaştırır. Karşılaştırmada eşleşen bilgilerin durumları yenilenir, hata durumlar ise düzeltilir. İlk olarak, birinci adımın durumu Kalman filtresi için tanımlanması gerekir. Nesnenin mevcut bilgisine dayanarak hareket eden bir nesnenin konumunu tahmin etmek istediğimiz sebebli, problemimiz için sabit bir hız modeli varsayıyoruz. O halde, tek boyutlu hareket eden bir nesnenin dinamikleri şu şekilde tanımlanabilir, Denklem 3.6.

$$x_t = \frac{1}{2}a,\Delta t^2 + v_{t-1}\Delta t + x_{t-1}, \tag{3.6}$$

$$v_t = v_{t-1} + a\Delta t, \tag{3.7}$$

Burada, x_t – t. inci çerçeve için yenı tahmin edilen durum, a – ivme, Δt – zaman, v_t – t. inci çerçeve için anlık hız, x_{t-1} ve v_{t-1} bir önceki çerçevedeki nesnelerin konum ve hız durumu. Yani, bir boyutlu uzayda hareket eden bir nesnenin dinamiği, Denklem 3.6'deki gibi konum denklemi, ve Denklem 3.7'deki gibi konumdan türetilmiş birinci dereceden türevi ile ifade edilebilmektedir. Bu genelleme çerçevesinde, bir boyutlu uzayda hareket eden nesnenin durum denklemini iki boyutlu bir uzaya uyarlayarak genişlettiğimizde, bir nesnenin iki boyutlu bir uzaydaki dinamik durumu Denklem 3.8'deki gibi oluşturabiliriz.

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} \tag{3.8}$$

Bir önceki duruma göre cari durum, veya cari duruma göre bir sonraki yeni durum nasıl tahmin edileceğini açıklayalım.

$$x_t = x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}a\Delta t^2, \tag{3.10}$$

$$y_t = y_{t-1} + \dot{y}_{t-1}\Delta t + \frac{1}{2}a\Delta t^2, \tag{3.11}$$

$$\dot{x}_t = \dot{x}_{t-1} + a\Delta t, \tag{3.12}$$

$$\dot{y}_t = \dot{y}_{t-1} + a\Delta t, \tag{3.13}$$

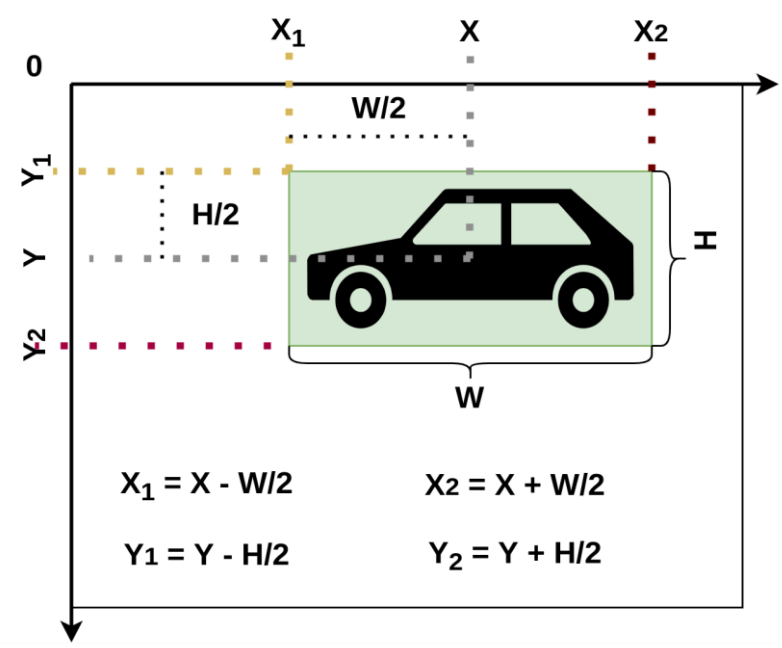
Bu denklemleri, Denklem 3.8'deki yerlerine getirip konulduğunda, bir objenin genel dinamik durum denklemi elde edilir, Denklem 3.14.

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t^2/2 \\ \Delta t^2/2 \\ \Delta t \\ \Delta t \end{bmatrix} \cdot a + W_{t-1}, \tag{3.14}$$

Sonuç olarak, Kalman filtresi tabanlı araç takip formülü oluşturulmuş oldu, Denklem 3.15.

$$X_t = AX_{t-1} + Bu_{t-1},\ Y_t = AY_{t-1} + Bu_{t-1}\ ,$$
$$W_t = AW_{t-1} + Bu_{t-1}, H_t = AH_{t-1} + Bu_{t-1}\ , \quad (3.15)$$

Bilindiği üzere, YOLOv3 algoritmasına dayalı tespit algoritması [[sınıf, güven değeri, [X, Y, W, H]], …] gibi vektör sonuçları üretir. Burada, X – bir çerçevedeki tespit edilen bir nesnenin sınırlayıcı kutusunun merkezi x noktasıdır, Y ise merkezi y nokta, W ve H değerler ise sınırlayıcı kutunun eni ve boyudur. Kalman Filtresi girdi olarak, bir çerçevedeki tüm nesnelerin girdilerini, [[X1, Y1, X2, Y2], …] gibi konum tarzında almaktadır. Dolaysıyla, araç algılama sonuçlarının formatı araç takip formatine dönüştürülması gerekir. Dönüştürme işlemi Şekil 3.12. ile ayrıntılı olarak gösterilmiştir. X_1, Y_1, X_2, Y_2 durumların tamamı Denklem 3.15'e göre tahmin edilemektedir. Girdi olarak, [[X1, Y1, X2, Y2], …] vektörü alan Kalaman filtresi, çıktı olarak [[0, [X1, Y1, X2, Y2]], [1, [X1, Y1, X2, Y2]], …, [n, [X1, Y1, X2, Y2]]] şeklinde çıktı döndürür. 0, 1 ve n değerleri, araçlara (nesnelere) yegane olarak atanan kimliklerdir.



Şekil 3.12. YOLOv3 sınırlayıcı kutu değerlerinin koordinatlara dönüşümü.

3.3.2. Merkezi nokta tabanlı araç takip yaklaşımı

Araçların algılanan sınırlayıcı kutularının merkezi nokta özelliklerine göre araç takibi yapan merkezi nokta tabanlı araç takibi algoritması, Kalman filtresine dayalı nesne izleme yaklaşımı gibi bir lineer sistem problemidir. Bu yaklaşımda, nesne tespit biriminin üretmiş oluduğu sınırlayıcı kutu bilgileri (nesnelerin konum ve sınıf özellikleri) göz önünde bulundurularak araçlar takip edilir. Bu araç takip etme algoritması; araç algılama, aynı konumda tespit edilen nesneleri için çoklu tespit eleme, oylama, ve kamera titreşim filtrelemesi, yegane kimlik atama ve nesnelerin konum tahmini olmak üzere beş adımdan oluşmaktadır. Araç algılama biriminde, YOLOv3 gibi sağlam, nesne konumunu yüksek doğrulukla belirleyebilen ve hızlı çalışabilen algoritma, kavşak ve otoyol görüntüleri ile tekrar eğitilmiş olmasına rağmen, araçların birbirine yakın veya benzer öznitelikleri sebebiyle araçların sınıflandırmasında sık sık yanılsama meydana gelmektedir. Bunun dışında, aynı konumdaki bir araba; hem araba, hem kamyon ve hem de otobüs olmak üzere üç araç varmış gibi algılanabiliyor. Bu durum, tamamen YOLOv3'dan kaynaklan maktadır.

Algoritma 1. Aynı konumda tespit edilen çoklu sınırlayıcı kutuları eleme.

```
1:  Araçların sınırlayıcı kutu bilgilerini içeren, boş olmayan küme – S, n – sınırlayıcı
    kutu sayısı, silinecek sınırlayıcı kutuların kimlik listesi – Ss, çoklu sınırlayıcı
    kutulardan arındırılmış yeni liste – Sy. Bir sınırlayıcı kutu şu bilgileri içermektedir –
    S[0] =[sınıf, güven, [x,y,w,h]]
2:  function duplication_check(S)
3:    for i ← 0 to n do
4:      for j ← i+1 to n do
5:        S_iou ←sınırlayıcı_kutu_iou (S_i,2, S_j,2)
6:        if S_iou >= 0.95 then
7:          if S_i,1 >= S_j,1 then
8:            Ss ←i
9:    for i ← 0 to n do
10:     if i not in Ss then
11:       Sy ← i
12:   return Sy
13: end function
```

Algoritma 2. Araç takip algoritmasının birinci katmanı - oylama filtresi.

Araçların sınırlayıcı kutu bilgilerini içeren, boş olmayan küme – S, n – sınırlayıcı kutu sayısı, Th – çerçeve eşik değeri, Ag – geçici kimlikli araç listesi, m – geçiçi kimlikli araçların sayısı

function voter(*S, frame_id***)**

  if *frame_id = 0* **then**

    vote ←0

    for *i ← 0* ***to*** *n* **do**

      vote ←vote + 1

      Ag ←(S_i *, i, vote)*

  else

    m ← length(Ag)

    for *i ← 0* **to** *n* **do**

      if S_i **not in** *Ag* **then**

        vote ←vote + 1

        Ag ← (S_i *, i, vote)*

      else

        $Ag_{i,2}$ *←*$Ag_{i,2}$ *+ 1*

  return Ag

end function

F - Otuz çerçeve içerdiği nesneler ile çerçeveler listesi. Bu çalışmada, genel olarak, bir saneyelik video çerçeveler alınmıştır. Frame id: i ←0. Fn – oylamadan geçirilmiş yeni liste.

function voting_filter(F)

  while *i <= 30* **then do**

    Fn ← voter(F_i*, i)*

    i ← i + 1

  Fn ←max{Fn}

  return Fn

  end of function

YOLOv3 ne kadar iyi eğitilmiş olsa dahi, aynı konumda çoklu nesne tespiti kaçınılmaz bir sorundur. Diğer yandan, bir araba hem araba hem de kamyon olarak, bir otobüs ise hem kamyon hem araba olarak iki veya üç nesne varmış gibi tespit

edilebiliyor. Aynı konumda, ikili, üçlü veya çoklu araç algılama sorununu gidermek amaçlı, merkezi nokta tabanlı takip algoritmasına “duplication check and eliminate”, yani, aynı konumdaki çoklu algılamaları giderici bir modül eklenmiştir. Bu modül; aynı konumda tespit edilen çoklu nesnelerin en yüksek güven değerli olanını seçip kalanlarını eleme mantığına dayanmaktadır. Bu maksatle, “intersection over union” yaklaşımı kullanılmıştır. Konumları yaklaşık veya kesin olarak aynı olan nesnelerin kapladıkları alanlar birbiriyle %95’ten fazla örtüşüyorsa, bu aynı konumdaki nesneler kontrol edilerek eleme işlemi yapılmaktadır. Bu işlem, Algoritma 1’deki sözde kod ile ayrıntlı olarak gösterilmiştir.

Aynı nesneyi; çoklu, üçlü veya ikili olarak yanlış tespit etme sorununun dışında, bir araç ard arda gelen farklı çerçevelerde farklı sınıflara dahil edilebilmektedir. Bu problem de araç takip algoritmalarını büyük ölçüde yanlış pozitif sonuçlar üretmesine neden olmaktadır. Yani, bir çerçevede görünen araba sonraki çerçevede ya kamyon veya otobüs olarak yanlış tespit edilebilmektedir. Bu sorunu giderebilmek amaçlı nesne takip algoritmamıza birinici katman olarak oylama filtresi eklenmiştir. Oylama katman filtresi, belirlenen eşik sayısındaki çerçeveleri inceleyerek bir objenin gerçek sınıfını belirlemeye çalışmaktadır. Mesala, eşik değerini 30 çerçeve olarak belirlendiği zaman, sahneye giren her bir yeni araç, kendisine kalıcı yegane bir kimlik atanmadan önce bu otuz çerçeve boyunca gözlemlenmektedir. Bu şekilde, nesne takip algoritmamız bir nesneyi gerçek kalıcı listesine yegane kimlik atayarak dahil etmeden önce geçici kimlikle otuz çerçeve boyunca gözlemlemekte ve bu 30 çerçevede gözlemlenen nesne kaç kere araba, kaç kere kamyon ve kaç kez otobüs olarak tespit edildiği ile ilgili oyları saymaktadır. Netice olarak en çok oy alan sınıf adı gözlemlenen nesnenin sınfını belirlemekte, ve bu nesneye, sahneye girdiği andan itibaren sahneyi terk edinceye kadar değişmeyecek olan yegane bir kalıcı kimlik atanmaktadır. Başka bir deyişle, sınıfı belirlenen nesne, kendisine benzersiz kimlik atanarak, geçici listeden silnip, araç takip biriminin kalıcı listesine dahil edilmektedir. Böylece, kalıcı listeye alınan araç ta sahneden kaybolana kadar aynı kimlikle takip edilerek iz bilgileri çıkartırlmakatdır. Bu oylama algoritmasının sözde kodu, Algoritma 2 ile gösterilmektedir.

Çoklu ve yanlış pozitif tespitlerden arındırılmış araçlar; sahneye girdikleri andan itibaren sahneyi terk edene kadar ki olan zaman dilimi boyunca takip edilerek nesne izleri çıkarılmaktadır. Bu işlem araçları kalıcı listeye dahil etme ve ard arda gelen çerçevelerde bu aynı nesneleri birbirleriyle ilişkilendirme ve sahneyi terk eden doğru araçları derhal kalıcı listeden çıkarma olarak üç modülden oluşmaktadır. Algoritma 3 ile bu süreç ayrıntılı bir biçimde gösterilmiştir.

Algoritma 3. Araçları, kalıcı listeye ekleme, listeye eklenen araçları çerçeveler boyunca doğru bir biçimde ilişkilendirme, bazı araçların sahneden geçici olarak kaybolduğu veya nesne tespit aracının tespitinden kaynaklı sorunlar dolayısıyla sahnede görülmemesi durumunda araçların durumunu tahmin etme, ve sahneden gerçekte çıkan araçları derhal kalıcı listeden silme ve araçların izini çıkarma modülleri.

Araçların sınırlayıcı kutu bilgilerini içeren, boş olmayan küme – S, n – sınırlayıcı kutu sayısı, sınırlayıcı kutuların kimlik listesi – Ids, araçların çizdiği iz bilgileri – Coors. Araçların kaybolma ve görünme sayısını tutan listeler – K ve G. Kaybolan aracı azami kaç çerçeveye kadar takip edilecek, bu bilgiyi tutan yaş değeri – max_yas ←100. Vn, Vn_1 – n.ci ve n-1.ci çerçevede tespit edilmiş nesneler, N – yeni, P – bir önceki eski çerçevelerdeki araçların kimliklerini tutan listeler. Yegane kimlik için başlangıç değer; id ← 0.

function vehcile_tracker (*S, fr_id*)
  if *fr_id = 0* **then do**
    $S_{n-1} \leftarrow S$
    register(S_{n-1})
  else
    $S_n \leftarrow S$
    register(S_n)
    V_n = *association*(S_n, S_{n-1})
  return Vn //tracked_vehicles
end function

function register(bbox)
  Ids ← *Ids* U {id}
  Coors ← *Coors* U *{bbox}*
  K ← *K* U {0}
  G ← *G* U {1}

function deregister(id)
  Ids ← *Ids* \ {id}
  Coors ← *Coors* \ {bbox}
  K ← *K* \ {id}
  G ← *G* \ {id}

23: **function association (S_n, S_{n-1})**
24: **for** *i* ← *0* **to** *n* **do**
25: **for** *j* ← *0* **to** *n* **do**
26: D_{ij} ← *calculate_distance($S_n[i]$, $S_{n-1}[j]$)*
27: *P* ← *argsort (D_i)*
28: *N* ← *argmin (D_P)*
29:
30: **if** $D_{Pk, Nk} \leq 100$ **then**
31: S_{Pk} ← *Snk*
32: **else if** $D_{Pk, Nk}$ *and age* ≤ 100 **then**
33: $S_{predict}$ ← *prediction (id, Coors)*
34: S_{Pk} ← *Spredict*
35: **else**
36: *deregister(S_{id})*
37: **function prediction (id, Coors)**
38: temp_Bbox ← $Coors_{id}$
39: *n = length_of_tempBbox₀*
tempX ← $tempBbox_0$, $dx = (\sum_{k=0}^{n} tempX_k)/n$
40: *tempY* ← $tempBbox_1$, $dy = (\sum_{k=0}^{n} tempY_k)/n$
tempW ← $tempBbox_2$, $dw = (\sum_{k=0}^{n} tempW_k)/n$
41: *tempH* ← $tempBbox_3$, $dh = (\sum_{k=0}^{n} tempH_k)/n$
$newX_{n+1}$ ← $tempBbox_{0,n} + dx$
$newY_{n+1}$ ← $tempBbox_{1,n} + dy$
$newW_{n+1}$ ← $tempBbox_{2,n} + dw$
42: $newH_{n+1}$ ← $tempBbox_{3,n} + dh$

43: *return newBbox* ← $[newX_{n+1}, newY_{n+1}, newW_{n+1}, newH_{n+1}]$

3.4. Araçların Trafik Akış Bilgilerinin Hesaplanması

Nesne takip algoritması ile video sahnesine giren her araç sahneyi terk edene dek olan zaman süresi içinde takip edilerek, araç iz bilgileri çıkarılır. Ve bu araçların zamana bağlı izdüşüm bilgileri, ana modüle, yani araçların trafik akış bilgilerini hespalama birimine yönlendirilir. Bu ana birimin çalışma süreci Algoritma.4 ile gösterilmektedir. Ana birimin girdisi olan, araçların iz bilgilerini içeren çok boyutlu dizinin veri yapısı Denklem.3.16 ile gösterilmektedir. Bu dizi araç sınıf bilgisi, araçın benzersiz kimlik numarası, ve araçların çizdiği iz bilgilerinden oluşur.

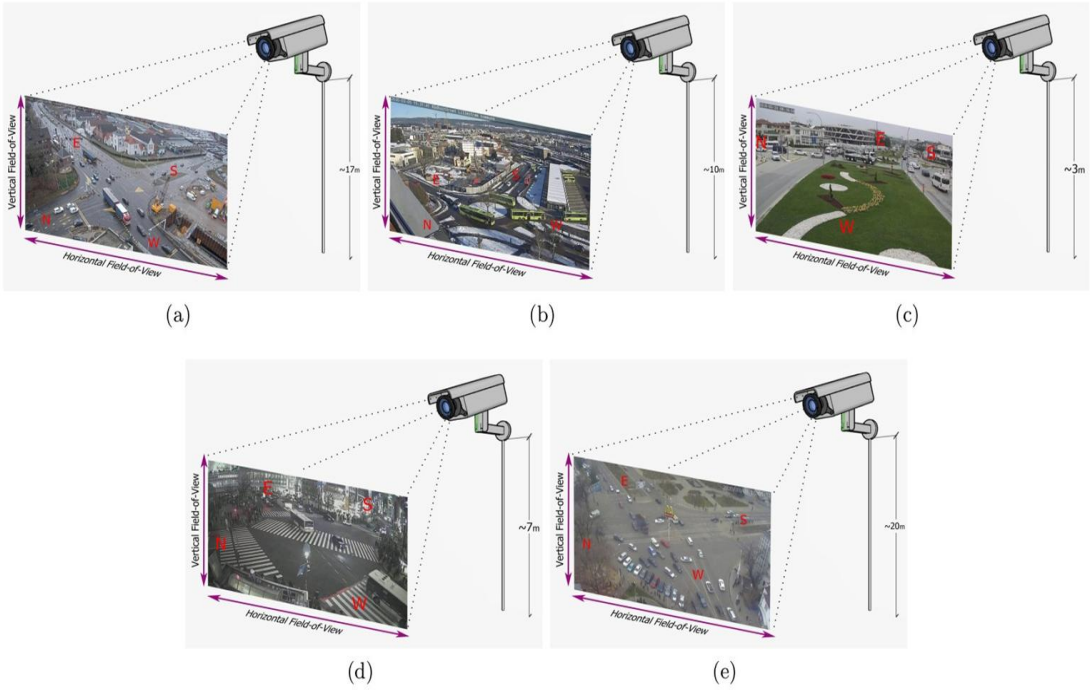
$$\begin{aligned} Tr &= [tr_0, tr_1, \ldots, tr_n], \\ tr_0 &= [vehicle_{label}, unique\_vehicle_{id}, vehicle\_coordinate], \\ vehicle_{coordinate} &= [[x_0, y_0], [x_1, y_1], \ldots, [x_n, y_n]] \end{aligned} \tag{3.16}$$

Algoritma 4. Araç takip verilerinden araçların zamana bağlı izlerini çıkarma.

1: *Araçların yegane kimlikleri ve sınırlayıcı kutu bilgilerinden oluşan araç takip verileri: AT*
2: **function trajectory_extraction(***AT***)**
3: *Tr* ← vehicle_tracker(AT)
4: *return Tr*
5: **end function**

3.4.1. Dört yönlü kavşaklar için trafik akış bilgisi hesaplama

Dört yönlü kavşaklarda, toplamda 16 adet giriş-çıkış yol kombinasyonu mevcuttur, Şekil 3.13. Örneğin, dört yönlü bir kavşakta Doğu, Batı, Kuzey, Güney olmak üzere dört yön vardır. Bu dört yandan DD, DB, DK, DG, …, GG gibi 16 tane trafik akış yönü oluşturulabilmektedir. Dört yönlü bir kavşaktan bu yönlere göre sayım veya kavşağa giren ve çıkan araçların toplam sayımı, yönlere göre anlık hız, ortalama hız, kavşağa giriş ve kavşaktan çıkış noktaları arasındaki geçen süre, ve kavşaktaki sürücü davranışları, kavşak çevresindeki hava kirliliği, veya yakıt israfı, hatta kavşakta harcanan zaman gibi değerli trafik akış bilgileri elde etmek mümkündür. Bu çalışmada, kavşaklardaki, araç sınıfına ve araçların izlediği rotalara göre kategorik ve toplam sayım, hız bilgileri elde edilmiştir. Bu trafik bilgilerini doğru ve gerçek zamanlı bir biçimde hesaplama işlemi; araçların gerçek zamanlı olarak tespit edilmesi, algılanan araçların ard arda gelen video çerçevelerinde araç takip algoritmalarıyla takip edilmesi, araçların izlerinin çıkarılması ve çıkarılan araç izlerinden trafik bilgilerinin hesaplanması gibi dört modülden oluşmaktadır. Araç algılama işlemi, kavşaklardan toplanan görüntülerle eğitilmiş YOLOv3 nesne tespit algoritması ile yapılarak, araçların türleri ve konumlarına ait bilgiler elde edilir. Elde edilen araç tespit verileri araç takip etme algoritmasının girdisi olarak kabul edilir. Araç tespit ünitesinin çıktısı olan araçlara ait sınırlayıcı kutu bilgilerini girdi olarak alan araç takip etme birimi, araçların sahneye girdiği an itibari ile araçlara yegana, değişmez kimlik atayarak, araçlar sahneyi terk edene kadar izler. Çıktı olarak her bir araça ait araç takip verilerini üretir. Araç izi çıkarma birimi ise bu verileri kullanarak her bir araçın iki boyutlu düzlemde çizdiği izleri çıkarır, ve çıktı olarak



Şekil 3.13. Örnek dört yönlü kavşaklar. a) Hollanda, b) İsveç, c) Türkiye, d) Japonya, e) Ukrayna. N-North (Kuzey), S-South (Güney), E-East (Doğu) ve W-West (Batı) yönleri temsil etmektedir.

Algoritma 5. Araçların zamana bağlı izlerinden trafik akış bilgilerini hesaplama.

1: *Araçların zamana bağlı olarak çıkarılan iz verilerinden oluşan vektör:Tr*
2: **function traffic_flow_estimation(***Tr***)**
3: *Traffic_Info* ← vehicle_tracker(Tr)
4: *return Traffic_Info*
5: **end function**

araçların çizdiği iz bilgilerini son ünite olan trafik bilgisi hesaplama birimine sevkeder. Araç iz çıkarım üntesinin çıktılarını girdi olarak alan trafik akış bilgisi hesaplama modülü, araç izlerinden; araç sınıfına, araçların giriş çıkış yönlerine (16 tane yöne) ve bir kavşağa giren-çıkan toplam araçlara ait sayım, anlık ve ortalama hız gibi trafik akış bilgilerini hesaplar. Bu işlemler, Algoritma.5 ile gösterilmiştir.

3.4.2. İki yönlü (geliş-gidişli) otoyollar için trafik bilgisi hesaplama

İki yönlü (geliş-gidişli) şehir içi yol veya otoyollarda hareket eden araçların kategorik ve toplam sayım verileri, anlık veya ortalama hız bilgilerini gerçek zamanlı olarak hesaplama işlemi de, kavşaklarda olduğu gibi araç algılama, araç takibi, zamana bağlı araç izlerinin çıkarımı ve trafik bilgilerini hesaplama gibi dört aşamadan oluşmaktadır. Ancak, 4 yönlü bir kavşakta 16 tane giriş-çıkışlı hareket yönü mevcuttur. Otoyollarda ise sadece geliş ve gidiş yönleri vardır. Dolayısıyla, kavşaklar için geliştirilen araç algılama birimi direk olarak otoyollarda araç algılama için kullanılamamaktadır. Kullanıldığı takdirde, araçların algılama doğruluğu düşmekte olup bu durum araç takip, araç iz çıkarımı ve trafik bilgilerini hesaplama birimlerinin doğruluğunu doğrudan etkilemektedir.

Onun dışında, kavşak ve otoyollardaki kameraların kurulum açısı ve konumları da farklıdır, bu durum da araç algılama sistemini olumsuz etkilemektedir. Araç algılama sistemlerini olumsuz etkileyen bir başka sebeb ise, otoyollardaki trafik akışının kavşaklardakine nazaran kat kat daha hızlı olmasıdır. Bu durum da kavşaklar için geliştirilen veri kümeleriyle eğitilen araç algılama sistemlerini otoyollar için kullanışsız hale getirmektedir. Bu nedenlerden ötürü, şehir içi yol veya otoyollarda kurulu olan kameralardan elde edilen görüntülerden oluşan otoyol veri kümesine ihtiyaç duyulmuştur. İhtiyaç duyulan veri kümesi, YouTube'den alınan şehir içi yol ve otoyol video görüntüleri kullanılarak oluşturulmuş ve YOLOv3 algoritması bu yeni oluşturulan otoyol veri kümesi ile eğitilerek otoyollar için özgü araç algılama modeli geliştirilmiştir. Otoyollar için geliştirilen trafik akış bilgilerini hesaplama sisteminin birinci aşaması olan araç algılama ünitesi bu şekilde geliştirilmiştir. Trafik akış bilgisi çıkarılmak üzere seçilen 9 tane otoyol video görüntüleri (Şekil 3.14.) bu model ile işlenerek araçlar algılanmış ve algılanan araçlara ait sınırlayıcı kutu bilgileri çıktı olarak üretilmiştir. Araç algılama biriminin çıktısı olan sınırlayıcı kutu bilgileri, araç takip etme biriminin girdisi olup, araçlar sınırlayıcı kutu tabanlı araç takip algoritmasından faydalanılarak takip edilmiştir. Araç takip biriminin çıktıları, araç iz verilerini çıkarma modülünün girdisi olarak kullanılmıştır. Zamana bağlı olarak, gerçek zamanlı bir biçimde çıkarılan araçların iz verilerinden kategorik ve

toplam sayım, hız gibi trafik akış verileri hesaplanmıştır. Tüm bu işlemler, Algoritma.6 ile sözde ko olarak ayrıntılı bir şekilde gösterilmektedir.

Algoritma 6. Şehir içi yol veya otoyollarda araçların trafik akış bilgilerini hesaplama.

1: ***Veri:*** *Video kaynağı (dosya yolu/URL/kamera IP): Ardışık karelere ayrıştırılan video kaynağını, Φ ve i←0.* ***Sonuç:*** *Toplam ve kategorik araç sayısı, N_{toplam}, N_{araba}, N_{kamyon}, $N_{otobüs}$, hız bilgisi: v_{araba}, v_{kamyon}, $v_{otobüs}$.*

2: Değişkenleri tanımlama;

3: **while** *ardışık karelerin sonu değilse, Φ* **do**

4: *Çerçeveleri oku: $f_i \leftarrow \Phi_i$ and $f_{i-1} \leftarrow \Phi_{i-1}$;*

5: *$B_i \leftarrow AracAlgilama\ (fi)$;*

6: *$B_{i-1} \leftarrow AracAlgilama\ (fi_{-1})$*

7: *$VT_i \leftarrow AracTakip\ (B_i, B_{i-1})$*

8: *$Tr \leftarrow AracIzCikarim\ (VT_i)$*

9: *$N_{total}, N_{car}, N_{truck}, N_{bus} \leftarrow AracSayma\ (Tr)$*

10: *$N_{total} \leftarrow N_{car} + N_{truck} + N_{bus}$*

11: *$v_{car}, v_{truck}, v_{bus} \leftarrow AracHizHesaplama\ (Tr)$*

12: *$trafik\_bilgisi \leftarrow N_{toplam}, N_{araba}, N_{kamyon}, N_{otobüs}, v_{araraba}, v_{kamyon}, v_{otobüs}$*

13: *$i \leftarrow i + 1$*

14: *trafik_bilgisi*

(a) Video 1 (b) Video 2 (c) Video 3

(d) Video 4 (e) Video 5 (f) Video 6

(g) Video 7 (h) Video 8 (i) Video 9

Şekil 3.14. Çalışmada kullanılan şehir içi yol ve otoyol örnekler.

.

BÖLÜM 4. BULGU VE TARTIŞMA

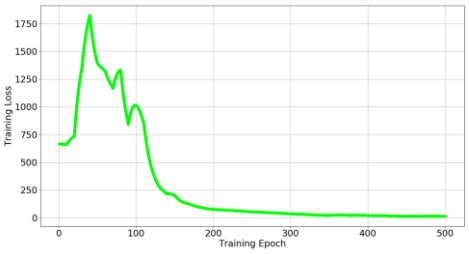
4.1. Kavşaklar için Trafik Akış Bilgisi Hesaplama

Kavşaklar için trafik akış bilgisi hesaplama sistemi için Dünya'nın muhtelif yerlerinde bulunan kavşaklar incelenmiş ve bu çalışma için beş tanesi seçilmiştir. Bu kavşaklardan çekilen örnek videolar ile ilgili detaylı bilgiler Tablo 4.1. ile gösterilmektedir. Bu kavşakların her birinden bir saat olmak üzere, toplamda beş saatlik video görüntüleri çekilmiştir. Sonrasında, her bir kavşağa özgü araç tespit modelleri eğitmek amaçlı veri kümeleri hazırlanmıştır. Video görüntülerinin tamamı her saniyede 30 görüntü çerçevesinden oluşmaktadır, dolayısıyla bir dakikada 1800 tane, bir satte ise 108000 adet görüntü ortaya çıkmaktadır. Bu görüntülerdeki araçları manüel olarak etiketlemek zamanı kaybı ve çok zor olması sebebiyle, görüntüler, YOLOv3 nesne tespit algoritmasının genel maksatlı eğitilmiş modeli ile işlenmiş ve görüntülerdeki araçlar tespit edilmiştir. Tespit edilen araçların ne kadar doğru tespit edildiği veya tespit edilemeyen araçlar varsa düzeltilmek üzere ön tespit işleminden geçirilen görüntüler iki bağımsız uzman tarfından incelenmiş ve görüntülerdeki yanlış algılamalar düzeltilmiştir.

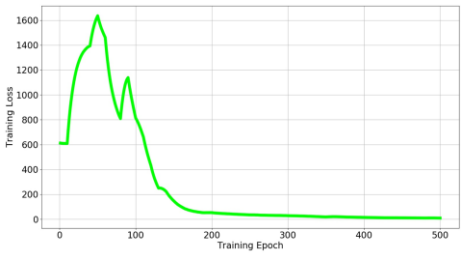
Tablo 4.1. Örnek kavşakların test videoları ile ilgili ayrıntılı bilgiler ve seçilme nedenleri.

No.	Seçilme nedeni	Süresi (dak.)	Ülke	YouTube bağlantısı (https://www.youtube.com/)
a	Öğlen, yüksek konumlu kamera, geniş açılı çekim	4	Hollanda	watch?v=R8t5F nLYKU
b	Öğlen, yüksek konumlu kamera, kenardan çekim	6	İsveç	watch?v=ariYGLo9T44
c	Sabah, düşük yükseklik, yatay geniş açılı çekim	3	Türkiye	watch?v=9ZTLpKDzwNo
d	Akşam, alçak irtifalı geneş açılı çekim	4	Japonya	watch?v=kkryf5Z8UHw
e	Öğlen, yüksek irtifalı çekim	9	Ukrayna	watch?v=CthbczCzYys

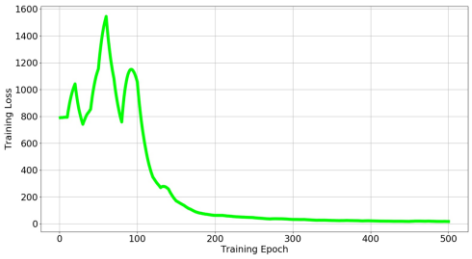
Gerekli düzeltmeler yapılan etiketlenmiş görüntülerden oluşan veri kümeleri ile beş kavşak için ayrı ayrı YOLOv3 tabanlı araç algılama ağırlık modelleri geliştirilmiştir. Geliştirilen ağlık modellerinin her biri, YOLOv3 algoritmasi ile 24 saat süreli eğitim süresinden geçirilimiş ve hata (kayıp) fonksiyonu değeri %10'un altına indiğinde eğitim süreci durdurulmuş ve araç tespit ağırlık modelleri test süreci için hazır olarak kabul edilmiştir. Eğitim sürecindeki kata fonksiyon değerinin eğitim devir sayısına (epoches) bağlı grafikleri Şekil 4.1. ile detaylı olarak gösterilmektedir.



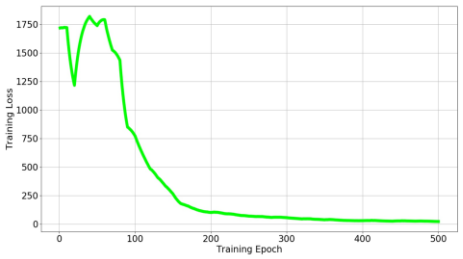
a). Hollanda



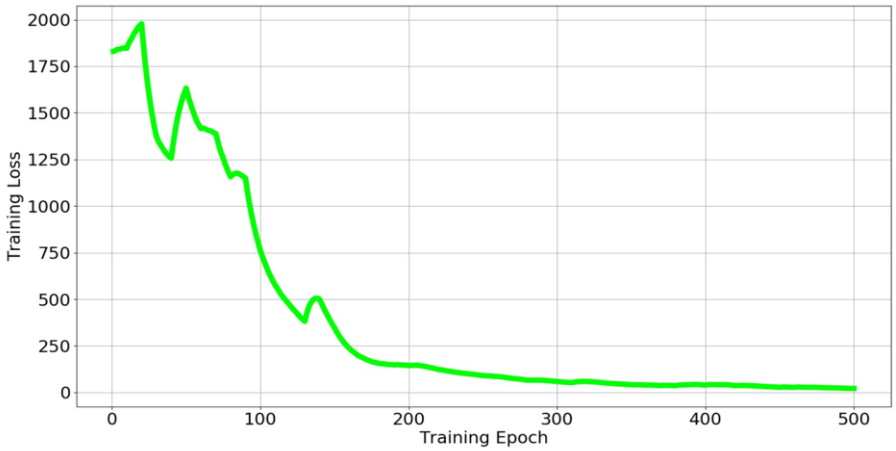
b). İsveç



c). Türkiye

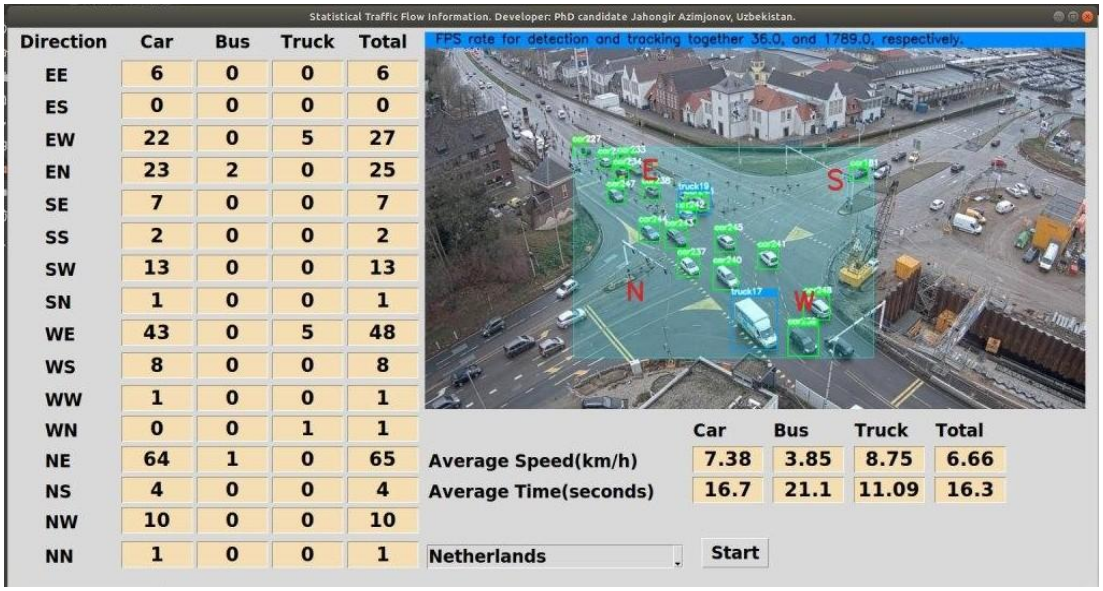


d). Japonya



e). Ukrayna

Şekil 4.1. Örnek kavşaklar için geliştirilen araç algılama modellerinin eğitim sürecine ait hata fonksiyonnun değeri ve devir sayısı arasındaki bağlantı grafiği.



Şekil 4.2. Geliştirilen, gerçek zamanlı trafik akış bilgilerini çıkarma yazılımının arayüzü

YOLOv3 tabanlı, her bir kavşağa özgü araç algılama ağırlık modelleri oluşturulduktan sonra, bu modeller Tablo 4.1.'de gösterilen deney videolarıyla test edilmiştir. Test sonuçları araç algılama ve araç takip sonuçları olarak değil, bilakis, yönlere göre ve toplam araç sayım ve hız ölçümü olarak verilmiştir. Çünkü, araç sayım ve hız ölçüm verileri doğrudan araç iz verisinden hesaplanmaktadır, araç iz verisi ise araç algılama ve araç takip verilerinden çıkarılmaktadır. Yönlere, araç sınıflarına göre araç sayım, toplam araç sayım ve anlık ve ortalama hız gibi trafik akış bilgileri gerçek zamanlı olarak hesaplanmış, hesaplanan sonuçları anlık olarak göstermek için form tabanlı bir ara yüz geliştirilmiştir. Bu arayüz, Python 3 programlama dilinin Tkinter kütüphanesi kullanılarak geliştirilmiştir, Şekil 4.2.

Kavşak görüntüleri, YOLOv3 nesne tespit algoritması ve merkezi nokta tabanlı araç takip metoduyla işlenerek geliştirilen trafik akış bilgisi çıkarma sisteminin yanında, YOLOv3 ve Kalman filtresine dayalı araç takip algoritmalarını kullanarak trafik akış bilgisi hesaplama sistemi de oluşturulmuştur. Oluşturulan bu sistemler ile beş kavşağa ait video görüntüleri işlenmiş yönler ve araç sınıflarına göre ve toplam araç sayım bilgileri hesaplanmıştır. İki sisteminin de doğrulukları, uzman insan tarafından, çıplak gözle sayarak elde edilen araç sayım sonuçları ile karşılaştırılmıştır. Karşılaştırma sonucunda birbirinden ilginç sonuçlar elde edilmiştir.

Tablo 4.2. Hollanda ve İsveç'teki kavşaklar için sayım sonuçları. G-Ground Truth, K-Kalman, C-Centroid.

	Araba			Doğ.(%)		**Otobüs**			Doğ.(%)		**Kamyon**			Doğ.(%)		Ort.Doğ.(%)	
	G	K	C	K	C	G	K	C	K	C	G	K	C	K	C	K	C
Yön	**Hollanda**																
E-E	0	1	6	50	14.3	0	0	0	100	100	0	0	0	100	100		
E-S	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
E-W	37	12	29	32.4	78.4	0	0	0	100	100	7	6	6	85.7	85.7		
E-N	24	12	24	50	100	2	2	2	100	100	0	0	0	100	100		
S-S	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
S-W	13	13	13	100	100	0	0	0	100	100	0	0	0	100	100		
S-N	1	1	1	100	100	0	0	0	100	100	0	0	0	100	100		
S-E	7	5	7	71.4	100	0	0	0	100	100	0	0	0	100	100		
W-W	0	0	3	100	25	0	0	0	100	100	0	0	0	100	100	**76.0**	**84.5**
W-N	0	0	0	100	100	0	0	0	100	100	1	1	1	100	100		
W-E	61	39	54	63.9	88.5	0	0	0	100	100	5	4	6	80	83.3		
W-S	7	0	8	12.5	87.5	0	0	0	100	100	0	0	0	100	100		
N-N	0	0	1	100	50	0	0	0	100	100	0	0	0	100	100		
N-E	65	56	64	86.2	98.5	1	1	1	100	100	0	0	0	100	100		
N-S	4	4	4	100	100	0	0	0	100	100	0	0	0	100	100		
N-W	16	4	17	25	94.1	0	0	0	100	100	0	0	0	100	100		
Avg.Acc.(%)				**75.4**	**80.3**	Avg.Acc.(%)			**100**	**100**	Avg.Acc.(%)			**97.4**	**97.6**		
Yön	**İsveç**																
E-E	0	0	0	100	100	0	0	0	100	100	1	0	1	50	100		
E-S	2	2	2	100	100	0	0	0	100	100	0	0	0	100	100		
E-W	9	8	9	88.9	100	0	0	0	100	100	0	0	0	100	100		
E-N	3	0	3	25	100	0	0	0	100	100	0	0	0	100	100		
S-S	0	0	0	100	100	1	0	1	50	100	0	0	0	100	100		
S-W	21	20	16	95.2	76.2	3	1	2	33.3	66.7	0	0	0	100	100		
S-N	22	22	21	100	95.5	2	1	3	50	66.7	1	1	1	100	100		
S-E	2	3	2	66.7	100	0	1	0	50	100	0	0	0	100	100		
W-W	1	0	3	50	33.3	2	0	3	33.3	66.7	0	0	0	100	100		
W-N	23	23	22	100	95.7	0	0	0	100	100	0	0	0	100	100	**81.7**	**88.4**
W-E	6	7	7	85.7	85.7	0	0	0	100	100	0	0	0	100	100		
W-S	21	0	23	4.5	91.3	2	0	2	33.3	100	1	0	1	50	100		
N-N	0	0	2	100	33.3	0	0	0	100	100	0	0	0	100	100		
N-E	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
N-S	32	29	30	90.6	93.8	2	2	2	100	100	0	0	0	100	100		
N-W	13	13	13	100	100	0	0	1	100	100	0	0	0	100	100		
Avg.Acc.(%)				**78.2**	**85.5**	Avg.Acc.(%)			**100**	**100**	Avg.Acc.(%)			**100**	**100**		

Genel olarak, merkez nokta tabanlı araç izleme yaklaşımı Kalman filtresi tabanlı araç takip yönteminden daha iyi sonuçlar üretmiştir. Fakat araçların kavşağa giriş ve çıkış yönleri birbirine çok yakınsa, o zaman merkezi nokta tabanlı takip algoritması iyi

sonuçlar üretemiyor, çünkü bu algoritma, araçların yön ve hız bilgilerine göre araç takip etmemektedir. Kalman filtresine dayalı takip algoritması, yön ve hız bilgilerine dayanarak araç takip işlemini gerçekleştirdiği için U-dönüşlerde yüksek doğrulukla çalışmaktadır. Bu durum, Tablo 4.2.'de E-E, W-W ve N-N yön çizgilerinde görülebilen kavşakta bir araç U-dönüşü yaptığında söz konusudur. Kalman filtresi tabanlı araç takip yaklaşımı, araçların anlık ve geçmişe dönük yön durum bilgisini tutuyor olması nedeniyle bunun gibi durumlarda araç takibini kolay kaybetmez. Ancak, Kalman filtresi tabanlı izleyici, bir çerçevede 30'dan fazla araç olduğunda sınırlayıcı kutu yaklaşımıyla aynı sayıdaki çerçeveyi işleyememektedir, algoritmanın icra hızı keskin düşmektedir. Çünkü Kalman filtresinin en kötü durumdaki zaman karmaşıklığı $O(n^3)$. Merkezi nokta tabanlı araç takip algoritması ise bir "arama ve eşleştirme problemi" olduğundan, $O(n)$ zaman karmaşıklığına sahiptir. Dolayısıyla, geliştirilen araç takip algoritması Kalaman filtresi tabanlı yaklaşıma nazaran çok daha hızlı çalışmaktadır. Deneyler sırasında, bir çerçevedeki araç sayısı arttığında Kalman filtre tabanlı izleyicinin 5 fps'den fazla işlem yapamadığı gözlemlenmiştir.

Kameranın yüksekliğinin ve açısının, araç sayma doğruluğu üzerinde büyük etkisi olduğu gözlemlenmiştir. Kameranın konumu, yatay ve dikey görüş açıları düzgün ayarlanmazsa, yoğun bir biçimda meydana gelen yanılsama ve oklüzyon sorunları nedeniyle sistemin sayım doğruluğu azalmaktadır Şekil 3.13. Türkiye'deki kavşağa yerleştirilen kameranın konumu ve görüş açısı düzgün ayarlanmadığından ötürü araç görüntülerinin kısmi ve tam üst üste gelme sorunu meydana gelmiştir. Bu yüzden, düşük doğruluklu sayım sonuçları elde edilmiştir. Öte yandan, Japonya'daki kavşak kamerasının yüksekliği yüksek olmasa da kamera açısı düzgün ayarlandığı için daha iyi araç sayım sonuçları üretilmiştir. Algılama ve ilişkilendirme algoritmaları, yüksek doğruluklu sayım sonuçları elde etmede çok önemli bir rol oynadığı gözlemlenmiştir. Örneğin, şiddetli yağmur, kar, kamera sallanması, kapanma ve yanılsama gibi herhangi bir nedenle araç algılama başarısız olduğu hallerde; kaybolan araçlar için bir tahmin süreci yürütülse bile, yanlış tahminler, sistemin genel doğruluğunu azaltmıştır.

Tablo 4.3. Türkiye ve Japonya'daki kavşaklar için sayım sonuçları. G-Ground Truth, K-Kalman, C-Centroid.

	Araba			Doğ.(%)		**Otobüs**			Doğ.(%)		**Kamyon**			Doğ.(%)		Ort.Doğ.(%)	
	G	K	C	K	C	G	K	C	K	C	G	K	C	K	C	K	C
Yön	**Türkiye**																
E-E	0	8	2	11.1	33.3	0	0	0	100	100	0	2	0	33.3	100		
E-S	3	9	3	33.3	100	0	0	0	100	100	0	3	0	25	100		
E-W	2	2	3	100	66.7	10	1	7	10	70	13	0	7	7.1	53.8		
E-N	8	0	5	11.1	62.5	0	0	2	100	33.3	2	0	3	33.3	66.7		
S-S	0	0	0	100	100	0	0	0	100	100	0	0	1	100	50		
S-W	3	3	2	100	66.7	0	0	0	100	100	0	1	0	50	100		
S-N	26	1	19	3.8	73.1	0	0	0	100	100	11	9	7	81.8	63.6		
S-E	15	19	13	78.9	86.7	0	2	0	33.3	100	3	6	3	50	100		
W-W	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100	**64.6**	**78.7**
W-N	1	0	2	50	50	0	0	0	100	100	1	0	0	50	50		
W-E	2	0	5	33.3	40	4	0	7	20	57.1	5	0	6	16.7	83.3		
W-S	2	3	2	66.7	100	0	0	0	100	100	0	0	0	100	100		
N-N	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
N-E	23	19	25	82.6	92	0	0	1	100	100	3	2	0	66.7	25		
N-S	1	1	1	100	100	0	0	0	100	50	8	1	5	12.5	62.5		
N-W	3	3	2	66.7	66.7	0	0	0	100	100	0	0	0	100	100		
Avg.Acc.(%)				**60.2**	**75.3**	Avg.Acc.(%)			**81.8**	**89.3**	Avg.Acc.(%)			**57.5**	**82.1**		
Yön	**Japonya**																
E-E	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
E-S	17	16	17	94.1	100	0	1	0	50	100	0	0	0	100	100		
E-W	12	10	12	83.3	100	1	0	1	50	100	0	0	0	100	100		
E-N	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
S-S	0	0	5	100	16.7	0	0	0	100	100	0	0	0	100	100		
S-W	8	6	7	75	87.5	0	0	0	100	100	0	0	0	100	100		
S-N	22	18	21	81.8	95.5	1	0	0	50	50	1	1	1	100	100		
S-E	2	2	2	100	100	0	0	0	100	100	0	0	0	100	100		
W-W	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
W-N	2	0	1	33.3	50	1	1	0	100	50	0	0	0	100	100	**81.9**	**84.7**
W-E	17	15	17	88.2	100	2	0	2	33.3	100	0	0	0	100	100		
W-S	0	0	0	100	100	0	0	0	100	100	0	0	0	100	100		
N-N	0	3	0	25	100	0	0	0	100	100	0	0	0	100	100		
N-E	4	4	6	100	66.7	0	0	0	100	100	0	0	0	100	100		
N-S	19	14	15	73.7	78.9	0	0	0	100	100	0	0	0	100	100		
N-W	1	2	2	50	50	0	0	0	100	100	0	0	0	100	100		
Avg.Acc.(%)				**83.1**	**88.4**	Avg.Acc.(%)			**83.3**	**92.3**	Avg.Acc.(%)			**100**	**100**		

Tablo 4.4. Ukrayna'daki kavşak için sayım sonuçları. G-Ground Truth, K-Kalman, C-Centroid.

	Araba			Doğ.(%)		**Otobüs**			Doğ.(%)		**Kamyon**			Doğ.(%)		Ort.Doğ.(%)	
	G	K	C	K	C	G	K	C	K	C	G	K	C	K	C	K	C
Yön	**Ukrayna**																
E-E	4	16	6	25	66.7	0	0	2	100	33.3	0	1	2	50	33.3		
E-S	29	30	27	96.7	93.1	7	7	6	100	85.7	0	0	0	100	100		
E-W	89	76	88	85.4	98.9	5	4	4	80	80	2	4	3	50	66.7		
E-N	23	33	25	69.7	92	0	1	1	50	50	0	0	0	100	100		
S-S	10	5	19	50	52.6	0	0	0	100	100	0	0	1	100	100		
S-W	24	15	21	62.5	87.5	0	0	0	100	100	3	1	3	33.3	100		
S-N	101	81	83	80.2	82.2	2	1	1	50	50	2	0	0	33.3	33.3		
S-E	58	68	65	85.3	89.2	3	3	3	100	100	2	3	2	66.7	100		
W-W	7	11	8	63.6	87.5	0	0	0	100	100	0	0	1	100	50		
W-N	16	16	13	100	81.2	1	1	1	100	100	0	0	0	100	100		
W-E	68	54	69	79.4	98.6	4	4	4	100	100	3	4	4	75	75	**72.4**	**82.9**
W-S	13	1	16	7.7	81.2	0	0	0	100	100	0	0	0	100	100		
N-N	3	3	6	100	50	0	0	0	100	100	0	0	0	100	100		
N-E	15	20	16	75	93.8	0	0	0	100	100	2	1	0	50	33.3		
N-S	70	63	63	90	90	2	3	3	66.7	66.7	0	0	0	100	100		
N-W	44	32	36	72.7	81.8	3	2	2	66.7	66.7	0	0	0	100	100		
Avg.Acc.(%)				**69.7**	**81.6**	Avg.Acc.(%)			**90.8**	**84.5**	Avg.Acc.(%)			**77.6**	**81.4**		

Ukrayna'daki kavşak trafiği diğer kavşaklardaki trafik akışından biraz daha düzensiz ve bilgi çıkarma açısından zordur, Tablo 4.4. Bu kavşaktaki kamera yüksekliğinin ve görüş açısının oldukça uygun bir biçimde ayarlanmış olmasına rağmen, trafik akışının düzensizliği, araç takip algoritmalarını olumsuz etkilemiş, dolayısıyla hem Kalman hem de merkezi nokta tabanlı takip algoritmalar diğer kavşaklara göre istenilen performansı sergileyememiştir.

İlginç bir şekilde, Tablo 4.5.'te kamyon tipi araçlar Hollanda'daki kavşakta diğer araç tiplerine göre daha hızlı hareket ettiği görülmektedir. Video kaydının detaylı incelemesinde tırların çoğunlukla D-B ve B-B yönünde aktığı ve bu yön ana hat olması itibariyle araçların genellikle kavşağı diğer yönlere göre daha hızlı geçtiği görülmiştir Şekil 4.2. Bu nedenle, kamyonlar daha büyük/ağır araçlar olmalarına rağmen, bu kavşaktan en hızlı geçen araç sınıfı olarak görülmektedir.

Tablo 4.5. Kavşaklardaki araçların ortalama hareket hızı ve kavşaktan geçiş süresi.

Ortalama Hız (km/h) ve Kavşağı Geçme Süresi (saniye.)						
	Araba		Kamyon		Otobüs	
Ülke	km/h	saniye	km/h	saniye	km/h	saniye
Hollanda	6.5	18.2	3.9	18.7	7.4	19.6
İsveç	4.6	32.7	4.1	48.7	9.7	36.2
Türkiye	6.2	38.3	5.8	37.5	4.3	33.0
Japonya	12.0	29.9	14.6	29.9	6.6	34.3
Ukrayna	6.4	79.0	5.1	79.0	6.1	70.9

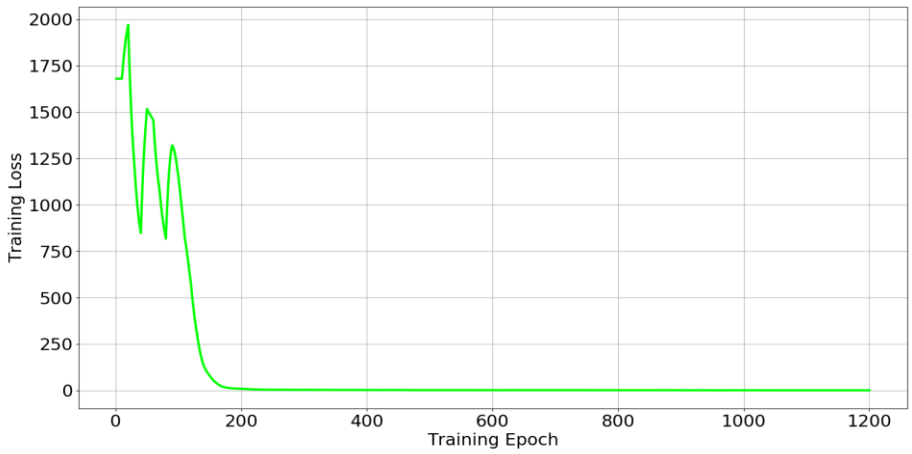
4.2. Otoyollar için Trafik Akış Bilgisi Hesaplama

Otoyollar için trafik akış bilgisi hesaplama sistemi için Dünya'nın muhtelif yerlerinde bulunan şehir içi yol ve otoyollardan çekilmiş video örnekleri incelenmiştir. İnceleme sürecinde, otoyollardan direk olarak çevirimiçi video kaydı yapan video örneği bulunamamış dolayısıyla, Youtube'de olan geçmişte çekilen video kayıtlarıyla yetinilmiştir. Bu çalışma için 12 tane otoyol seçilmiş olup, 9 tanesi eğitim için diğer üç tanesi ise test kümesi olarak kullanılmıştır.

Tablo 4.6. Deney otoyol videoları ile ilgili ayrıntılı bilgiler ve seçilme nedenleri.

No.	Seçilme nedeni	Süresi (dak.)	Konumu	YouTube bağlantısı (https://www.youtube.com/)
a	Trafiğin çok yoğun olduğu vakit performansı	6.5	A3 Otoyolu, Eashing, İngiltere	watch?v=UM0hX7nomi8
b	Olağandışı ağır yüklü kamyon yavaş geçiyor	1.5	M25 Karayolu, Tilbury, Londra, İngiltere	watch?v=SLedNXzxsoo
c	İki şeritli otoyol trafiği.	14	Fransız Alpleri	watch?v=nt3D26lrkho
d	Birçok motosikletin geçtiği ve görüş açısının ağaçlarla kapatıldığı otoyol.	5	Changi Otoyolu, Singapur	izle?v=lbYKYpqVEmw
e	Karayolu, gece trafiği	34.5	Seul, Güney Kore	watch?v=xEtM1I1Afhc
f	Çok araba geçen otoyol	4.5	M25 Karayolu, Surrey, İngiltere	watch?v=byUXOWzyiyo
g	Trafik ışıkları yakınındaki yoğun trafikli normal yol	1.5	Taiwan	watch?v=WxgtahHmhiw
h	İki gelişli yol birleşimi	5	Varşova, Polonya	watch?v=MNn9qKG2UFI
i	Geliş-gidişli yol, anormal trafik akışlı	6.5	Serdivan, Sakarya, Türkiye	watch?v=6bR3jL7q-iI

Bu otoyollardan çekilen örnek videolar ile ilgili ayrıntılı bilgiler Tablo 4.6. ile gösterilmektedir. Deney otoyol videolarından elde edilen trafik görüntülerinin eğitim için ayrılan kısmı, YOLOv3 genel amaçlı nesne tespiti ağırlık modeli kullanılarak ön tespit işleminden geçirilmiş ve ön tespit işlemi ile etiketlenmiş görüntüler iki bağımsız uzman tarafından incelenerek, yanlış etiketlemeler düzeltilmiş, etiketlenmeyen araçlar etiketlenip tüm otoyollar için tek bir veri kümesi oluşturulmuştur. Oluşturulan veri kümesi ile YOLOv3 algoritması eğitilerek otoyollar için özgü bir araç tespit ağırlık modeli oluşturulmuştur. Eğitim süreci 48 saat ve 100.000 devirde (epoches) gerçekleştirilmiştir. Hata fonksiyonu değeri, %10'luk bir değerin altına indiğinde eğitim işlemi durdurulmuştur. Eğitim sürecinin hata fonksiyonu ve devir arasındaki bağlılık grafiği Şekil 4.3. ile gösterilmektedir.

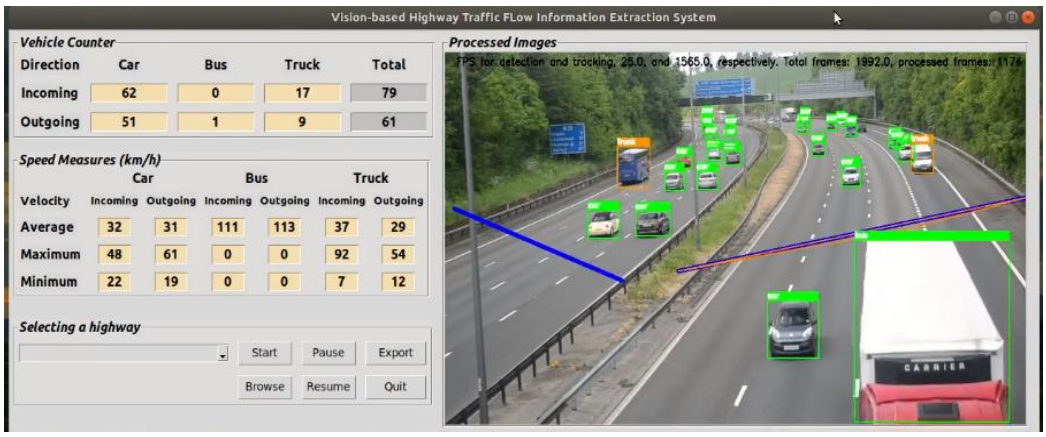


Şekil 4.3. Otoyollar için eğitilmiş modelin hata fonksiyonu ve eğitim deviri arasındaki bağlılık grafiği.

Hazırlanan araç tespit modeli ile deney videolar işlenerek araçlar tespit edilmiş, algılanan araçlar, araç takip sisteminin girdileri olarak kullanılmıştır. Araç takip sisteminin çıktıları olan izleme verilerinden araçların zamana bağlı iz bilgileri çıkarılmıştır. Araçların yörünge (iz) bilgilerini temsil eden bir örnek Şekil 4.4. ile gösterilmektedir. Araçların zamana bağlı olarak çıkartılan yörünge bilgilerinden araçların sınıfına ve hareket yönüne sayım ve toplam sayım, araçların hızları gibi trafik akış verileri hesaplanmıştır. Bu hesaplamaları yapmak geliştirilen yazılım sistemi ile birbirinden ilginç sonuçlar elde edilmiş. Aşağıda, elde edilen tüm sonuçlar

ayrıntılı olarak açıklanmıştır. Geliştirilen yazılımın ara yüzü Şekil 4.5. ile gösterilmektedir.

Şekil 4.4. Araçların, araç takip verilerinden çıkarılan yörüngeleri.



Şekil 4.5. Otoyollardan trafik akış bilgisi hesaplama sisteminin ara yüzü.

Araç yörüngelerin doğru bir biçimde çıkarılması, ‘yüksek seviyeli trafik akışı bilgilerini’ hesaplamada büyük bir rol oynar. Yani, doğru bir şekilde çıkarılan araç yörüngeleri yüksek doğruluklu araç sayım ve hız gibi trafik akış bilgilerini

hesaplamaya imkân sağlar. Araç yörüngelerinin çıkarılması ise tamamen araç algılama ve izleme bilgileri gibi düşük seviyeli trafik verilerine bağlıdır. Buna karşılık, görüş tabanlı araç algılama ve takip sistemleri, çoğunlukla derin veya makine öğrenimi algoritmaları gibi modern veri işleme yöntemlerine dayanır. Görüntü işleme sistemlerinden başarılı bir çıktı almak için giriş video verilerinin net olması gerekir. Bu açıdan kameraların doğru bir biçimde konuşlandırılması çok önemlidir. Mesala, kameranın, görüntüsü çekilen otoyol alanından uzaklığı, kamera konumunun otoyol yüzeyinden yüksekliği ve çekilen alanın yatay ve dikey görüş alanı doğru bir şekilde ayarlanarak, kamera kurulumu yapıldığında, tüm trafik bilgileri titiz bir biçimde hesaplanabildiği görülmüştür. Dolayısıyla, kameraların otoyol ve kavşakların en uygun yerine, uygun bir şekilde yerleştirilmesi büyük önem taşımaktadır.

Bu çalışmada, araçlar; araba, kamyon ve otobüs gibi üç temel sınıfa ayırılarak tespit edilmiştir. Otomobil ve otobüsün tanımlanmasında ve tespitinde pek sorun olmazken, minibüs, kamyonet, pikap, küçük kamyon ve TIR'ların hepsinin kamyon sınıfında sayılması nedeniyle kamyon sınıfı sorun yaratmıştır. Özellikle, minibüslerin, genellikle ilk kadrajda yanlış araç olarak algılandığı ve düzeltilmezse bu durum takibin doğruluğunu olumsuz yönde etkilediği gözlemlenmiştir. Bu nedenle, araç veri setleri hazırlanma esnasında, araçların sınfı titizlikle tanımlanması gerekmektedir.

Tablo 4.7. Bu çalışma sonuçlarının mevcut çalışmalar sonuçlarıyla karşılaştırma. K*-Kalman, B-Sınırlayıcı Kutu, PCA[1]-Principal Component Analysis, SSD[2]-Single Shot Detector, ORB[3]-Oriented fast and Rotated Brief.

Ref.	Algılayıcı	Takip Edici	Platf.	Sınıf.	FPS	Doğ.(%)
Liu ve ark.	Arka Plan Çıkarma	-	CPU	No	10	91.5
Yang ve ark.	Arka Plan Çıkarma	Kalman filtresi	CPU	No	12	92.2
Abdelwahab	Derin Sinir Ağları	Kande-Lucas-Tomasi	GPU	No	15	90.4
Rosas ve ark.	İyileştirilmiş PCA[1]	-	CPU	No	24	92.9
Meng ve ark.	SSD[2] 512x512	Korelasyon Eşleştirme	CPU	Yes	25	93.1
Song ve ark.	YOLOv3	ORB[3]	GPU	Yes	30	93.2
K*	YOLOv3	Kalman filtresi	GPU	Yes	20	87.2
B**	YOLOv3	Sınırlayıcı Kutu	GPU	Yes	35	94.5

Tablo 4.7. mevcut çalışmanın daha önce yayınlanmış çalışmalarla karşılaştırmasını göstermektedir. Karşılaştırma, araç tespit/takip algoritmaları, bilgi işlem platformları, sınıflandırma, saniyede çerçeve işleme (fps üst sınırı) olarak ve ilgili makalelerde bildirilen doğruluk değerleri temel alınarak yapılmıştır. Uygulamalarımız için doğruluk sonuçları (hem Kalman filtresi hem de sınırlayıcı kutu yaklaşımları) araç sınıfları dikkate alınıp ağırlıklı ortalama formülü kullanılarak hesaplanmıştır. Ancak, bildirilen doğruluk değerlerinin çoğu ideal koşullar altında elde edilmiştir. Tablo 4.7.'den de görüleceği gibi, sınırlayıcı kutu algoritması hem fps sayısı hem de doğruluk açısından mevcut algoritmalardan daha üstün performans sergilemektedir.

Tablo 4.8. Deney otoyol videolarından hesaplanan araç sayım sonuçları. G-Temel Doğru, K-Kalman, B-Bbox. I-Incoming ve O-Outgoing.

	Araba			**Doğ.(%)**		**Kamyon**			**Doğ.(%)**		**Otobüs**			**Doğ.(%)**		**Ort.Doğ.(%)**	
	G	***K***	***B***	***K***	***B***	***G***	***K***	***B***	***K***	***B***	***G***	***K***	***B***	***K***	***B***	***K***	***B***
Video 1 (Şekil 4.5.a)																	
I	159	154	154	96,9	96,9	45	41	41	91,1	91,1	0	0	0	100	100	95,6	95,6
O	245	253	254	96,8	96,5	96	109	121	88,1	79,3	0	0	0	100	100	94,4	91,7
Video 2 (Şekil 4.5.b)																	
I	34	25	32	73,5	94,1	10	4	11	40	90,9	0	0	0	100	100	65,9	93,4
O	106	129	101	82,2	95,3	12	13	12	92,3	100	0	0	0	100	100	83,2	95,8
Video 3 (Şekil 4.5.c)																	
I	397	405	394	98	99,2	166	152	165	91,6	99,4	0	0	0	100	100	96,1	99,3
O	269	272	268	98,9	99,6	147	153	144	96,1	98	0	0	0	100	100	97,9	99
Video 4 (Şekil 4.5.d)																	
I	430	515	496	83,5	86,7	115	141	109	81,6	94,8	12	35	20	34	60	82,1	87,8
O	468	546	497	85,7	94,2	154	164	151	93,9	98,1	8	10	8	80	100	87,6	95,2
Video 5 (Şekil 4.5.e)																	
I	621	599	573	96,5	92,3	221	185	176	83,7	79,6	31	24	33	77	94	92,6	89,1
O	926	1033	977	89,6	94,8	382	408	369	93,6	96,6	44	49	49	90	90	90,7	95,2
Video 6 (Şekil 4.5.f)																	
I	95	109	100	87,2	95	35	39	34	89,7	97,1	3	5	5	60	60	87,2	94,8
O	77	92	77	83,7	100	34	36	33	94,4	97,1	2	3	2	67	100	86,6	99,1
Video 7 (Şekil 4.5.g)																	
I	46	47	47	97,9	97,9	11	13	12	84,6	91,7	6	6	6	100	100	95,8	97
O	35	52	40	67,3	87,5	7	10	9	70	77,8	6	1	6	17	100	61,4	87,7
Video 8 (Şekil 4.5.h)																	
I1	233	250	237	93,2	98,3	72	75	72	96	100	2	16	6	13	33	93,3	98,3
I2	30	33	32	90,9	93,8	4	6	3	66,7	75	2	2	3	100	67	88,7	90,2
Video 9 (Şekil 4.5.i)																	
I	347	412	355	84,2	97,7	38	44	34	86,4	89,5	1	2	1	50	100	84,3	96,9
O	275	323	318	85,1	86,5	61	54	58	88,5	95,1	2	3	3	67	67	85,6	87,9

Tablo 4.8.'de sunulan kategorik, yöne göre ve toplam araç sayım çıktıları analiz edilerek, Kalman filtresi ve Sınırlayıcı Kutu yaklaşımlarına dayalı çalışmaların sonuçları arasındaki farklar aşağıdaki şekilde sıralanmıştır:

Video 1: Bu deneyde kamera, gelen yön görüntüsünü, giden yön görüntüsünden daha iyi bir görüş açısıyla çekecek bir konuma yerleştirilmiş. Sonuç olarak, gelen istikametteki araçların görüntüsü daha iyi görüş açısıyla çekilmiş, bu yüzden de yanılsama veya oklüzyon olayları meydana gelmemiş ve araç tespit algoritması daha iyi daha sorunsuz çalışmıştır. Bundan dolayı, hem Kalman filtresi hem de sınırlayıcı kutu tabanlı nesne izleme algoritmaları hemen hemen eşit doğrulukla çalışmışlar. Bu durum, araç sayım sonuçlardan görülebilmektedir. Ancak, giden yönde daha fazla tıkanıklık gözlemlenmiştir. Kalman filtresi tabanlı nesne takip sistemi, sınırlayıcı kutu tabanlı izleme algoritmasına kıyasla kısmi tıkanma durumlarında iyi performans sergilemiştir. Ayrıca, bazı durumlarda, arabalar ve küçük boyutlu kamyonların sınıfı tam olarak ayrıt edilememiş olmasına rağmen Kalman filtresi yaklaşımı bu durumdan pek etkilenmemiştir.

Video 2: Bu deneyde, kamera gelen yönü daha iyi görüş açısıyla çekmektedir, ancak çekim sırasında kamera titreşimi meydana gelmiştir. Kameranın titremesinden sebebiyle, araçların video çerçevelerindeki konumları farklı yerlerdeymiş gibi çekilmiş. Dolaysıyla, araçlar farklı hız ve yönlere hareket ediyormuş gibi bir yanılsama vucüde gelmiştir. Bu durumdan en çok Kalman filtresi tabanlı takip algoritması muzdarip olmuş, çünkü Kalman filtresi herhangi bir küçük (minör) harekete karşı çok duyarlıdır. Ancak, sınırlayıcı kutu araç takip algoritması bu durumda daha iyi performansla çalışmıştır, bunun sebebi ise, bu algoritmanın kamera titreşim filtresinin içermesidir. Sınırlayıcı kutu algoritmasına gömülen kamera titreşim filtresi, bu algoritmayı bu tür olaylara karşı daha dayanıklı hale getirmiştir. Öte yandan, küçük boyutlu kamyonlar, genellikle, gelen yöndeki arabalar olarak algılanmış, Kalman filtresi onları giden yöndeki arabalar listesine ilginç bir şekilde eklemiştir. Bunun nedeni çıkış yönünde hızlı hareketler ve gelen yönde anormal yavaş hareketler olarak gösterilebilir, çünkü Kalman filtresi araçların hız durumlarına karşı aşırı duyarlıdır.

Video 3: Bu videoda, hem gelen hem de giden yönler geniş bir görüş açısı ile çekilmiştir. Bu nedenle, nesne algılama sistemi çok yüksek doğrulukla çalışmış, bu durum, her iki araç takip sisteminin sonuçlarına olumlu etkide bulunmuş. Araç takip algoritmaları yüksek doğruluklu sonuçlar ürettiğinden, araçların yörüngeleri düzgün bir şekilde çıkarılmış. Düzgün araç yörüngelerinden çok yüksek doğruluklu araç sayım çıktıları hesaplanmıştır.

Video 4: Bu deneyde, kamera anormal bir şekilde sağa ve sola hareket edip, şiddetli bir biçimde titremektedir. Onun dışında, geliş yönünde, görüş açısını anormal bir şekilde engelleyen ağaç dalı ve yaprakları görünmektedir. Bu nedenle, Kalman filtresi tabanlı izleyici, araçları düşük doğrulukla takip etmiş, bu durum da araç sayım sistemine olumsuz yansımıştır. Özellikle, araç algılama algoritması, bir aracı tam tıkanma nedeniyle hem kamyon hem de otobüs olarak birçok kez algılamıştır. Bu durum Kalman filtresini otobüs tipi araçların sayımında oldukça başarısız kılmıştır. Ama yeni bir kamera titreşim ve oylama filtrelerine sahip sınırlayıcı kutu yaklaşımı, gidiş yönündeki tüm araç türlerine ait araç sayım işlemini yüksek doğrulukla gerçekleştirmiştir.

Video 5: Bu video gece saatlerinde çekilmiş olmasına rağmen, sonuçlar her iki yaklaşım için de oldukça başarılıydı. Aslında, böyle durumlarda, araçların şekillerinin ve piksellerinin renk değerleri birbiriyle kolaylıkla karıştırılabilmektedir. Ayrıca, araçların farlarının neden olduğu, araçların gelen istikamette algılanmasını ve araç tiplerini ayırt etme kabiliyetini olumsuz etkileyen anormal parlaklık olayları vardır. Bu deneyde, Kalman filtresi otobüsleri gelen yönde sınırlayıcı kutu yaklaşımından daha düşük bir doğrulukla izlemiş, dolayısıyla, otobüs sayımında sınırlayıcı kutu yaklaşımına nazaran daha az doğrulukla çalışmıştır.

Video 6: Bu videoda, çıkış yönü geniş görüş açılı, geliş yönü ise biraz dar görüş açılı çekilmiştir. Kalman filtresi, otobüsler dışında, diğer iki türdeki araçların sayma işlemini iyi bir şekilde gerçekleştirmiştir. Öte yandan, kamyonlar bazen otobüs olarak tespit edilmiş, algılama ünitesinin yanlış sınıflandırması nedeniyle, her iki yönde de kamyonlar için Kalman filtresi biraz düşük performans sergilemiştir. Genel

olarak, araç algılama biriminin yanlış sınıflandırma durumlarından olumsuz etkilenen her iki araç takip algoritması giden yöndeki araçların sayımında daha düşük doğrulukla çalışmıştır.

Video 7: Bu videoda, geliş yönü geniş görüş açısı ile gidiş yönü ise dar bir görüş açısı ile çekilmiştir. Bu nedenle, nesne algılama algoritması, gelen araçları daha yüksek doğrulukla, giden araçları ise düşük doğrulukla algılamıştır. Algılama ünitesindeki düşük doğruluk nedeniyle, Kalman filtresi çoğu zaman giden istikametteki araçları takip düzgün bir şekilde takip edememiş. Ancak, sınırlayıcı kutu algoritması bundan fazla etkilenmemiştir. Çünkü sınırlayıcı kutusu, herhangi bir araçı kendi kalıcı listesine dâhil etmeden önce oylama filtresinden geçirmektedir. Bu oylama filtresi katmanı, araç algılama modülünün birçok eksiğini düzeltmeye hizmet etmektedir.

Video 8: Bu videoda gelen-1 ve gelen-2 şeritleri bulunmaktadır. Bir şerit geniş bir görüş açısı ile diğeri ise dar bir görüş açısı ile çekilmiştir. Ancak, bu videoda nesne tespiti için kameranın yüksekliği en uygun olarak ayarlanmıştır. Böylece, arabalar, her iki şeritte de çok yüksek doğrulukla tespit edilmiş, bu durum, her iki izleme algoritmasının yüksek doğrulukta araç takip çıktıları üretmesine katkıda bulunmuştur. Ancak, tırlar yüksek doğrulukla takip edilmesine rağmen, gelen-1 şeridindeki otobüsler için epeyce yanlış pozitif durumları mevcuttu.

Video 9: Bu videoda gelen ve giden yönler geniş görüş açıları ile çekilmiştir, fakat kameranın hafif titremesi mevcuttur. Bu durum, Kalman filtresi tabanlı takip algoritmasını olumsuz etkilemiştir. Kalman filtresi yaklaşımı, özellikle, arabaları takip etmede bu durumdan olumsuz etkilenmiştir.

Genel olarak, kamera sarsıntısı Kalman filtreleme izleyicisinin doğruluğunu olumsuz etkilerken, sınırlayıcı kutu izleyici algoritması, kamera titreşimi filtresi sayesinde daha iyi sonuçlara ulaşmıştır. Kalman filtresinin, kamera sarsıntısında olumsuz etkilenmesi ve araç takibinde zorlanmasının nedeni; tahmin fonksiyonlarıdır. Kalman filtresinin tahmin fonksiyonları, önceki hareket hızı ve ivme parametrelerine ve

nesnenin çeşitli durum bilgilerine göre kaybolan bir nesnenin; mevcut ve sonraki durumunu tahmin etmektedir. Tahmin işlevleri, Kalman filtre izlemesini kısmi veya tam tıkanmaya veya parlaklık yanılsama olaylarına karşı daha güçlü kılarken, bu durum, kamera sallanması nedeniyle algılanan nesne konumunun küçük hareketlerine karşı bu algoritmayı yetersiz kılmaktadır.

Ayrıca, YOLOv3 algılama algoritmasının kararsız bir nesne konumlandırma sorunu vardır. Algoritma, ardışık çerçevelerdeki sabit veya hareketli aynı nesneleri, asıl bulundukları yerden biraz dah farklı konumlarda algılayabilmektedir. Araçların konumlarındaki bu küçük kaymalar, kaybolan nesnenin kesin veya daha yakın sonraki durumunu tahmin etmek için tahmin işlevlerini yanlış yönlendirmektedir. Bununla birlikte, YOLOv3'ün genel önermeli nesne algılama modeli, araçları sık sık yanlış algılamakta veya yanlış sınıflandırmaktadır. Çoğu durumda, önerilen sınırlayıcı kutu algoritması, yanlış algılama ve yanlış sınıflandırma sorununu oylama filtresi yaklaşımıyla çözmüşken, Kalman filtresine dayalı araç takip algoritması bu tür durumlarda zayıf kaldığı, hatta çoğunlukla başarısız olduğu gözlemlenmiştir. Buna ek olarak, belirli bir karede 30'dan fazla nesne olduğu hallerde Kalman filtresi tabanlı araç izleme algoritmasının görüntü işleme performansı keskin bir şekilde saniyede 5-6 fps'e düşerken, sınırlayıcı kutu algoritması bir çerçeveyi çok yüksek fps ile işleyebilmiştir.

Tablo 4.9. Araçların ortalama, maxsimum ve minimum hız değerleri (km/saat biriminde gösterilmiştir).

	Araba						Kamyon						Otobüs					
	Orta.		Max.		Min.		Orta.		Max.		Min.		Orta.		Max.		Min.	
Ref.	I	O	I	O	I	O	I	O	I	O	I	O	I	O	I	O	I	O
a	82	96	104	144	36	36	71	80	145	157	36	42	-	-	-	-	-	-
b	21	71	49	96	10	55	34	74	87	108	20	44	-	-	-	-	-	-
c	89	91	136	143	36	37	79	86	117	103	33	32	-	-	-	-	-	-
d	63	74	66	75	19	18	62	78	64	85	19	19	66	79	72	85	35	18
e	80	92	114	125	21	45	75	91	109	124	29	33	73	81	87	90	35	42
f	75	79	85	93	26	34	78	82	115	119	30	36	74	72	85	92	29	25
g	12	28	20	56	12	12	39	46	46	54	11	24	27	29	36	44	14	17
h*	35	24	79	36	4	13	43	33	73	65	5	13	45	52	64	70	25	34
i	72	65	90	83	16	23	85	73	95	88	12	44	24	29	50	53	3	5

Bu çalışmada, esasen, araçların kategorik, yöne göre ve toplam sayım işlemine daha çok odaklanılmışsa da, araçların hız bilgileri de titiz bir biçimde hesaplanmıştır. Çıkarılan hız bilgileri tam olarak doğrulanmamış olduğundan dolayı, kesin bir tartışma yapılamamıştır. Ancak, tahmini olarak hesaplanan araçların hızları Tablo 4.9. ile gösterilmiştir.

BÖLÜM 5. SONUÇ VE ÖNERİLER

Bu çalışmada, şehir/kentsel alanların trafiğini izleme, yönetme ve planlama sistemlerinde yardımcı olabilecek; kavşak, şehir içi yol ve otoyollar için görü tabanlı (vision-based), gerçek zamanlı akıllı trafik akış bilgisi hesaplama ve izleme (monitoring) sistemleri geliştirilmiştir. Çalışmada, birçok kavşak, şehir içi yol ve otoyol örnekleri incelenmiştir. İncelemeler sonucunda, kavşak ve otoyollar için ayrı ayrı trafik akış izleme sistemleri geliştirilmesi uygun görülmüştür. Çünkü, kavşak ve karayollar; yapıları ve onlarda hareket eden araç ve nesnelerin davranışları itibariyle birbirinden farklı oldukları tespit edilmiştir. Kavşak ve otoyollar sıkı bir biçimde, titizlikle incelenerek, kavşaklar için beş deney kavşak, otoyollar için ise 9 deney karayolu seçilmiştir. Ve deneyler, seçilen kavşak ve otoyollardan elde edilen videolar üzerinde gerçekleştirilmiştir. Geliştirilen görü tabanlı araç izleme sistemi; dört ana hattan oluşup dört çeşit ana çıktı üretmektedir. Bu sonuçlar, aşağıda hatlara göre detaylı olarak açıklanmıştır.

Birinci hatta, araçlar tespit edilmektedir, yani araçların iki boyutlu görüntü düzlemindeki konumu, sınıfı ve tespitin ne kadar doğrulunu gösteren güven değeri gibi çıktıları alınmaktadır. Geliştirilen yazılım sistemi gerçek zamanlı olmasından ötürü, yazılımın tüm hatları, görevlerini gerçek zamanlı olarak icra etmeleri gerekmektedir. Dolayısıyla, araç tespiti birimi için birçok, genel amaçlı nesne algılama algoritmaları incelenmiş ve netice itibariyle YOLOv3 algoritmasının kullanılmasına karar verilmiştir. Ancak, YOLOv3 algoritmasının araç tespit doğruluğu %57'nin altında olması nedeniyle, YOLOv3 algoritması deney kavşaklardan elde edilen video görüntüleriyle tekrardan eğitilerek, kavşak ve otoyollar için ayrı araç tespit ağırlık modelleri oluşturulmuştur. Eğitim için kullanılan video görüntülerinin sayısı 200.000'den fazla olması sebebiyle, etiketleme süreci iki aşamaya ayrılmıştır. Birin aşamada, video görüntüler, YOLOv3 ile ön etiketleme işleminden geçirilmiştir. Sonrasında ise iki bağımız uzman tarafından ön etiketlenmiş

görüntüler incelenmiş, yanlış etiketlemeler düzeltilimiş, etiketlenmeyen araçlar etiketlenmiş, böylece görüntülerdeki eksikler giderilerek daha sağlıklı kavşak ve otoyol araç tespit veri kümeleri oluşturulmuştur. Oluşturulan veri kümeleri tekrardan eğitilmiş, netice olarak hem kavşaklar hem de otoyollar için %90'nın üstünde algılama doğruluğu elde edilmiştir. Deney videolar, bu oluşturulan araç tespit ağırlık modelleri ile işlenmiştir. Araç tespit çıktıları ikinci aşama olan araç takip sistemlerinin girdisi olarak kullanılmıştır.

İkinci hat olan araç izlemede ise araçlar, ardışık video çerçevelerinde, sahneye girdiği an itibariyle sahneyi terk edene kadarki zaman diliminde yegane kimlikler atanarak takip edilmiştir. Araç takip hattı için ayrı ayrı iki algoritma kullanılmıştır. Birincisi, var olan Kalman filtresi yaklaşımına dayalı araç takip sistemi, ikincisi ise bu çalışamada geliştirilen merkezi ve sınırlayıcı kutu tabanlı araç takip yaklaşımlarıdır. Her iki yaklaşım da girdi olarak araç tespit sisteminin çıktıları olan sınırlayıcı kutuları almakta, sınırlayıcı kutulara yegane kimlik atamakta, ardışık video çerçevelerinde araçların durumunu yegane kimlikleri kaybetmeksizin yenilemekte, ve araçlar sahneyi terk etmesi itibarıyla onları takip etmeyi bırakmaktadır. Bunun, dışında, araç takip sisteminin çok önemli bir bileşeni vardır, bu bileşen, araçların anlık kaybolması veya sahnede görünmeyişi nedeniyle mevcut veya sonraki konum durumlarının tahmin edilerek belirlenmesidir. Araçların sahneden anlık kayboluşunun ana nedneleri, ya araç tespit algoritmasından, ya kötü hava şartları ya illüzyon ya da oklüzyon sorunlarından kaynaklanmaktadır. Bu sorunlar sebebiyle araçlar, sahneyi terk etmedikleri halde sahnede görünmemektedirler. Bu durumlarda, araç tespit algoritmaları devre dışı kalmaktadır. Meydana gelen bu tür sorunlar, ancak araç takip algoritmaları tahmin fonksiyonları kullanılarak çözebilmektedir.

Netice itibariyle, üçüncü hatta, araç takip sisteminin çıktılarından araçların yörüngeleri çıkarılmıştır. Dördüncü, son hatta ise çıkarılan araçların yörüngeleri kullanılarak araç sayım, hız ve süre gibi trafik akış bilgileri hesaplanmıştır. Hesaplanan trafik akış sistemlerinin çıktıları, şehir trafik izleme, yönetme ve planlama araçlarında kullanılmak üzere Sakarya Büyük Şehir Belediyesinin Trafik

Şube Müdürlüğüne iletilmiştir. Onlar bu çıktıları ilgili araçlarda kullanarak, bu çalışmada geliştirilen yazılım sistemlerinin çıktılarının kıymetli ve yararlı olduğunu bildirmişlerdir. Gelecekte, bu çalışmada geliştirilen yazılım araçları, trafik yönetme ve planlama araçlarıyla bütünleştirerek aynı çatı altında toplanabileceği öngörülmüştür.

Çalışma sonucunda elde edilen çıktılar değerlendirilerek, gelecekte yapılacak bu tür sistemler için üç ana unsurun çok önemli olduğu belirlenmiştir. Bu unsurlar titizlikle gerçekleştirildiğinde, trafik akış izleme ve yönetme sistemleri daha doğru, daha performanslı daha optimal, daha maliyetsiz olarak geliştiririlebileceği öngörülmektedir. Birinci unsur, trafik akış izleme sistemlerinde, sensörlerin doğru seçilmesi ve seçilen sensörün (kamera olması arzu edilir) doğru yükseklikte, yatay ve dikey görüş açılarının doğru bir biçimde ayarlanarak kavşak veya otoyolun en uygun mekanına konuşlandırmasıdır. Bu adım, doğru yapıldığında, araç tespit ve araç takip sistemlerinin daha başarılı çalışmalarına büyük katkıda bulunacağı düşünülmektedir.

İkinci ve üçüncü unsurlar olan, araç tespit ve takip sistemleri birleştirilerek optimize edilmesidir. Araç tespitinde konum belirleme çok baskın bir öneme sahiptir. Yalnız, ardışık video çerçevelerindeki araçların konumları, önceki çerçeveler için tutulmamaktadır. Eğer önceki çerçevelerdeki konumlar, araç takip sistemiyle tutularak, sıradaki gelen çerçeve, görüntü işleme birimine sevk edilmeden önce araçların tahmini konumları tespit edilirse ve araç tespit algoritması sadece tahmin edilen bölgelere yoğunlaşırsa, araç tespit algoritması girdi görüntünün gereksiz yerlerinde araç aramak zorunda kalmayacağı, dolayısıyla, girdi görüntü üzerinde gereksiz hesaplamalar yapmayacağı, bu da araç tespit performansına olumlu olarak yansıyacağı düşünülmektedir. Bu üç ana hat doğru ve optimal yapılarak, daha maliyetsiz trafik akış izleme ve yönetme sistemleri geliştirilebileceği öngörülmektedir.

KAYNAKLAR

Abdelwahab, M. (2019). Fast approach for efficient vehicle counting. Electronics Letters, 55(1):20-22.

Adeleke, O. O., Jimoh, Y. A., and Akinpelu, M. A. (2013). Development of an advanced public transportation system for captive commuters on urban arterials in ilorin, nigeria. Alexandria Engineering Journal, 52(3):447-454.

Ahmad, F., Basit, A., Ahmad, H., Mahmud, S. A., Khan, G. M., and Yousaf, F. Z. (2013). Feasibility of deploying wireless sensor-based road side solutions for intelligent transportation systems. In 2013 International Conference on Connected Vehicles and Expo (ICCVE), pages 320-326.

Andersen, J. and Sutclie, S. (2000). Intelligent transport systems (its) - an overview. IFAC Proceedings Volumes, 33(18):99{106. IFAC Conference on Technology Transfer in Developing Countries: Automation in Infrastructure Creation (DECOM-TT 2000), Pretoria, South Africa, 5-7 July 2000.

Andrecut, M. (2009). Parallel gpu implementation of iterative pca algorithms. Journal of computational biology : a journal of computational molecular cell biology, 16:1593-9.

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., and Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5167-5176.

Arrospide, J., Salgado, L., Nieto, M., and Jaureguizar, F. (2008). On-board robust vehicle detection and tracking using adaptive quality evaluation. In 2008 15th IEEE International Conference on Image Processing, pages 2008-2011.

Aytekin, B. and Altu, E. (2010). Increasing driving safety with a multiple vehicle detection and tracking system using ongoing vehicle shadow information. In 2010 IEEE International Conference on Systems, Man and Cybernetics, pages 3650-3656.

Azimjonov, J. and Özmen, A. (2021). A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on high-ways. Advanced Engineering Informatics, 50:101393.

Badino, H., Franke, U., and Mester, R. (2012). Free space computation using stochastic occupancy grids and dynamic programming.

Barth, A. and Franke, U. (2008). Where will the oncoming vehicle be the next second? In 2008 IEEE Intelligent Vehicles Symposium, pages 1068-1073.

Barth, A. and Franke, U. (2009). Estimating the driving state of oncoming vehicles from a moving platform using stereo vision. IEEE Transactions on Intelligent Transportation Systems, 10(4):560-571.

Barth, A. and Franke, U. (2010). Tracking oncoming and turning vehicles at intersections. In 13th International IEEE Conference on Intelligent Transportation Systems, pages 861-868.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. ArXiv.

Bota, S. and Nedevschi, S. (2011). Tracking multiple objects in urban traffic environments using dense stereo and optical flow. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 791-796.

Broggi, A., Cattani, S., Cardarelli, E., Kriel, B., McDaniel, M. S., and Chang, H. (2011). Disparity space image's features analysis for error prediction of a stereo obstacle detector for heavy duty vehicles. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 80-86.

Catalin, G. and Nedevschi, S. (2008). Object tracking from stereo sequences using particle filter. In 2008 4th International Conference on Intelligent Computer Communication and Processing, pages 279-282.

Chan, Y.-M., Huang, S.-S., Fu, L.-C., and Hsiao, P.-Y. (2007). Vehicle detection under various lighting conditions by incorporating particle filter. In 2007 IEEE Intelligent Transportation Systems Conference, pages 534-539.

Chang, W.-C. and Cho, C.-W. (2008). Real-time side vehicle tracking using parts-based boosting. In 2008 IEEE International Conference on Systems, Man and Cybernetics, pages 3370-3375.

Chauhan, N. K. and Singh, K. (2018). A review on conventional machine learning vs deep learning. In 2018 International Conference on Computing, Power and Communication Technologies (GUCON), pages 347-352.

Chen, Y., Ardila-Gomez, A., and Frame, G. (2017). Achieving energy savings by intelligent transportation systems investments in the context of smart cities. Transportation Research Part D: Transport and Environment, 54:381-396.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3):273{297.

Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research, 2:265-292.

Cui, J., Liu, F., Li, Z., and Jia, Z. (2010). Vehicle localization using a single camera. In 2010 IEEE Intelligent Vehicles Symposium, pages 871-876.

Danescu, R., Oniga, F., and Nedevschi, S. (2011). Modeling and tracking the driving environment with a particle-based occupancy grid. IEEE Transactions on Intelligent Transportation Systems, 12(4):1331-1342.

Datondji, S. R. E., Dupuis, Y., Subirats, P., and Vasseur, P. (2016). A survey of vision-based traffic monitoring of road intersections. IEEE Transactions on Intelligent Transportation Systems, 17(10):2681-2698.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248-255.

El Mokaddem, Y., Jawab, F., and Sad, L. E. (2019). Intelligent transportations systems: Review of current challenges and success factors: The case of developing countries. In 2019 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA), pages 1-6.

Erbs, F., Barth, A., and Franke, U. (2011). Moving vehicle detection by optimal segmentation of the dynamic stixel world. In 2011 IEEE Intelligent Vehicles Symposium (IV), pages 951-956.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98-136.

Fernandez-Sanjurjo, M., Bosquet, B., Mucientes, M., and Brea, V. M. (2019). Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence, 85:410-420.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381-395.

Franke, U., Rabe, C., Badino, H., and Gehrig, S. (2005). 6d-vision: Fusion of stereo and motion for robust environment perception. In Kropatsch, W. G., Sablatnig, R., and Hanbury, A., editors, Pattern Recognition, pages 216-223, Berlin, Heidelberg. Springer Berlin Heidelberg.

Geetha, S. and Cicilia, D. (2017). IoT enabled intelligent bus transportation system. In 2017 2nd International Conference on Communication and Electronics Systems (ICCES), pages 7-11.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In the 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354-3361.

Girshick, R. (2015). Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1440-1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 580-587.

Grinberg, M., Ohr, F., and Beyerer, J. (2009). Feature-based probabilistic data association (fbpda) for visual multi-target detection and tracking under occlusions and split and merge effects. In 2009 12th International IEEE Conference on Intelligent Transportation Systems, pages 1-8.

Guerrero-Ibanez, J., Zeadally, S., and Contreras-Castillo, J. (2018). Sensor technologies for intelligent transportation systems. Sensors (Basel, Switzerland), 18(4):1212.

Hall, R. and Intihar, C. (1997). Commercial vehicle operations: Government interfaces and intelligent transportation systems. Institute of Transportation Studies, UC Berkeley, Institute of Transportation Studies, Research Reports, Working Papers, Proceedings.

Haselhoff, A. and Kummert, A. (2009). An evolutionarily optimized vehicle tracker in collaboration with a detection system. In 2009 12th International IEEE Conference on Intelligent Transportation Systems, pages 1-6.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770-778.

Hermes, C., Einhaus, J., Hahn, M., Whler, C., and Kummert, F. (2010). Vehicle tracking and motion prediction in complex urban scenarios. In 2010 IEEE Intelligent Vehicles Symposium, pages 26-33.

Higuchi, T., Martin, P., Chakraborty, S., and Srivastava, M. (2015). Anonycast: Privacy-preserving location distribution for anonymous crowd tracking systems. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp'15, page 11191130, New York, NY, USA. Association for Computing Machinery.

Hilario, C., Collado, J., Armingol, J., and de la Escalera, A. (2006). Visual perception and tracking of vehicles for driver assistance systems. In 2006 IEEE Intelligent Vehicles Symposium, pages 94-99.

Hoffmann, C. (2006). Fusing multiple 2d visual features for vehicle detection. In 2006 IEEE Intelligent Vehicles Symposium, pages 406-411.

Idler, C., Schweiger, R., Paulus, D., Mdhlisch, M., and Ritter, W. (2006). Realtime vision based multi-target-tracking with particle filters in automotive applications. In 2006 IEEE Intelligent Vehicles Symposium, pages 188-193.

Inrix, I. (2019). INRIX: Congestion costs each American 97 hours, $1,348 a year. Accessed: 2021-08-20.

Islam, M. J., Wu, Q. M. J., Ahmadi, M., and Sid-Ahmed, M. A. (2007). Investigating the performance of Naive- Bayes classifiers and K- nearest neighbor classiers. In 2007 International Conference on Convergence Information Technology (ICCIT 2007), pages 1541-1546.

Ismagilova, E., Hughes, L., Dwivedi, Y. K., and Raman, K. R. (2019). Smart cities: Advances in research and information systems perspective. International Journal of Information Management, 47:88-100.

Javed, M. A., Zeadally, S., and Hamida, E. B. (2019). Data analytics for cooperative intelligent transport systems. Vehicular Communications, 15:63-72.

Jocher, G., Stoken, A., and Borovec, J. (2021). Yolov5: The fastest object detection algorithm. Accessed: 2021-08-20.

Kanhere, N., Pundlik, S., and Bircheld, S. (2005). Vehicle segmentation and tracking from a low-angle o-axis camera. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 1152-1157.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):881-892.

Kavukcuoglu, K., Ranzato, M., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 1605-1612.

Kem, O., Balbo, F., and Zimmermann, A. (2017). Traveler-oriented advanced traveler information system based on the dynamic discovery of resources: Potentials and challenges. Transportation Research Procedia, 22:635{644. "19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey".

Khan, R., Zawoad, S., Haque, M. M., and Hasan, R. (2014). Otit: Towards secure provenance modeling for location proofs. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, page 8798, New York, NY, USA. Association for Computing Machinery.

Kitt, B., Ranft, B., and Lategahn, H. (2010). Detection and tracking of independently moving objects in urban environments. In 13th International IEEE Conference on Intelligent Transportation Systems, pages 1396-1401.

Kohavi, R. and Provost, F. (1998). Glossary of terms. Machine Learning, 30(2):271-274.

Kowsari, T., Beauchemin, S. S., and Cho, J. (2011). Real-time vehicle detection and tracking using stereo vision and multi-view AdaBoost. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 1255-1260.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classication with deep convolutional neural networks. Communications of the ACM, 60(6):84-90.

Kuncheva, L. I. (2006). On the optimality of Naive Bayes with dependent binary features. Pattern Recognition Letters, 27(7):830-837.

Lategahn, H., Graf, T., Hasberg, C., Kitt, B., and E ertz, J. (2011). Mapping in dynamic environments using stereo vision. In 2011 IEEE Intelligent Vehicles Symposium (IV), pages 150-156.

Leal-Taixe, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). MOTchallenge 2015: Towards a benchmark for multi-target tracking. ArXiv, abs/1504.01942.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553):436-444.

Lee, S., Tewolde, G., and Kwon, J. (2014). Design and implementation of a vehicle tracking system using GPS/GSM/GPRS technology and smartphone application. In 2014 IEEE World Forum on Internet of Things (WF-IoT), pages 353-358.

Lee, Y., Lin, Y., and Wahba, G. (2004). Multicategory support vector machines. Journal of the American Statistical Association, 99(465):67-81.

Lim, Y.-C., Lee, C.-H., Kwon, S., and Kim, J. (2011). Event-driven track management method for robust multi-vehicle tracking. In 2011 IEEE Intelligent Vehicles Symposium (IV), pages 189{194.

Lim, Y.-C., Lee, C.-H., Kwon, S., and Lee, J.-h. (2010). A fusion method of data association and virtual detection for minimizing track loss and false track. In2010 IEEE Intelligent Vehicles Symposium, pages 301-306.

Lin, S.-W., Ying, K.-C., Chen, S.-C., and Lee, Z.-J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Systems with Applications, 35(4):1817{1824.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, Computer Vision - ECCV 2014, pages 740-755, Cham. Springer International Publishing.

Liu, F., Zeng, Z., and Jiang, R. (2017). A video-based real-time adaptive vehiclecounting system for urban roads. PLOS ONE, 12(11):1-16.

Liu, P., Wang, G., Yu, Z., Guo, X., and Lu, W. (2019). Vehicle tracking based on shape information and inter-frame motion vector. Computers and Electrical Engineering, 78:22-31.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, Computer Vision { ECCV 2016, pages 21-37. Springer International Publishing.

Liu, W., Wen, X., Duan, B., Yuan, H., and Wang, N. (2007). Rear vehicle detection and tracking for lane change assist. In 2007 IEEE Intelligent Vehicles Symposium, pages 252-257.

Lombardi, P., Giordano, S., Farouh, H., and Yousef, W. (2012). Modeling the smart city performance. Innovation: The European Journal of Social Science Research, 25(2):137-149.

Lyu, S., Chang, M.-C., Du, D., Wen, L., Qi, H., Li, Y., Wei, Y., Ke, L., Hu, T., Del Coco, M., Carcagn, P., Anisimov, D., Bochinski, E., Galasso, F., Bunyak, F., Han, G., Ye, H., Wang, H., Palaniappan, K., Ozcan, K., Wang, L., Wang, L., Lauer, M., Watcharapinchai, N., Song, N., Al-Shakarji, N. M., Wang, S., Amin, S., Rujikietgumjorn, S., Khanova, T., Sikora, T., Kutschbach, T., Eiselein, V., Tian, W., Xue, X., Yu, X., Lu, Y., Zheng, Y., Huang, Y., and Zhang, Y. (2017). UA-DETRAC 2017: Report of avss2017 amp; iwt4s challenge on advanced traffic monitoring. In 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 1-7.

Mandellos, N. A., Keramitsoglou, I., and Kiranoudis, C. T. (2011). A background subtraction algorithm for detecting and tracking vehicles. Expert Systems with Applications, 38(3):1619-1631.

Manen, S., Gygli, M., Dai, D., and Van Gool, L. (2017). Pathtrack: Fast trajectory annotation with path supervision. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 290-299.

Martinez, F. J., Toh, C.-K., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2010). Emergency services in future intelligent transportation systems based on vehicular communication networks. IEEE Intelligent Transportation Systems Magazine, 2(2):6-20.

Mei, X. and Ling, H. (2011). Robust visual tracking and vehicle classification via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(11):2259-2272.

Meng, Q., Song, H., Zhang, Y., Zhang, X., Li, G., and Yang, Y. (2020). Video-based vehicle counting for expressway: A novel approach based on vehicle detection and correlation-matched tracking using image data from ptz cameras. Mathematical Problems in Engineering, 2020:1969408.

Milan, A., Leal-Taixe, L., Reid, I., Roth, S., and Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. ArXiv, abs/1603.00831.

Milletler, B. (2019). World population prospects 2019. Accessed: 2021-08-20.

Moqqaddem, S., Ruichek, Y., Touahni, R., and Sbihi, A. (2011). A spectral clustering and Kalman filtering based objects detection and tracking using stereo vision with linear cameras. In 2011 IEEE Intelligent Vehicles Symposium (IV), pages 902-907.

Nguyen, B. and Brilakis, I. (2018). Real-time validation of vision-based overheight vehicle detection system. Advanced Engineering Informatics, 38:67-80.

Ni, D. (2016). Traffic flow theory. pages 379-386. Butterworth-Heinemann.

Nuevo, J., Parra, I., Sjberg, J., and Bergasa, L. M. (2010). Estimating surrounding vehicles' poses using computer vision. In 13th International IEEE Conference on Intelligent Transportation Systems, pages 1863-1868.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 1717-1724.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2015). Is object localization for free? Weakly supervised learning with convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Pitts, W. and McCulloch, W. S. (1947). How we know universals is the perception of auditory and visual forms. The bulletin of mathematical biophysics, 9(3):127-147.

Rabe, C., Franke, U., and Gehrig, S. (2007). Fast detection of moving objects in complex scenarios. In 2007 IEEE Intelligent Vehicles Symposium, pages 398-403.

Reddy, P. B., Reddy, B. R., and Asadi, S. (2018). Evaluation of maintenance management in construction to reduce the maintenance cost. International Journal of Mechanical Engineering and Technology, 9(3):367-374.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection.

Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6517-6525.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. ArXiv, abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6):1137-1149.

Rokach, L. and Maimon, O. (2005). Top-down induction of decision trees classifiers - a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35(4):476-487.

Rosas-Arias, L., Portillo-Portillo, J., Hernandez-Suarez, A., Olivares-Mercado, J., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Sandoval Orozco, A. L., and Garca Villalba, L. J. (2019). Vehicle counting in video sequences: An incremental subspace learning approach. Sensors, 19(13).

Rouf, I., Miller, R., Mustafa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., and Seskar, I. (2010). Security and privacy vulnerabilities of in car wireless networks: A tire pressure monitoring system case study. In Proceedings of the 19th USENIX Conference on Security, USENIX Security'10, page 21, USA. USENIX Association.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088):533-536.

S., K. and S., V. (2018). Contour-based object tracking in video scenes through optical flow and gabor features. Optik, 157:787-797.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks.

Shahgholian, M. and Gharavian, D. (2018). Advanced traffic management systems: An overview and a development strategy.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556.

Sivaraman, S. and Trivedi, M. M. (2010). A general active-learning framework for on-road vehicle recognition and tracking. IEEE Transactions on Intelligent Transportation Systems, 11(2):267-276.

Sivaraman, S. and Trivedi, M. M. (2011). Combining monocular and stereovision for real-time vehicle ranging and tracking on multilane highways. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 1249-1254.

Sivaraman, S. and Trivedi, M. M. (2013). Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. IEEE Transactions on Intelligent Transportation Systems, 14(4):1773-1795.

Song, D., Tharmarasa, R., Florea, M. C., Duclos-Hindie, N., Fernando, X. N., and Kirubarajan, T. (2019). Multi-vehicle tracking with microscopic traffic flow model-based particle filtering. Automatica, 105:28-35.

Song, H., Liang, H., Li, H., Dai, Z., and Yun, X. (2019b). Vision-based vehicle detection and counting system using deep learning in highway scenes. European Transport Research Review, 11(1):51.

Sotiris, K. (2007). Supervised machine learning: A review of classification techniques. Informatica (Ljubljana), 31.

Sullivan, F. (2018). A guide to intelligent transportation systems and best practices. Accessed: 2021-08-20.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1-9.

Takeuchi, A., Mita, S., and McAllester, D. (2010). On-road vehicle tracking using deformable object model and particle filter with integrated likelihoods. In 2010 IEEE Intelligent Vehicles Symposium, pages 1014-1021. Tehrani Niknejad, H., Takeuchi, A., Mita, S., and McAllester, D. (2012). On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation. IEEE Transactions on Intelligent Transportation Systems, 13(2):748-758.

Thiagarajan, A., Biagioni, J., Gerlich, T., and Eriksson, J. (2010). Cooperative transit tracking using smartphones. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10, page 8598, New York, NY, USA. Association for Computing Machinery.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. (2019). MOTs: Multi-object tracking and segmentation. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7934-7943.

Wadley, F. M. (1952). Probit analysis: A statistical treatment of the sigmoid response curve. Science, 116(3011):286-287.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.-C., Qi, H., Lim, J., Yang, M.-H., and Lyu, S. (2020). UA-DETRAC: A new benchmark and protocol for multiobject detection and tracking. Computer Vision and Image Understanding, 193:102907.

Wu, J., Xu, H., Zhang, Y., Tian, Y., and Song, X. (2020). Real-time queue length detection with roadside lidar data. Sensors, 20(8):23-42.

Xiao, X., Sun, Z., and Shen, W. (2020). A Kalman filter algorithm for identifying track irregularities of railway bridges using vehicle dynamic responses. Mechanical Systems and Signal Processing, 138:106582.

Yamaguchi, K., Kato, T., and Ninomiya, Y. (2006a). Moving obstacle detection using monocular vision. In 2006 IEEE Intelligent Vehicles Symposium, pages 288-293.

Yamaguchi, K., Kato, T., and Ninomiya, Y. (2006b). Vehicle ego-motion estimation and moving object detection using a monocular camera. In 18th International Conference on Pattern Recognition (ICPR'06), volume 4, pages 610-613.

Yang, H. and Qu, S. (2018). Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition. IET Intelligent Transport Systems, 12(1):75-85.

Yang, T., Cappelle, C., Ruichek, Y., and El Bagdouri, M. (2019). Online multiobject tracking combining optical flow and compressive tracking in the Markov decision process. Journal of Visual Communication and Image Representation, 58:178-186.

Yang, Y. and Duan, Z. (2020). An effective co-evolutionary algorithm based on artificial bee colony and differential evolution for time series predicting optimization. Complex and Intelligent Systems, 6(2):299-308.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. ACM Comput. Surv., 38(4):13-58.

Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., and Chen, C. (2011). Data-driven intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, 12(4):1624-1639.

Zhao, Y., Zhu, X., Guo, W., She, B., Yue, H., and Li, M. (2019a). Exploring the weekly travel patterns of private vehicles using automatic vehicle identification data: A case study of Wuhan, China. Sustainability, 11(21):52-61.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. (2019b). Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems, 30(11):3212-3232.

Zheng, H., Chang, W., and Wu, J. (2019). Traffic flow monitoring systems in smart cities: Coverage and distinguishability among vehicles. Journal of Parallel and Distributed Computing, 127:224-237.

Zhou, Y., Dey, K. C., Chowdhury, M., and Wang, K.-C. (2017). Process for evaluating the data transfer performance of wireless traffic sensors for real-time intelligent transportation systems applications. IET Intelligent Transport Systems, 11(1):18-27.

Zhu, Y., Comaniciu, D., Pellkofer, M., and Koehler, T. (2006). Reliable detection of overtaking vehicles using robust information fusion. IEEE Transactions on Intelligent Transportation Systems, 7(4):401-414.

Zhu, Y., Comaniciu, D., Ramesh, V., Pellkofer, M., and Koehler, T. (2005). An integrated framework of vision-based vehicle detection with knowledge fusion. In IEEE Proceedings. Intelligent Vehicles Symposium, 2005., pages 199-204.

ÖZGEÇMİŞ

Adı Soyadı : Jahongir Azimjonov

ÖĞRENİM DURUMU

Derece	Eğitim Birimi	Mezuniyet Yılı
Doktora	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar Mühendisliği	Devam ediyor
Yüksek Lisans	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar Mühendisliği	2016
Lisans	Taşkent Bilişim Bilimleri Üniversitesi / Mühendislik Fakültesi / Bilgisayar Mühendisliği	2012
Lise	Andican Fen Lisesi	2008

İŞ DENEYİMİ

Yıl	Yer	Görev
2019-2020	Sakarya Üniversitesi	Araştırmacı
2015-2016	Natura Otomasyon A.Ş	Araştırmacı

YABANCI DİL

İngilizce, Rusça, Özbekçe

ESERLER (makale, bildiri, proje vb.)

1. Rule Based Metadata Extraction Framework from Academic Articles.
2. Evaluation of Distance Learning Students Performance Using Fuzzy Logıc.
3. A Real-Time Vehicle Detection and A Novel Vehicle Tracking Systems for Estimating and Monitoring Traffic Flow on Highways.

HOBİLER

Kitap okuma, spor, yüzme.