

**KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ**

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME TEZİ

**COCO VERİ SETİ İLE GERÇEK ZAMANLI NESNE TESPİT
ALGORİTMALARININ KARŞILAŞTIRILMASI**

**Cengizhan PARLAK
Mehmet Ertuğrul EVRENSEL
Enes BAŞPINAR**

KOCAELİ 2021

KOCAELİ ÜNİVERSİTESİ

**KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ**

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME TEZİ

**COCO VERİ SETİ İLE GERÇEK ZAMANLI NESNE TESPİT
ALGORİTMALARININ KARŞILAŞTIRILMASI**

**Cengizhan PARLAK
Mehmet Ertuğrul EVRENSEL
Enes BAŞPINAR**

Doç.Dr. Sevinç İlhan OMURCA

Danışman, Kocaeli Univ.

.....

Dr.Öğr.Üyesi Alev Mutlu
Jüri Üyesi, Kocaeli Univ.
Arş.Gör. Emin Ölmez
Jüri Üyesi, Kocaeli Univ.

Tezin Savunulduğu Tarih: 27.05.2021

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, canlı görüntü üzerinde çalışan nesne tespiti algoritmalarının COCO veri seti üzerinde karşılaştırılması amacıyla gerçekleştirilmiştir.

Tez çalışmamızda desteklerini esirgemeyen ve çalışmalarımıza yön veren, bize güvenen ve yürek lendiren danışmanımız Sevinç İlhan OMURCA'ya teşekkürlerimizi sunarız.

Mayıs 2021

Cengizhan PARLAK
Enes BAŞPINAR
Mehmet Ertuğrul EVRENSEL

İÇİNDEKİLER

ÖNSÖZ ve TEŞEKKÜR	i
İÇİNDEKİLER.....	ii
ŞEKİLLER DİZİNİ	iv
TABLOLAR DİZİNİ.....	viii
KISALTMALAR	ix
ÖZET	x
ABSTRACT	xi
GİRİŞ	i
1. LİTERATÜR ÖZETİ.....	ii
1.1 Benzer Çalışmalar	ii
1.2. Yöntem	xlvi
1.2.1. Görüntü iletim sistemi	xlvi
1.2.2. Görüntü işleme yöntemi	xlvi
1.3. Uygulama	lxix
2. METODOLOJİ.....	lii
2.1 Nesne Tespiti	lii
2.2 Nesne Tespiti Türleri	lii
2.2.1. Nesne tespitinin çalışması	liv
2.3 Metrikler	lv
2.4. Nesne Tespiti Algoritma Çeşitleri.....	lvi
2.4.1. İki aşamalı nesne tespiti	lvi
2.4.2. Tek aşamalı nesne tespiti	lvii
2.5. Isı haritası tabanlı nesne tespiti	lviii
3. NESNE TESPİTİNDE KULLANILAN YÖNTEMLER	lx
3.1. CNN Modelinin Gelişimi	lxii
3.2. CNN Temelli İki Aşamalı Tespit Modelleri	lxiii
3.2.1. RCNN.....	lxiii
3.2.2. SPPNet.....	lxiv
3.2.3. Fast RCNN	lxv
3.2.4. Faster RCNN	lxvi
3.2.5. Feature Pyramid Networks.....	lxvii
3.3. CNN Temelli Tek Aşamalı Tespit Modelleri	lxviii
3.3.1. You Only Look Once (YOLO).....	lxviii
3.3.2. Single Shot MultiBox Detector (SSD).....	lxviii
3.3.3. RetinaNet.....	lxix
3.4. Derin Öğrenme Algoritmalarında Tespit İşleminin Hızlandırılması	lxix
3.5. Modelin Seçimi.....	lxx
4. YOLO ALGORİTMALARININ KARŞILAŞTIRILMASI	lxxiii
4.1 YOLO Algoritması	lxxiii
4.2. YOLO Sürümleri	lxxvi

4.2.1 YOLOv3	lxxvii
4.2.2 YOLOv4	lxxvii
4.2.3 YOLOv5	lxxviii
4.2.4 PP-YOLO	lxxx
4.3. Gerçek Zamanlı Nesne Tespit Algoritma ve Sürümlerinin Performans ve Başarı Oranlarının Karşılaştırılması	lxxxi
5. SONUÇLAR VE ÖNERİLER	lxxxiv
KAYNAKLAR.....	xc
ÖZGEÇMİŞ	xcvii

ŞEKİLLER DİZİNİ

Şekil 1.1. Resim çoğaltma işlemi	2
Şekil 1.2. İHA üzerinden küçük nesnelerin tespiti	4
Şekil 1.3. Görüntü işleme ve nesne tespiti işlemleri.....	4
Şekil 1.4. Nesne tespiti yapılan kümelemeler üzerinden nesne tespiti ve kutuların çizilmesi.....	5
Şekil 1.5. Kalman filtresiyle nesne tespiti işlemleri ve takibi	6
Şekil 1.6. Görüntü işleme tekniklerinin uygulanması ve birleştirilmesi	8
Şekil 1.7. Hareketli nesnelerin tespiti ve takibi işleminin algoritması	10
Şekil 1.8. Resim konvolüsyon işlemi (Kim vd. (2013))	14
Şekil 1.9. Farklı ölçeklerde R-CNN ile seçici arama örneği	15
Şekil 1.10. Nesne Tespiti için Blok Diyagram Gösterimi.....	17
Şekil 1.11. Sinir ağ modeli ve derin öğrenme framework’ü kullanan takımların dağılımı.....	19
Şekil 1.12. İlk 3 GPU katılımcılarının sinir ağları yapıları.....	20
Şekil 1.13. GPU katılımcılarının genel sonuçları	21
Şekil 1.14. Pascal VOC2007 verileri üzerinde test edilen farklı nesne algılama algoritmalarının karşılaştırması (eğitimler VOC2007 ve VOC2012 verilerinde yapılmıştır.).....	22
Şekil 1.15. Giriş çözünürlüğünün doğrulukla karşılaştırılması	24
Şekil 1.16. Giriş çözünürlüğünün algılanan kare oranına etkisi.....	24
Şekil 1.17. NVIDIA Jetson TX2’de çalışan algoritmaların, farklı çözünürlüklerdeki genel hassasiyet ortalamaları	25
Şekil 1.18. Ortofoto görüntüsü	32
Şekil 1.19. SYM görünümü	33
Şekil 1.20. Normalize edilmiş Sayısal Yüzey Modeli	33
Şekil 1.21. Heumoes heyelan mozaiği görüntü sınırları.....	34
Şekil 1.22. Heumoes heyelanının kesintisiz mozaığı	35
Şekil 1.23. Süper Sauze heyelanının DTM’si	35
Şekil 1.24. Boş ve inceleneyecek istasyon görüntüleri	37
Şekil 1.25. Histogram analizi, görüntü seçimi, RGB değerleri	39
Şekil 1.26. İHA sistemi üzerinde kullanılan yer kontrol istasyonu	39
Şekil 1.27. YOLO nesne tespiti örneği	41
Şekil 1.28. YOLO modelindeki izgara yapısı.....	41
Şekil 1.29. YOLO modelinde sınırlayıcı kutu	41
Şekil 1.30. YOLO modelinin örnek yapısı	42
Şekil 1.31. YOLO modelinde tespit edilen nesnelerin etiketlenmesi	43
Şekil 1.32. YOLO modelinin evrişimli sinir ağındaki yapısı	43
Şekil 2.1. Nesne tespiti algoritma yapısı	50
Şekil 2.2. Intersection over Union	51
Şekil 2.3. YOLO algoritmasının işleyiş düzeni	52
Şekil 2.4. Isı haritası tabanlı nesne algılama modeli	53
Şekil 3.1. Nesne tanıma ve nesne tespiti işlemlerinin gerçekleştirimi	54
Şekil 3.2. Derin öğrenme algoritmalarının gelişimi	56

Şekil 3.3. SPPNet modelinin diğer modellerden farklı işleyen havuz katmanı	58
Şekil 3.4. Evrişimli nöron ağlarının temel işleyiş adımları	58
Şekil 3.5. Özellik Piramit Ağlarında ölçeklendirilen her orijinal resim tahmin aşamasında tekrar işleme dahil edilir.....	60
Şekil 4.1. YOLO algoritmasının işleyiş düzeni	64
Şekil 4.2. Sınırlayıcı kutu tanımı	65
Şekil 4.3. Sınırlayıcı kutuların matrisi	65
Şekil 4.4. Non-max suppression metoduyla sınırlayıcı kutuların azaltılması	66
Şekil 4.5. YOLOv3’ün diğer modellerle karşılaştırılması	66
Şekil 4.6. YOLOv4’ün diğer algoritmalarla karşılaştırılması.....	67
Şekil 4.7. YOLOv5’in varyasyonlarının karşılaştırılması	68
Şekil 4.8. PP-YOLO’nun diğer varyasyonlarla karşılaştırılması.....	69

TABLOLAR DİZİNİ

Tablo 1.1. CNN sınıflandırma algoritmalarının karşılaştırmalı örneği.....	16
Tablo 1.2. Gerçek zamanlı nesne tespitinde kullanılan algoritmaların karşılaştırılması.....	41
Tablo 4.1. YOLOv3 performans tablosu.....	67
Tablo 4.2. YOLOv4 performans tablosu.....	68
Tablo 4.3. YOLOv5 performans tablosu.....	69
Tablo 4.4. PP YOLO performans tablosu	70
Tablo 4.5. SSD ve R-FCN algoritmalarının performansı	70
Tablo 4.6. RetinaNet algoritmasının performansı.....	71
Tablo 4.7. EfficientDet algoritmasının performansı	71
Tablo 4.8. YOLO algoritmalarının performansı	72
Tablo 5.1. Nesne tespit algoritmalarının performanslarının karşılaştırılması.....	74

KISALTMALAR

İHA	: İnsansız Hava Aracı
UAV	: Unmanned Aerial Vehicle (İnsansız Hava Aracı)
FPS	: Frames Per Second (Saniye başına kare sayısı)
ML	: Machine Learning (Makine öğrenmesi)
CNN	: Convolutional Neural Network (Evrişimli Nöron Ağısı)
SSD	: Single Shot Detection (Tek Atış Tespit Modeli)
RGB	: Red-Green-Blue (Kırmızı, yeşil, mavi renk modeli)
HSV	: Hue, Saturation, Value (Ton, doygunluk, değer renk modeli)
MODAT	: Moving Object Detection and Tracking (Hareket eden nesne tespiti ve takibi)
KLT	: Kanade Lucas Tomasi (Kanade Lucas Tomasi özellik izleyicisi)
SURF	: Speeded up robust features (hızlandırılmış sağlam özellikler)
RSC	: Random sample consensus (Rastgele örnek uzlaşması)
IoU	: Intersection over Union (Birleşimlerin kesimimi)
AP	: Average Precision (Ortalama hassasiyet)
AR	: Average Recall (Ortalama geri çağrıma)
mAP	: mean Average Precision (Ortalama hassasiyetlerin ortalaması)
DL	: Deep Learning (Derin öğrenme)
RCNN	: Region-based Convolutional Neural Network (Bölge Temelli Evrişimli Nöron Ağısı)
RoI	: Region of Interest (İlgili bölgesi)
RPN	: Regional Proposal Network (Bölge teklif ağısı)
YOLO	: You Only Look Once (Sadece Bir Kez Bak)
HOG	: Histogram of Oriented Gradients (Yönelimli Gradyanların Histogramı)
DPM	: Deformable Part Models (Deforme Olabilen Parçalar Modeli)
ReLU	: Rectified Linear Units (Düzeltilmiş Lineer Birimler)
GPU	: Graphics Processing Unit (Grafik İşlemci Birimi)
SVM	: Support Vectore Machine (Destek Vektör Makineleri)
SPPNet	: Spatial Pyramid Pooling (Mekansal Piramit Havuz Ağları)
FPN	: Feature Pyramid Networks (Özellik Piramit Ağları)
RFCN	: Region-based Fully Convolutional Network (Bölge-temelli Tamamen Bağlı Evrişim Ağları)
BoF	: Bag of Feature (Özelliklerin toplamı)
BoS	: Bag of Sample (Örneklerin toplamı)
CSP	: Center and Scale Prediction (Merkezle ve ölçekte tahmini)

COCO VERİ SETİ ÜZERİNDEN GERÇEK ZAMANLI NESNE TESPİT ALGORİTMALARININ KARŞILAŞTIRILMASI

ÖZET

Günümüzde İnsansız Hava Araçları maliyet, kullanım kolaylığı ile birlikte tüm yönleriyle değerlendirildiğinde: bireysel, özel veya kamusal alanlarda kendilerine büyük bir kullanım yeri bulabilmektedir. Bu kullanımların büyük bir kısmında sistemden alınan görüntünden nesnelerin tespit ve takibi işlemi bahse konu olmaktadır. Bu tespit işlemi için zamanla çeşitli algoritmalar ve modeller geliştirilmiştir. Bu algoritma ve modeller, farklı kapsam ile karmaşıklıktaki problemleri hedef alıp bunlara yönelik çözümler sunmaktadır.

Güncel kullanımda olan nesne tespit algoritmalarının çoğunda evrişimli nöron ağları tercih edilmektedir. Fast R-CNN, Faster R-CNN, SSD, R-FCN ve YOLO gibi nesne tespit algoritmaları da bu evrişimli nöron ağlarına dayanmaktadır. YOLO algoritması diğer nesne tespit algoritmalarına kıyasla hesaplama hızı yüksek olmasına kıyasla başarı oranının yeterli seviyede kalması sebebiyle her seviyedeki uygulamalarda kullanımı yaygındır.

Bu çalışmada, nesne tespiti algoritmaları ile bunların nasıl gerçekleştiğine yer verilmekte, nesne tespiti konusu detaylıca incelenmekte ve YOLO modelinin varsayınlarıyla birlikte, diğer gerçek zamanlı nesne tespit algoritmalarının da başarı oranı ve hız ölçütleri karşılaştırılmıştır.

Anahtar kelimeler: Bilgisayarlı Görü, Görüntü İşleme, Nesne Tespit, Makine Öğrenmesi, Derin Öğrenme

COMPARISON OF REAL-TIME OBJECT DETECTION ALGORITHMS VIA COCO DATASET

ABSTRACT

Nowadays, Unmanned Aerial Vehicles can be used in a wide range scenarios through individual use to private or government based usages. This common use of UAVs is a result of relatively cheap costs (compared to benefits and other options' costs), ease of use and all the other proven and successful aspects combined. In most scenarios that include UAVs, video or image feeds retrieved from UAV cameras process in the ground station by machine learning algorithms and object detection as well as tracking these objects can occur through these algorithms. Late detection algorithms aim for high success rates and high detection speeds for various problems, in different datasets, input specifications and even specialized for detecting specific objects.

Convolutional Neural Networks are preferred in most of the object detection algorithms in the time being. Algorithms such as Fast R-CNN, Faster R-CNN, SSD, R-FCN and YOLO are based on these Convolutional Neural Networks. YOLO algorithm is commonly preferred in late applications due to its light-weight nature, high detection speed and a little compromise from the success rate compared to improvements in the other mentioned aspects.

In this study, the object detection concept itself, object detection algorithms, including how they work and what are the advantages as well as disadvantages of these algorithms are discussed and different variations of YOLO model and other real-time object detection algorithms are compared based on accuracy and speed in detail.

Keywords: Computer Vision, Image Processing, Object Detection, Machine Learning, Deep Learning

GİRİŞ

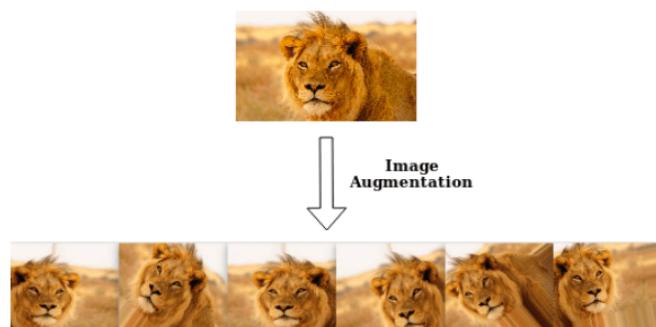
Son günlerde askeri alanda başarılarından hepimizin fazlasıyla göz aşinalığı kazandığı İnsansız Hava Aracı teknolojisinin yıldızı parlamaktadır. İnsansız hava araçları, ortaya çıktığı zamanlarda kumanda vasıtasyyla insanlar tarafından kullanılmaktaydı. Ancak günümüzde her alanda kendisini hissettiren insansız teknolojilerin gelişmesiyle birlikte her alanda görebilmektedir. Bunun faydasını ise insan sağlığının önemli olduğu ve insan gücünün yeterli olmadığı alanlarda oldukça yoğun bir şekilde fark ediyoruz. İnsansız hava araçları da bu bahsedilen gelişmelerden olumlu yönde etkilenmektedir.

Buna paralel olarak nesne tespiti ve nesne takibi algoritmaları da hızlıca gelişim gösterdi. Ancak insansız hava araçlarından gelen görüntülerde hala nesne izleme zorlanılan alanların başında gelmektedir. Uçağın aerodinamiği sebebiyle ağırlık kısıtı bulunmaktadır. Bu, istenen ekipmanların kullanılabilmesine engel olmaktadır. Dolayısıyla yüksek performansta olan kameraların seçilememesi çözünürlüğün ve FPS'in düşük olmasında neden olmakta; görüntü işleme için kullanılan kartın da performanslı seçilememesine neden olmaktadır. Drone'lar yerine sabit kanatlı insansız hava araçlarının kullanılması ise durağan kalmamasından dolayı görüntünün kısa sürede işlenmesini gerektirdiği için işleme hızının çok daha hızlı olmasını gerekmektedir.

1. LİTERATÜR ÖZETİ

1.1 Benzer Çalışmalar

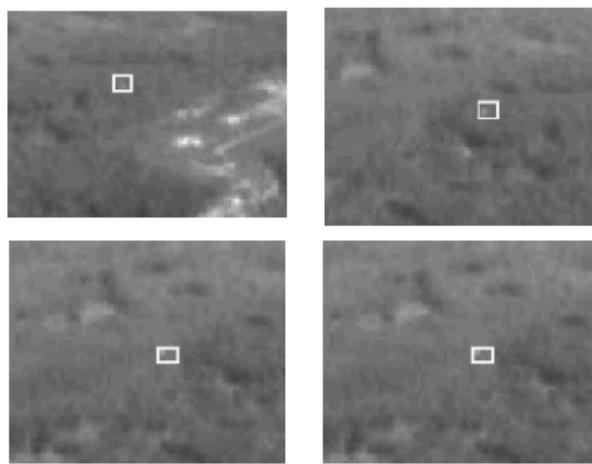
Berat Mert Alibaba ve Sedat Özer tarafından yazılan “SyNet: An Ensemble Network for Object Detection in UAV Images” [36] isimli makalede, çok nesneli algılama algoritmalarının avantajlı özelliklerinin birleştirilmesi ile daha yüksek bir performans elde edilmesi üzerine debynmiştir. Derin öğrenme tabanlı nesne algılama algoritmaları, tek aşamalı dedektörler (single-stage detector) ve çok aşamalı dedektörler (multi-stage detector) olmak üzere iki ana kategori içerisinde sınıflandırılabilir. Ancak birbirine göre artıları ve eksileri vardır. Makalede önerilen SyNet ağı ise bunların iyi yanlarını, tek-aşamalı dedektörün daha az false-negative oranını ve çok-aşamalı dedektörün daha yüksek kaliteye sahip olmasını birleştirerek daha iyi bir yöntem sağlar. Havadan görüntülerde daha da iyi sonuçlar elde etmek için Cascade R-CNN ve CenterNet algılama algoritmalarının (detection algorithm) iyi özellikleri ele alınır. Intersection over Union (IoU) metriği kullanılarak algılanması gereken küçük nesnelerde performansı ölçmek için önerilen çözümde Cascade R-CNN algoritmasından faydalılmıştır. Buna ek olarak, CenterNet’te algılanan nesnelerin merkezine yönlendirilir ve bu sayede farklılıklarını ölçeklendirmek için daha güvenilir bir yöntemin oluşturulması sağlanmıştır. Önerilen yaklaşım iki adım içermektedir. İlk veri setindeki dengesizlik sorununu çözmek için resim çoğaltma (image augmentation) işlemidir.



Şekil 1.1. Resim çoğaltma işlemi [36]

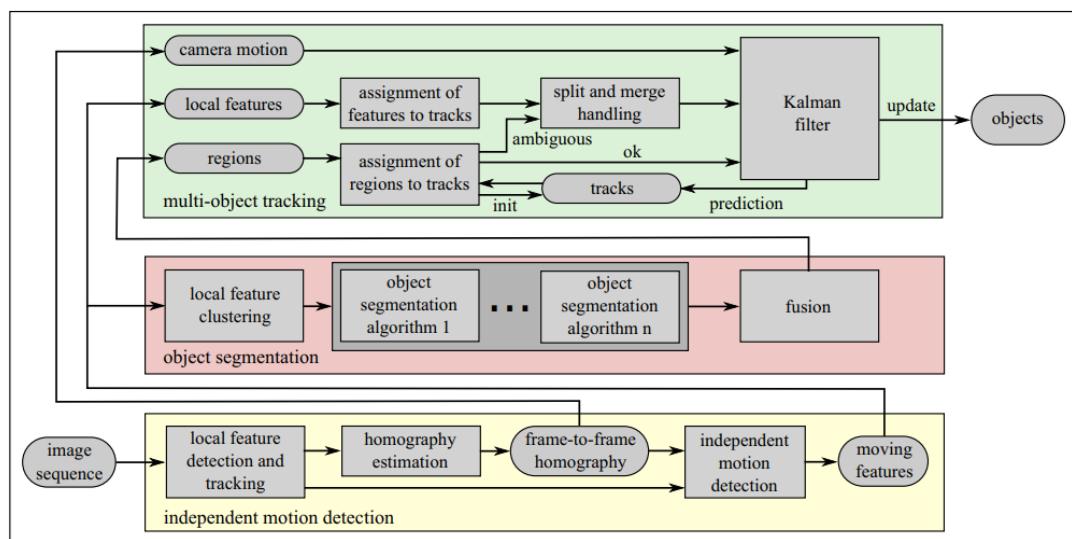
İkincisi ise dedektörlerin tahminlerini birleştirmek için ağırlıklı sınırlayıcı kutu füzyonu (weighted bounding box fusion) yöntemi kullanılarak en güvenilir skoru belirlemektir. Ve bu yeni yöntem, MS-COCO (zemindeki görüntüler) ve visDrone (havadan çekilmiş görüntüler) veri setlerinde test edilmiştir. Sırasıyla %52.1 mAPIoU=0.75 ve %52.1 mAPIoU=0.75 doğruluk sonuçları elde edilmiştir.

Nils Tijtgat, Bruno Volckaert ve Filip De Turck tarafından yazılan “Object Tracking in Unmanned Aerial Vehicle (UAV) Videos Using a Combined Approach” [37] isimli makalede, küçük nesnelerin uzay-zamansal bölüme yöntemi (spatio-temporal segmentation method) kullanılarak izlendiği ve daha büyük nesnelerin değiştirilmiş bir istatistiksel deformable model kullanılarak izlendiği İHA videolarında nesne takibi için bir birleşik yaklaşım önermektedir. Bilindiği gibi İHA videolarının, ani hareket, düşük çözünürlük, gürültülü görüntüler, dağınık arka plan, düşük kontrast ve küçük hedef boyutu nedeniyle işlenmesi zor olmaktadır. İşlemlerin yalnızca küçük bir görüntü bölgesinde gerçekleştirilmesi ve “fast snake deformation method” kullanılarak hesaplama karmaşıklığı önemli ölçüde azaltılmıştır. İHA'daki hızlı hareket eden kamera, genellikle harekette ani kesintilere neden olmaktadır. Bu da İHA görüntülerinde hareketli veya sabit hedefleri izlemeyi çok zor bir konu haline getirir çünkü doğru hareket hesaplaması zorlu bir sorun olmuştur ve olmaya devam etmektedir. İzlemeyi daha da zorlaştıran diğer faktörler arasında düşük çözünürlük, gürültülü görüntüler, dağınık arka plan, tıkanma, lçek değişikliği, düşük kontrast ve hedefin küçük boyutu sayılabilir. Mevcut izleme algoritmalarının çoğu, net özelliklere ve sınırlara sahip büyük nesneler için tasarlanmıştır. Küçük hareketli nesnelerin genellikle farklı bir algoritma kullanılarak işlenmesi gerekmektedir. Bu makalede, küçük nesnelerin uzay-zamansal bölüme yöntemi kullanılarak izlendiği ve daha büyük nesnelerin değiştirilmiş bir istatistiksel deformable model kullanılarak izlendiği İHA videolarında nesne takibi için birleşik bir yaklaşım önerilmiştir.



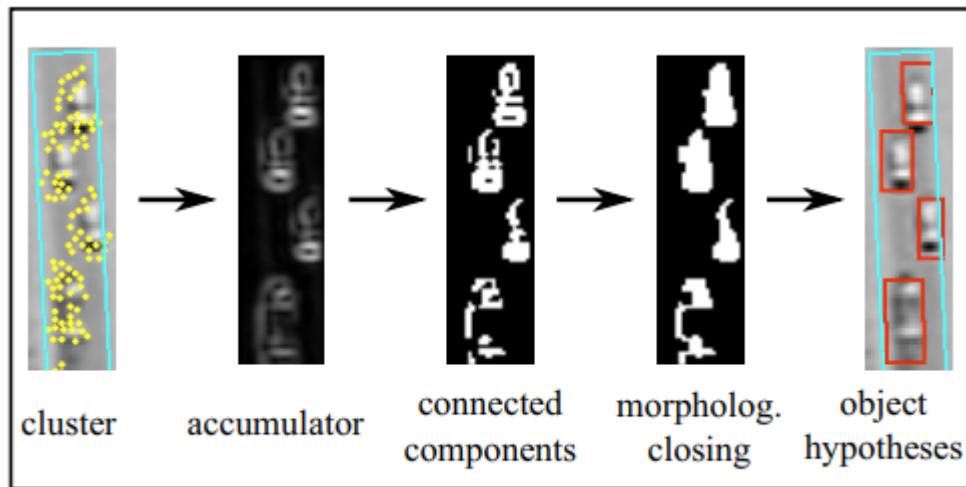
Şekil 1.2. İHA üzerinden küçük nesnelerin tespiti [37]

Michael Teutsch ve Wolfgang Kruger tarafından yazılan “Detection, Segmentation, and Tracking of Moving Objects in UAV Videos” [38] isimli makalede, sabit olmayan yerel görüntü özelliklerinin kümelenmesi ve ardından nesne segmentasyonu yoluyla, nesne hipotezlerini temsil eden bölgeler oluşturulur tespit edilmesine degenilmiştir ve çoklu nesne izleme, bir Kalman filtresi kullanılarak ve kamera hareketi dikkate alınarak tanıtılmıştır. Tekli veya çoklu hareketli nesnelerin doğu bir şekilde izlenmesi önemlidir ancak elde edilmesi sırasında zorluklarla karşılaşılmaktadır. Kamera hareketi, görüntüde yalnızca birkaç pikselin küçük nesne görüntümleri, değişen nesne arka planı, nesne birleştirme, gölgeleme veya gürültü bu zorlukların öne çıkanlarıdır.



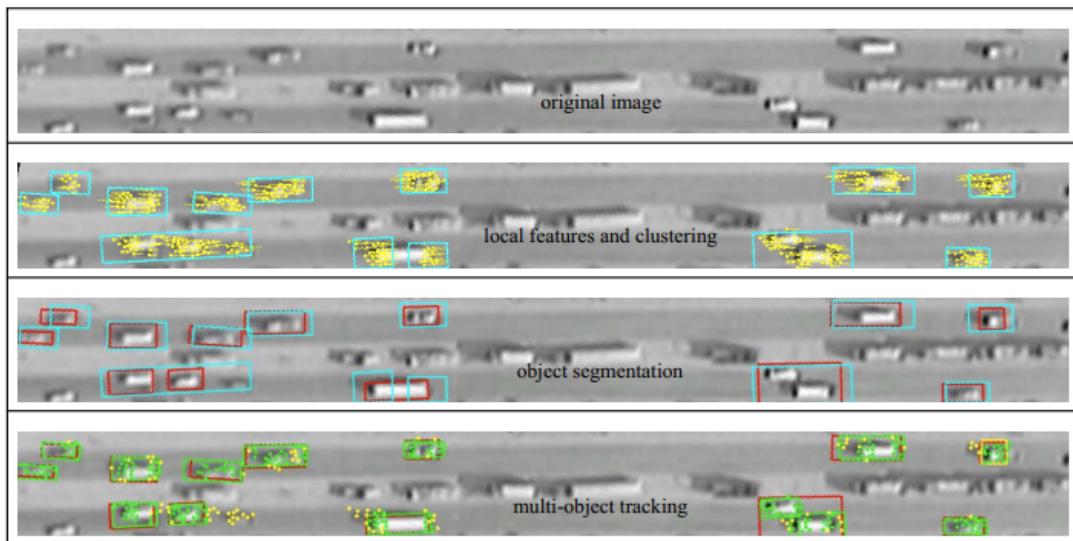
Şekil 1.3. Görüntü işleme ve nesne tespiti işlemleri [38]

Çalışmada yere dik olarak yönlendirilmiş görsel-optik kamera ile küçük bir İHA kullanılır ve bu zorluklarla ilgilenen çoklu nesne takibi için üç katmanlı bir işleme zinciri sunulmuştur. İlk katmanda, yerel görüntü özellikleri tespit edilir ve sabit ve hareketli özellikler olarak kategorize edilir. Nesne hipotezleri, ikinci katmanda, özellik kümelemesi ve görünümeye dayalı nesne segmentasyonu hareket ettirilerek oluşturulur. Üçüncü katmanda, birinci ve ikinci katmanın çıktıları, araçlar birbirini geçtiklerinde ortaya çıkan bölünme ve birleştirme durumlarının işlenmesi dahil olmak üzere izleme için kullanılır. Öncelikle bağımsız hareket algılama katmanında, izlenen sahnenin statik arka planına göre bağımsız hareketi gösteren yerel görüntü özellikleri tespit edilir. Köşe özellik takibi (corner feature tracking), çerçeveden çerçeveye hizalamanın yanında yerel görüntünün küresel görüntü dönüşümleri olarak homografları tahmin etmek için kullanılır. Hareket eden nesnelerdeki özellikleri statik arka plandaki özelliklerden ayırmak için bağımsız hareket algılanır. Çoklu görüntüye sahip geometrik kısıtlamalarla paralaks ayırtırmaları yerine homograflar kullanmak yeterlidir, çünkü görüntülerimiz esas olarak daha yüksek irtifalarda çalışan İHA'lardan alınmıştır. Bağımsız hareket algılama esnasında, özelliklerden görelî hızları tahmin etmeleri gereklidir ve çok nesneli izleme katmanında, homograflar Kalman filtresi için kontrol girdisi oluşturmak için kullanılır. Bağımsız hareket algılama katmanının ana çıktısı, statik arka plana göre hareketli olarak sınıflandırılan yerel görüntü özellikleridir. Her özelliğin çıktı nitelikleri, görüntü konumu ve göreceli hızdır.



Şekil 1.4. Nesne tespiti yapılan kümelemeler üzerinden nesne tespiti ve kutuların çizilmesi [38]

Çoklu nesne izlemede ise nesnelerin takibinin birbirine karışmasının önlenmesi gereklidir. Çalışmada obje ve kamera hareketi çoğunlukla doğrusal olduğu için Kalman filtresine karar verilmiştir. Üstelik uygulaması kolay ve hızlıdır. Kalman滤resi tarafından beş parametreye bakılır: nesne merkezi, boyut ve yönelim. Ardından sınırlayıcı kutuların atanma aşamasına geçilir. Bölgenin sınırlayıcı kutu kesim alanı ve Kalman tahmini için minimum eşik aşılırsa, bir bölge belirli bir nesneyi takip etmek için atanır.

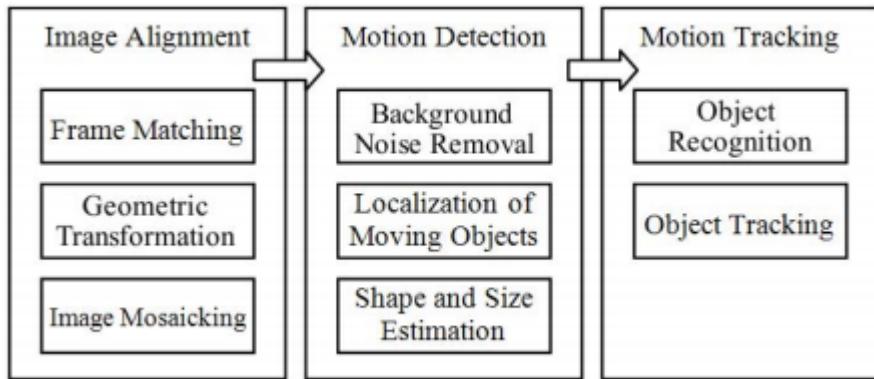


Şekil 1.5. Kalman filtresiyle nesne tespiti işlemleri ve takibi [38]

Michael Teutsch ve Wolfgang Kruger tarafından yazılan “Implementation of Tracking of a Moving Object Based on Camshift Approach with a UAV” [39] isimli makalede, C dili ile OpenCV kütüphanesi kullanılarak 650x360 çözünürlükte ve 30 fps kare hızındaki görüntü ile nesne takibi üzerinde durulmaktadır. Nesne izleme, nesne hareket ederken sonraki hareketlerini tahmin etme süreci olarak tanımlanabilir. Bu sürecin büyük bir kısmının, gerçek zamanlı kısıtlamaları karşılayan ve nesnenin hareket değişiklikleri, sahne aydınlatması, ölçek ve görünüm ve görüntü açısı gibi zorlu olan güvenilir izleme yöntemlerine sahip olması gereklidir. Problemin karmaşıklığını azaltmak için nesnelerin sayısı ve boyutu hakkındaki ön bilgiler de kullanılabilir. Bazı algoritmalar, nesnenin görünümü veya şekli, dokusu, rengi gibi özellikler kullanır. Çoğu izleme algoritması genellikle bu özelliklerin bir kombinasyonunu kullanır ve önerilen budur. Örneğin, KLT, kalmanfiltresi, kayma ve Camshift anlamına gelir. Nesne izleme iki adımdan oluşur: önce izlenen nesne seçilir ve ardından Camshift algoritması işlenir. Camshift algoritmasının prensibi Meanshift algoritması prensibine dayanmaktadır. Statik dağılımlar için Meanshift algoritması kullanılırken, dinamik dağılımlar için Camshift algoritması kullanılmaktadır. Bu fark, Camshift algoritmasının her kare için dağılımları yeniden hesaplamasına izin verir ve bu da onu videoda faydalı kılar çünkü bu, algoritmanın çerçeveler arasında bir nesneyi sürekli olarak izlemek için nesne hareketini tahmin etmesine olanak tanır. Bu nedenle, Camshift algoritmasının hareketli nesnelerin izlenmesinde etkili olduğu düşünülmektedir. Renk özelliklerine dayanmaktadır. RGB renk uzayının aydınlatma parlaklığındaki değişikliklere daha duyarlı olması nedeniyle Camshift algoritması, ışık yoğunluğundaki değişikliklerin izleme performansı üzerindeki etkisini azaltmak için görüntüleri RGB renk uzayından HSV renk uzayına dönüştürür. Camshift Algoritması şu şekilde çalışır: 1. Öncelikle, arama penceresinde izlenecek konumu seçin. Bu, kaydırılmış arama penceresidir. Renk tonu bileşenini izlenecek nesnenin HSV renk uzayından ayarlar. Bu ton, bir renk çarkı veya benzer bir renk seçim yöntemi kullanılarak da ayarlanabilir. 2. Araç kaydırma penceresinde ortalanmış olarak seçilen alanın olasılık dağılımını hesaplayın. Bu, nesneyi temsil eden bir renk histogramı olarak temsil edilir. Seçilen bölgede renk tonu bileşenine sahip piksel miktarı histogram aracılığıyla görülebilir. 3. Olasılık görüntüsünün ağırlık merkezini

bulmak için araç kaymasını yineleyin. Bu nokta, sıfırıncı an olarak kullanılır. Bu, arama penceresinin yeni merkez noktası olacaktır. 4. Bir sonraki video karesi için, arama penceresini yeni merkezde ortalayın ve ardından işlemi tekrarlamak için ikinci adıma gidin. I (x, y), arama penceresi içindeki (x, y) noktasındaki ve arama penceresi üzerindeki x ve aralıktaki ayrık olasılık görüntüsünün yoğunluğudur. Bu şekilde nesne izlemesi gerçekleştirilmiş olunur.

Aryo Wiman Nur Ibrahim, Pang Wee Ching, Gerald Seet G.L., Michael Lau W.S. ve Witold Czajewski tarafından yazılan “Moving Objects Detection and Tracking Framework for UAV-based Surveillance” [40] isimli makalede, İHA’dan nesne tespitinin zorluklarını, görüntü mozaikleme yoluyla ziyaret edilen alanları haritalayarak havadan görüntülerden yararlı bilgiler toplamak ve yakalanan videodaki hareketli nesneleri tespit etmek için bir sistem geliştirmek istenmiştir. Bunu yapmak için, çeşitli görüntü işleme tekniklerinin uygulanmasını ve birleştirilmesini kolaylaştırmak için Hareketli Nesneleri Algılama ve İzleme (MODAT) çerçevesi geliştirilmiştir. Bu çerçevenin ana amacı, insan operatörü video görüntüsündeki küçük hareketli nesneleri manuel olarak izleme ve izleme zorunluluğundan kurtarabilecek bir sistem uygulamaktır. MODAT çerçevesinin İHA videosuna uygulanması, dinamik arka planı ön plandan ayırt etme temel problemini daha basit problemlere ayırmayı içerir. Soruna sistematik adım adım yaklaşım, görüntü dizisindeki istenmeyen hareketleri ortadan kaldırın video görüntülerinin dengelenmesiyle başlar. İkinci işlem, hareketli hedeflerin tespitini kolaylaştırmak için stabilize arka plandan hareket segmentasyonudur. Son olarak, bölgelere ayrılmış hedef boyut ve şekil gibi özellik özelliklerine göre sınıflandırılır, böylece hareketli hedefler çeşitli izleme yöntemleri kullanılarak izlenebilir.



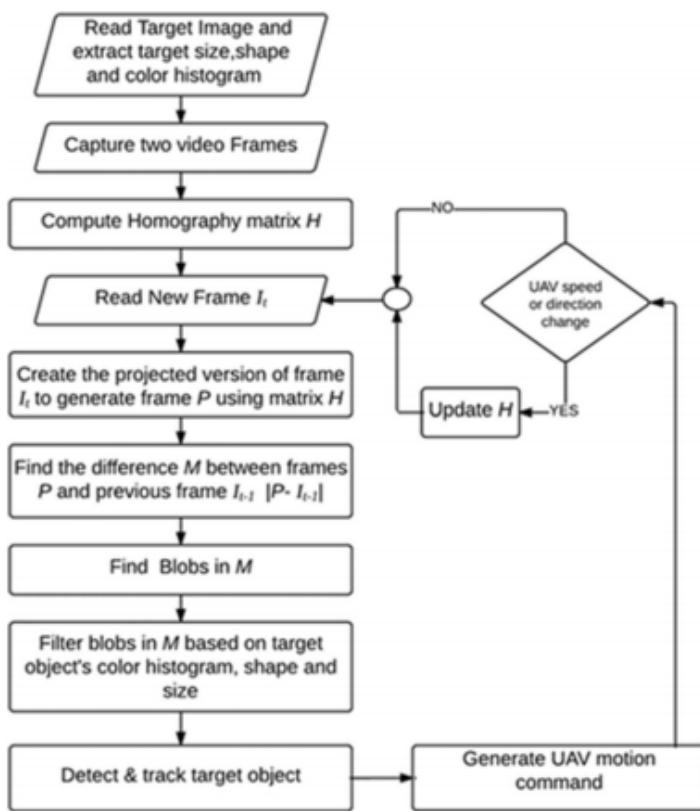
Şekil 1.6. Görüntü işleme tekniklerinin uygulanması ve birleştirilmesi [40]

Görüntü Hizalama Modülünün iki işlevi vardır. İlk işlevsellik, havadan görüntünün dinamik arka planının statik olana ayarlanması için ego-hareket dengelemesidir. İkinci olarak, bu modül, ayrı ayrı yakalanan görüntülerin bir kombinasyonundan global görüntüyü oluşturan mozaikleme işlemini gerçekleştirecektir. Hizalamanın iyi bir sonucu şunları üretebilir: (i) yakalanan görüntülerdeki hareketli nesneleri tespit etmek için yararlı olan hizalanmış görüntüler (yeterli ego-hareket telafisinden elde edilir) ve (ii) daha yüksek farkındalık yaratan pürüzüsüz mozaik-ked görüntüler gözlenen ortamın gelişmiş değerlendirmesini sağlar. Şekil 1'de gösterildiği gibi, bir Görüntü Hizalama Modülünün üç bileşeni vardır: Çerçeve Eşleştirme, Geometrik Dönüşüm ve Görüntü Mozaikleme. Ardışık kareleri hizalamak için, yakalanan görüntülerde mevcut ve geçmiş görüntü çerçeveleri arasında eşleştirilebilecek ortak bilgilerin olması gereklidir. Özellik tabanlı eşleştirme tekniği için SURF kullanılmıştır. Ardışık kareler arasında kare eşleştirme gerçekleştirdikten sonra, çıkarılan eşleşme noktaları, önceki karelerle hizalanabilmesi için görüntüyü bükmek için bir geometrik dönüştürme matrisini hesaplamak için kullanılacaktır. RANSAC, homografi tahmini için sağlam bir yöntemdir. MODAT çerçevesinde kullanılan hareket algılama yaklaşımı, COCOA sisteminden büyük ölçüde esinlenmiştir. Ancak, arka plan çıkarma veya çerçeve farklılaştırma gibi yaygın hareket algılama teknikleri, hizalanmış görüntülerde mükemmel şekilde çalışmaz. Bunun nedeni, hizalamada ki belirsizlik ve / veya diğer istenmeyen yapılardan kaynaklanan arka plan gürültüsüdür. Bu nedenle, bu tür gürültüyü filtrelemek için MODAT çerçevesine Dinamik Arka Plan Gürültüsü Giderme (Dynamic Background Noise Removal)

tekniği dahil edilmiştir. Geçerli çerçevede bir nesnenin hareket ettiği tespit edildiğinde, bu nesnenin önceki çerçevede olup olmadığını belirlemek için eşleştirme gerçekleştirilir. Nesne önceki karede bulunabiliyorsa (çünkü nesnenin o karede bulunan karakteristik özelliği ile bir eşleşme olduğundan), hedefe bir izleme yöntemi uygulanacaktır. Aksi takdirde, sistem nesnenin özelliklik karakteristik özelliklerini tanımlayacaktır.

Karam M. Abughalieh, Belal H. Sababha ve Nathir A. Rawashdeh tarafından yazılan “A video-based object detection and tracking system for weight sensitive UAVs” [41] isimli makalede, ağırlığa duyarlı İHA platformlarına monte edilebilecek kompakt boyutlu bir takip ve yönlendirme sistemi sunulmuştur. Sistem, bir İHA'nın görüş alanındaki nesneleri algılamak ve izlemek için renk filtrelemenin yanı sıra statik olmayan kameralarla kullanılabilen bir hareket algılama tekniğini birleştiriyor. Bu hibrit sistem, bir İHA kamerası ile çekilen düşük çözünürlüklü görüntüler için güvenilir bir izleme sistemi sağlar. Nesne izleme sistemlerinin iki ana bileşeni vardır: bir nesne algılayıcı ve bir nesne izleyici. Dedektörün rolü, hedef nesneyi video karelerinin her birinde veya nesne sahneye girdiğinde bulmak ve ardından izleyicinin sonraki video karelerinde izlemesi için onu etiketlemektir. Önerilen bu sisteme dedektör, önceden tanımlanmış hedef nesneyi bulmak için iki koşulu inceler: hareket algılama ve nesne renk histogramı. Hareket ve renk histogram projeksiyon filtrelerine bağlı olarak iki maske oluşturulur. İlgili nesneyi bulmak için her iki maskenin aday nesnelerinin kombinasyonu. Hareketli nesnelerin tespiti, basit çerçeve farklılaştırması kullanılarak gerçekleştirilebilir, ancak kamera hareketli bir platform üzerinde olduğundan, çıkarılan çerçeveler hizalanmadığından çıkışma sonucunda büyük miktarda gürültü olacaktır. Gürültüyü azaltmak ve daha iyi bir hareket algılama maskesi üretmek için, kare farklılaştırma işleminden önce bir kamera hareket dengeleme yöntemi uygulanır. Kamera hareket dengeleme yöntemleri, çerçeveler arasındaki piksel yer değiştirmelerini açıklayan matematiksel modeli bularak bitişik çerçeveleri hizalar. İki çerçeve arasındaki dönüşü ve ötelemeye tanımlayabilecek bir model, homografi adı verilen bir bijeksiyondur. Dört nokta kullanılarak elde edilen homografi aktarımının ancak tüm çerçeveye uygulanabilmesi

gerceği ilgilenilen bölgenin boyutunu ve şeklini sınırsız ve çok geniş yapar. Daha kaliteli bir transfer matrisi elde etmek için, daha yüksek kalitede eşleştirme noktaları gereklidir. Bu nedenle, anahtar nokta sayısını en üst düzeye çıkarırken nokta dedektörü tarafından taranan bölgeyi olabildiğince küçük tutmak için daha geniş bir ROI seçilir. Bir araya getirilen iki yöntem, daha düşük sayıda yüksek kaliteli anahtar nokta üreticek ve bu da ana nokta tanımlamasının bir sonraki aşamasında işleme maliyetinin düşmesine neden olacaktır. Çalışmada önerilen sistemin işleyişi aşağıdaki akış şemasındaki gibidir.



Şekil 1.7. Hareketli nesnelerin tespiti ve takibi işleminin algoritması [41]

İlk olarak hedef görüntü sistem tarafından okunur. Sistem, hedef görsel bilgiyi şu şekilde çıkaracaktır: genişlik / uzunluk oranı cinsinden şekil; renk histogramı; piksel cinsinden boyut. Daha sonra sistem, maskelenmiş alandaki anahtar nokta detektör kullanılarak hesaplanan homografi matrisini kullanarak her yeni çerçeveyi bir öncekiyle hizalayarak hedef için video dizisini taramaya başlayacaktır. Hizalama işleminden sonra, çerçeve farkı, hizalanan çerçeve ile önceki referans çerçevesi

arasında hesaplanır. Bu fark çerçevesi, hedefi ve diğer hareketli nesneleri içeren bloblar oluşturur. İlk adımda çıkarılan hedef görsel bilgilere dayanarak, bu bloblar, hedef nesneyi bulmak için filtrelenir. Şekil 3'te gösterildiği gibi kilit penceresindeki hedef konuma bağlı olarak İHA yön sinyalleri üretilir.

Bu çalışmada önerilen yöntemler OpenCV kitaplıklarını ile C ++ kullanılarak uygulanmıştır. Ek olarak, SIFT, SURF ve FAST algoritmaları için OpenCV kütüphaneleri kabul edildi. Sonuçlar, SURF'un anahtar noktaları çıkarma görevini beş kat daha hızlı bitirdiğini (yani %25 boyut maskesi üzerinde çalışırken), SIFT'in ise 1,6 kat daha hızlı bitirdiğini gösteriyor. FAST, görevi 15 kat daha hızlı yaptı. Bu sonuçlara dayanarak, SURF ve FAST algoritmaları SIFT'den daha hızlı performans gösterdi ve bu nedenle video dizisi işlemede kullanılıyor. Çerçeve hizalamadan sonraki aşama, nesneyi algılamak ve video karelerinde izlemektir. Nesne tespiti, hareketli blobları bulmak için çerçeve farklılaştırması uygulanarak yapılır. Hareket eden lekeler, hedef nesneyi bulmak için renk histogramı geri projeksiyonu aracılığıyla filtrelenir. Nesne algılamasından sonraki son aşama, nesneyi çerçeveler boyunca takip etmektir. Takip, kontur ağırlık merkezleri arasındaki minimum mesafeyi bularak tespit edilen nesneye en yakın aday nesneyi bularak yapılır. Önerilen algoritma, çerçeve hizalama için 13,1 fps ve nesne algılama dahil olmak üzere toplam 10,86 fps işlemeyi işledi.

Cazzato, D.; Cimarelli, C.; Sanchez-Lopez, J.L.; Voos, H.; Leo, M. tarafından yazılan "A Survey of Computer Vision Methods for 2D Object Detection from Unmanned Aerial Vehicles." [42] isimli makalede, insansız hava araçlarından 2 boyutlu nesne tespiti için kullanılan yöntemler üzerine araştırma sonuçları ve karşılaştırmaları yer almaktadır. Makalede şu bilgiler bulunmaktadır:

İHA'larda, çevreden bilgi almak için kullanılan çok sayıda ve çeşitli sensörler olabilir. Makalenin geri kalanında da sıkça bahsedilen ve burada da açıklanan 5 sensör bulunmaktadır. Bunlar: pasif sensör olan RGB kameraları; Dinamik Görme Sensörü (DVS: Dynamic Vision Sensor) kullanan durum-bazlı kameralar, yine pasif sensör olan termal kameralar, 3-boyutlu kameralar ve aktif sensör olan Işık Algılama

ve Ölçeklemedir. (LiDAR: Light Detection and Ranging) Çoğu sistemde tek bir sensör üzerinden alınan veriyle yapılan işlemler yeterli olmadığından birden fazla sensörün birlikte kullanılmasıyla ortaya çıkan çözümler tercih edilmektedir. İHA'larda kullanılan en yaygın sensör olarak RGB kameralar: düşük boyut ve ağırlık, maliyet, enerji tüketimi ve ölçümler sonucu alınan verinin çokluğu gibi özelliklerinden dolayı tercih edilirler.

Izabella Y. Merkulova, Sergey V. Shavetov, Oleg I. Borisov, Vladislav S. Gromov tarafından yazılan: "Object detection and tracking basics: Student Education." [43] isimli yazında, nesne tespiti ve takibinde kullanılan yöntemlerin temelleri ve çalışma prensipleri anlatılmaktadır. Makalede:

Giriş seviyesinde, obje takibinin temeli olarak okunabilecek makale önerileri yer almaktadır. Yeniliklerin ve konu hakkındaki gelişmelerin yer aldığı Faster R-CNN algoritmasıyla alakalı Ren vd.nin (2017) çalışması örnek verilmektedir. Faster R-CNN'in ILSVRC (Large Scale Visual Recognition Challenge) ve COCO (Common Objects in Context) 2015 yarışmalarında birinciliği kazanan algoritma olduğu belirtilmektedir. Diğer kullanılabilir ve önde gelen bir framework olarak Mask R-CNN yapısı örnek verilmektedir. Görüntü işlemenin temel adımları; işlemenin ve sonucun kalitesini etkileyebilecek bazı resim odaklı durumlar; nasıl giderileceği ile ilgili açıklamalar bulunmaktadır.

Gündüz için kamera görüntülerini yeterli olurken, gece çekilen görüntülerde kızılıtesi kamera daha uygundur. Termal görüntüleme canlı tespiti konusunda başarılı olur. Bu gibi kamera sensörleriyle birlikte lazer sensörleriyle de nesne tespiti yapılabilir. Genelde sorumlara çözüm bulmaya çalışırken birden çok sensör bir arada kullanılır.

Resimlerin işlenmesinden önce bilinmesi gereken noktalar vardır: görüntü alan cihazın perspektif izdüşümü (perspective projection), daha da önemli olarak perspektif bozukluğu (perspective distortion) gibi değerler önemlidir. Görüntüdeki bozulmanın boyutunu hesaba katarken ve işlemlerini gerçekleştirirken bunlar işimize yarar. Diğer iki nokta ise: resimlerin ayrıklığı (discreteness) ve kullanılan koordinat sistemi gibi bilgilerdir.

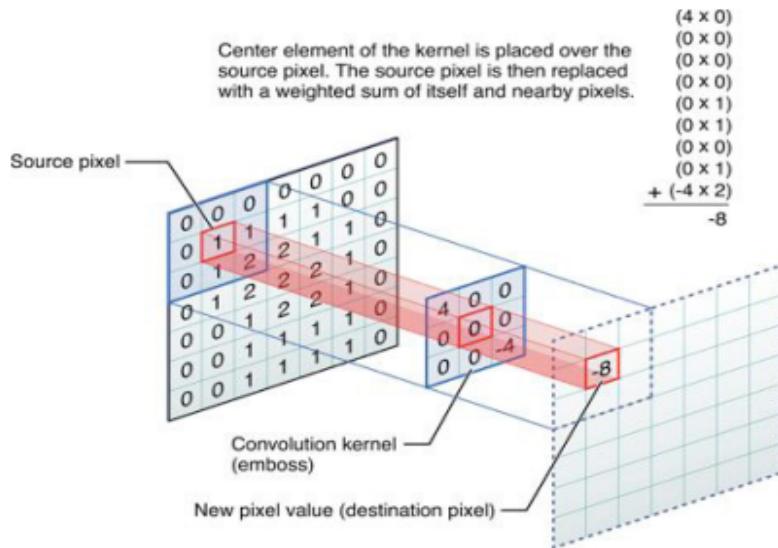
İnsan gözünün resimleri algılaması, sınıflandırması gibi gerçek biyolojik olaylar sinir ağları algoritmalarının temellerini oluşturur. Sinir ağları (neural networks): sinapslarla birbirine bağlanan nöronlardan oluşur. Bir nöron, kendisine gelen bilgiyi alır, basit işlemlerler yapar ve bir sonraki nörona gönderir. Bu yapıda 3 farklı tipte nöron vardır: giriş nöronları (input neurons), çıkış sinirleri (output neurons) ve gizli sinirler (hidden neuorns). Nöronlar arasındaki karakteristikler, yani sinapslar, nöronlar arasındaki ağırlık (weight) değeriyle belirlenir. Bu ağırlıklar giriş bilgisini uygun şekilde değiştirir. Bu ağırlıkların (sinapsların) matrisi ise spesifik olarak oluşturulmuş bir sistemin beyini (çalışma mantığını) bize verir. Sinir ağları sınıflandırma problemlerini gerçekleştirmek için, tahmin görevlerinde; ayrıştırma işlemlerinde (yüz tanıma, arama motorları da fotoğraf aranması) kullanılır. Basit bir ifadeyle sinir ağları: insan beyni tarafından gerçekleşen işlemlere benzer analitik hesaplamaları yapmak ve zor görevleri çözmek için kullanılır.

Hesaba katılması gereken bir diğer resim özelliği ise renktir. Renk, elektromanyetik radyasyonun bir karakteristiği olduğu için cihazın elektromanyetik spektrumu, renk algılamasını anlamak için, işleme görevlerini yapmak için dikkate alınmalıdır.

Geleneksel olarak, resim üzerinde nesne tespiti şu basit adımları izler: ön işleme, tespit edilen nesnelerin sınıflandırılması, sonuçların değerlendirilmesi. Ön işleme işlemi: resim kalitesinin arttırılması ve art arda gelen resimlerin analizinin basitleştirilmesi gibi görevleri içerir. Resimdeki bazı bozukluklar: çok düşük veya yüksek kontrast, gürültülü alanlar, eşit olmayan ışık dağılımı, yanlış renk varlığı, blurlanmış bölgelerin bulunmasıdır. Kontrast farklılığını düzenlemek için parlaklık histogramı kullanılır. Ön işlem aşaması şu konu başlıklarını kavramalıdır: nokta operatörlerinin kullanımı, lineer düzeltme, lineer olmayan gama düzeltmesi ve histogram eşitlemesi. Bu noktadan sonra yapılacak eğitim aşaması yalnızca zorunlu bir görev olarak gerçekleşir.

Resim işleme adımlarının en temellerinden biri resim konvolüsyonudur. Kim vd.nin (2013) [45] yaptığı çalışma, konvolüsyon seçenekleri için kullanılabilir. Aynı sahneye ait birden çok resim varken resim gürültüsü elimine edilebilir. Fakat birden

çok resme sahip olmadığımız zamanlarda kayan pencereler algoritması kullanılarak mekânsal filtreleme uygulanır ve gürültü azaltılır. Bu işlem, bahsedilen resim konvolüsyon işlemidir.



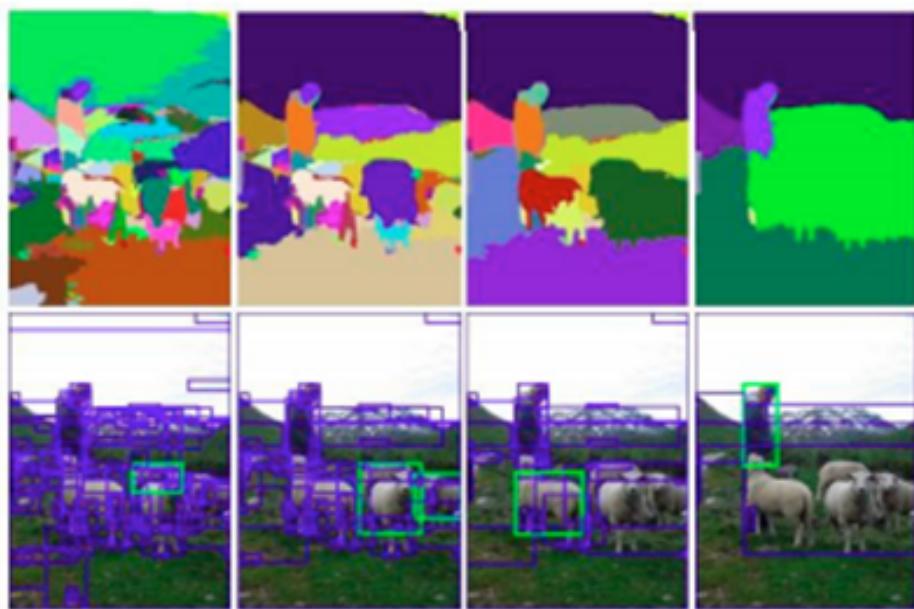
Şekil 1.8. Resim konvolüsyon işlemi (Kim vd. (2013)) [45]

Kenar tespiti algoritmaları ise, resim ön işlemesindeki bir sonraki önemli adımdır. Kenar tespitinin ana amacı resimdeki, önemli olaylardaki ve çevredeki gerçek zamanlı değişikliklerdeki parlaklık farkını bularak kenarların tespit edilmesidir. Nesnelerin kenarlarının tespit edilmesi için temel olarak Jain vd. (1995) [46], daha gelişmiş bir analiz olarak El-Sayed'in (2012) [47] çalışmaları faydalı olabilir.

Resim sınıflandırmasına geldiğimizde, ana sorun olan sahne ve bileşenlerinin tespiti karşımıza çıkar. Bu aşamada, nesne tespiti ve anlamsal segmentasyon işlerinin yalnızca sınıflandırma görevine indirgenebileceğini görmemiz gerekmektedir. Sınıflandırma ikili veya birden çok sınıfı olabilir. Başlangıçtan itibaren bakılacak olursa: teorik olarak kayan pencereler metodu kullanılabilir fakat çok kaynak tüketmektedir. Bunun için doğrulama amacıyla işe yaramayan bilgilerin görmezden geldiği (arka planın çıkarılması gibi) yöntemler kullanılmaya başlanmıştır. 2003'te AdaBoost makine öğrenme algoritması sınıflandırma için önerilmiştir. Bu çalışmanın detayları için Viola vd. (2003) [48] incelenebilir. Bunun ardından, lineer destek vektör makinesi (SVM: support vector machine) sınıflandırıcısı birçok araştırmada

kullanılmıştır. Bu yöntem aynı zamanda: maksimum aralık sınıflandırması (maximum gap classifier) olarak da bilinmektedir. Histogram ve yönelimli gradyanlarla birlikte kullanımı ise zamanla, sınıflandırma sorunu için genel bir araç haline gelmiştir. Fakat bu sayılan yöntemler manuel olarak geliştirilen yöntemlerdir. Sınıflandırma aşamasında önemli bir performans iyileştirmesi sunulması, evrişimli sinir ağlarının (convolutional neural networks, CNN) tanıtılmasıyla gerçekleşmiştir.

CNN kullanımının sınıflandırma problemine entegre edilmesinde çok çeşitli seçenekler mevcut olduğu için derin öğrenme algoritmasının analizi, eğitim işlemi gibi kısımlara büyük zaman ayrılmalıdır. CNN ilk ortaya çıktığında kayan pencereler metodunu kullanarak uygulanmıştır. Resmin tüm düzlemini farklı ölçeklerde analiz ederek gerçekleştirilmiştir. Yüksek performans gözetilmese de yöntemin hızının yavaşlığı dikkat çekmiştir. Sahnedeki büyük boytlardaki hassas bölgeler yüzünden nesnenin tam olarak konumunun bulunması büyük bir zorluk olarak ortaya çıkmıştır. Uijlings vd. (2013) [49] tarafından ortaya atılan R-CNN (region-based CNN), kayan pencereler yerine bölgeler üzerinden tespit yapılmasını önermiştir. Bu yaklaşımın, ilgilenilen nesnelerin geçici bölgeleri (regions of objects of interest, ROI) seçici arama alanları (selective search areas) ile oluşturulmuş olmasıdır.



Şekil 1.9. Farklı ölçeklerde R-CNN ile seçici arama örneği [49]

R-CNN'in hesaplama yükünü azaltmak için birçok çalışma yapılmıştır. Son zamanlardaki performans olarak en iyi yaklaşımlardan biri Chen vd. (2016) [50] tarafından 3 boyutlu varsayımlar yapılarak gerçekleştirilmektedir (Monocular 3D).

Öne sürülen bir diğer yaklaşım çok-ölçekli (multi-scale) CNN, tahmin ve tespit alt ağını tanıtmıştır.

Tablo 1.1. CNN sınıflandırma algoritmalarının karşılaştırmalı örneği [50]

Algorithm	Vehicles detection accuracy (%)	Pedestrians detection accuracy (%)	Consuming time (%)
Multi-scale CNN	76–90	68–84	0.4
Monocular 3D	79–92	63–80	4.2
Faster R-CNN	71–87	61–79	2.0

Nesne takibi kısmına gelindiğinde, yapılan iş daha zorlu hâle gelmektedir. Karmaşık arkaplan, ani nesne hareketleri, görüntüdeki bozukluklar; nesne benzerliği gibi sorunlar takip işlemini zorlaştırmaktadır. Tarihi olarak takipteki yaklaşım genelde Bayes çıkarmı kullanılarak istatistiksel formüllerle gerçekleştirilmiştir. Bu yöntemde, nesnenin önceki noktalarının her zaman bilinmesi ve poz gibi bilgilerle hesapların rekürsif olarak sürekli yapılması gerekmektedir. Uzatılmış Kalman Filtresi ve çok parçalı filtreler bu adımlarda sıkça kullanılmıştır.

Nesne takibi adımlarında yukarıda bahsedilen işlemlerden ise derin öğrenme algoritmalarının kullanılması son 10 yılda giderek kullanımı artan bir yaklaşım olmuştur. Tespit aşamasında kullanılan CNN ayrıca takip için de kullanılır. Ma vd. (2015) [51] takip için CNN kullanımı üzerine okunabilir.

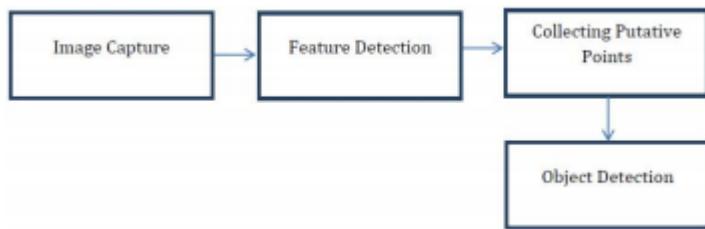
Görüntünün sınıflandırılması için eğitme metotlarının kullanılmasının gerekliliği; bu görevi gerçekleştirirken kullanılabilecek temel 2 alt kategori (binary sınıflandırma ve birden-çok-sınıf sınıflandırması). Sliding window tekniği; artı ve eksileri, bunun yerine kullanılan sınıflandırma metotları: AdaBoost ML algoritmasının kullanımı, SVM (Support vector machine) kullanarak sınıflandırma. En önemli ve bunların kullanımını ortadan kaldırın CNN'nin (Convolutioonal Neural Networks) kullanımı. CNN'nin kullanımı, farklı yaklaşımalar; hesaplama maliyetini düşürmek için ortaya

çıkan farklı CNN algoritmaları. R-CNN, multi-scale CNN algoritmaları. Obje takibinin görevi, temel adımları; kullanılan ilk yöntemler, CNN kullanılarak obje takibi.

Du, Dawei and Qi, Yuankai and Yu, Hongyang and Yang, Yifan and Duan, Kaiwen and Li, Guorong and Zhang, Weigang and Huang, Qingming and Tian, Qi tarafından yazılan “The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking” [52] adlı makalede insansız hava araçlarında nesne tespiti ve takibi üzerine karşılaştırmalar yer almaktadır.

Obje tespiti ve takibi için teknoloji ve algoritmaların belirli veri setleri üzerinde gösterdikleri performansın ve başarılarının değerlendirilmesi. İHA için uygun olan veri setlerinin azlığı üzerine yorumlar; hâlihazırda bulunan veri setlerinin yalnızca spesifik görevler için eğitim modeli yaratmaya elverişli olması üzerine birtakım çıkarımlarda bulunulmaktadır. İHA ile obje tespit ve takibi için olası zorluklar: yüksek yoğunluk (tanımlanması gereken birçok obje), küçük obje(ler), İHA hızı ve kamera rotasyonlarına bağlı kamera hareket hızı, gerçek zaman problemleridir (gerçek senaryolarda hızlı tepki ve doğruluk gerekliliği üzerine zorluklar). Bu problemleri çözmek için Campus ve CARPK gibi elle tutulur sayıda İHA veri setlerinin bulunduğu belirtilmektedir. Derin öğrenme frameworklerinin ayrıldığı 2 temel kategorinin: alan temelli (region-based; Faster-RCNN ve R-FCN) ve alandan bağımsız (region-free; SSD ve RON) veri setleri üzerinde test edilerek karşılaştırılmaları verilmiştir. Grafiksel yorumlarlar yapılmıştır. Obje takibi için kullanılan derin öğrenme metodlarının karşılaştırılması; verilen veri setlerindeki performansları, başarı seviyeleri. Gerçek zamanlı senaryolarda tek obje takibi (single object tracking; SOT) için derin öğrenme algoritmalarının başarı oranının diğer algoritmala oranla yüksekliği; yeni araştırmaların gerçek zaman problemlerine çözüm bulmak için çalıştığı yorumu ve bununla alakalı makale önerileri.

Jalled, Fares, and Ilia Voronkov'un yazdığı “Object detection using image processing.” [53] adlı makalede resim işleme ile nesne tespiti üzerine açıklamalar ve yöntemler yer almaktadır.



Şekil 1.10. Nesne Tespiti için Blok Diyagram Gösterimi [53]

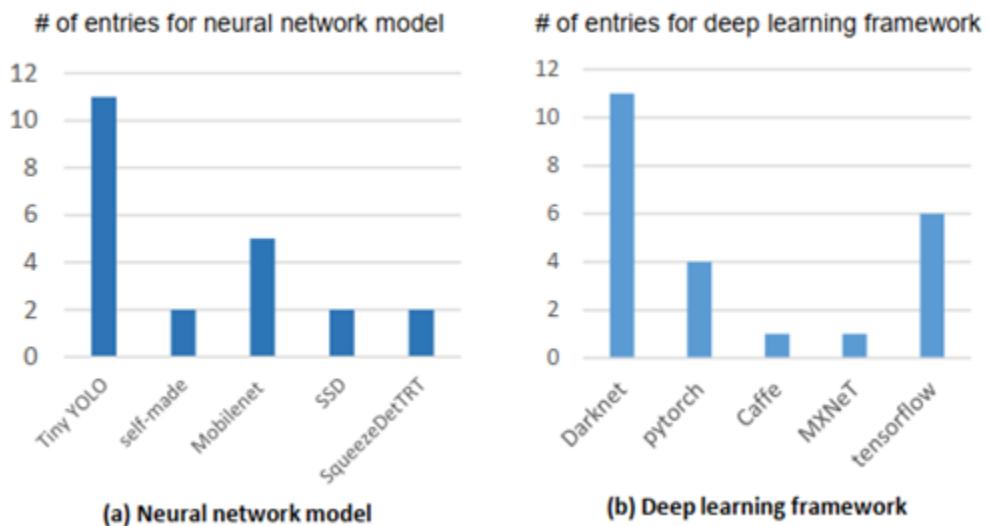
Objenin ve yüz tespiti için Haar Cascade algoritması kullanılarak OpenCV-Python kodu geliştirilmesi üzerine çalışmalar yapılmıştır. Voila-ones algoritması ve çalışma mantığı; Haar Cascade'e göre ana avantajı belirtilemiştir. Objenin tespitinin görüntü üzerinde temel olarak nasıl gerçekleştiğinin anlatımı yapılmıştır. Görüntü-temelli obje tespitin adımları; yüz tespiti, takip edilecek yüz özelliklerinin belirlenmesi ve takibi konusunda açıklamalar yapılmıştır. Nesne tespitinde kullanılan temel algoritmalar (Haar ve Voila), açıklamalar ve başarı, performans olarak karşılaştırımları verilmiştir.

Xu, X., Zhang, X., Yu, B., Hu, X. S., Rowen, C., Hu, J., & Shi, Y. tarafından yazılan “DAC-SDC Low Power Object Detection Challenge for UAV Applications.” [54] makalesinde düşük güç tüketimli sistemlerde nesne tespit üzerine yapılan bir yarışmadan alınan sonuçlar paylaşılmıştır.

İHA'lar için düşük güçte çalışan bir obje tespit çözümü bulunmasına yönelik yapılan yarışma; çıkan sonuçlar ve takımlar tarafından gerçekleştirilen İHA uygulamalarının performans karşılaştırmasını içermektedir. Nesne tespit frameworkleri, derin öğrenme frameworkleri; en çok kullanılan frameworkler ve bazı anahtar farklılıklarını verilmektedir. Frameworklerin karşılaştırılması ve bunlarla ilgili grafikler yer almaktadır.

“LPODC, sonuç performansını: çıktı, güç ve algılama doğruluğunun bir kombinasyonuna göre değerlendirir. Bu nedenle LPODC, İHA uygulamalarının özelliklerini tam olarak dikkate alır: gerçek zamanlı işleme, enerji kısıtlı gömülü platform ve algılama doğruluğu.” [54]

53 katılımcı takımdan 24'ünün GPU kategorisinde dizayn yaptığı ve bunların hepsinin derin öğrenme temelli yaklaşımlarda bulunduğu yorumu yer almaktır. Tiny YOLO tespit framework'ün hafif yapısı dolayısıyla GPU takımlarının çoğunuğunda kullanıldığı belirtilmektedir.



Şekil 1.11. Sinir ağ modeli ve derin öğrenme framework'ü kullanan takımların dağılımı [54]

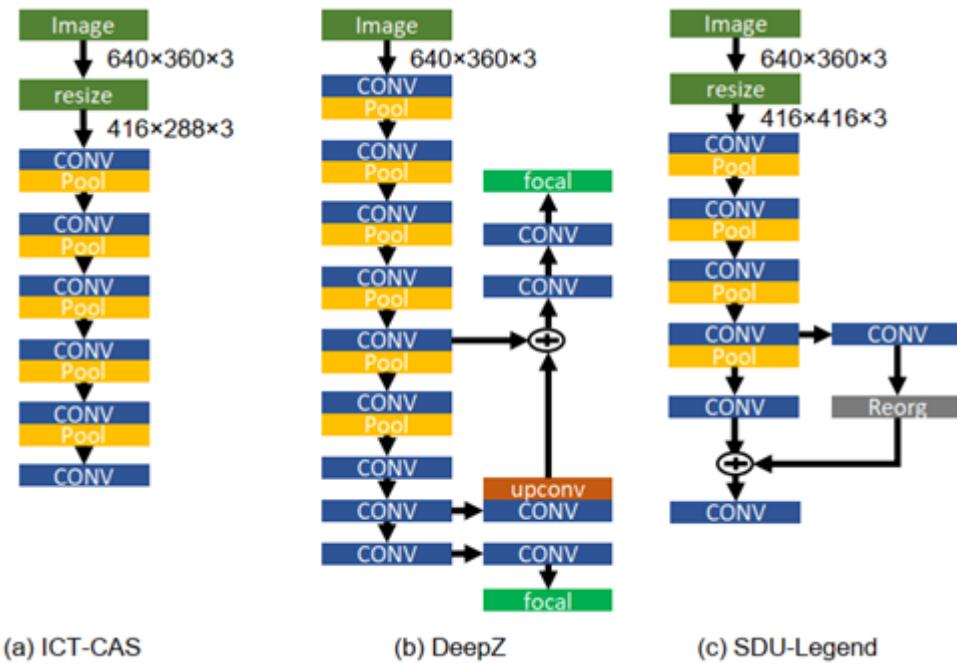
Takımların çoğunun ağ yapısına bazı revizyonlar ekleyerek yüksek başarı oranları elde ettiği görülmüştür. “Tiny YOLO orijinal olarak Darknet'te uygulandığı için, Darknet en popüler derin öğrenme framework'ü diyebiliriz.” En iyi performans gösteren ilk 3 takım YOLO modelini temel tasarım olarak alıp yapısal olarak geliştirmiş ve hesaplamalar optimizasyonlar yapmışlardır. 2 takım veri setinde 640x360 olarak yer alan resim çözünürlüklerini kendi algoritmalarına uyarlayarak değiştirmiştir.

ICT-CAS takımı tiny YOLO ağ yapısını kullanmıştır. Buna ek olarak, doğruluk geliştirmesi Tucker ayırmasını eklemiştir ve zor örneklem madenciliği (hard example mining) geliştirmesi yapmışlardır. İşleme hızı için ise düşük-bitler hesaplaması (low-bits computation) kullanmışlardır. Tucker ayırmasında, farklı ayırmaya parametreleri optimal doğruluk ve sonuç için test edilmiştir. Zor örnekleri elimine edip tekrar eğiterek doğruluk payını artırmışlardır. Çıkarım aşamasını

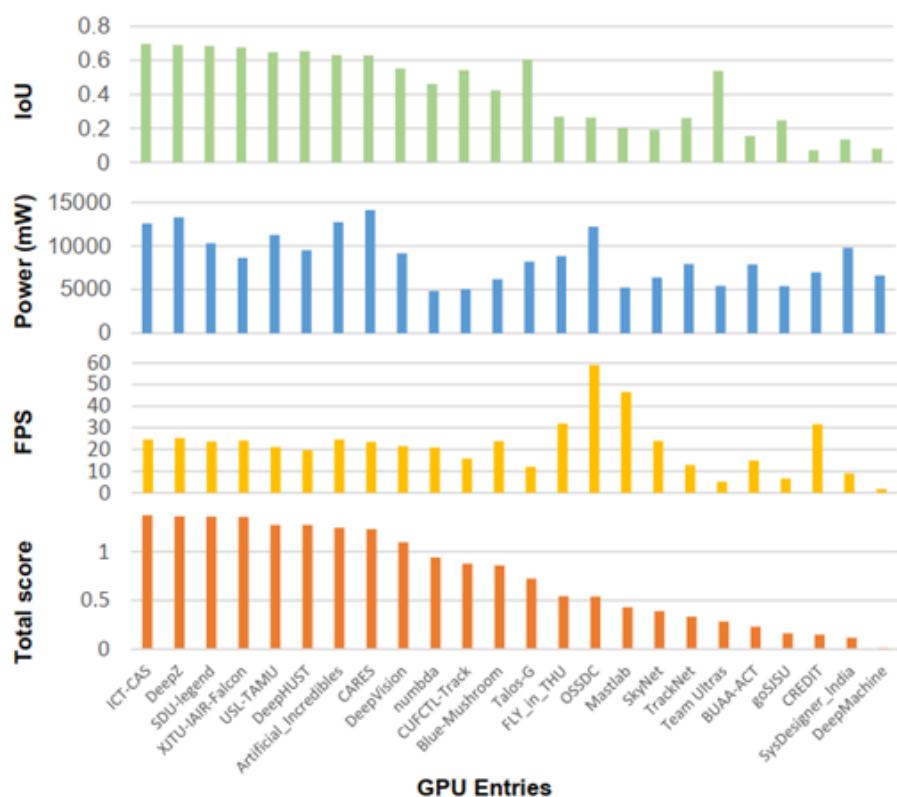
(inferencing stage) hızlandırmak için yarı hassasiyetli float nokta hesaplaması kullanmışlar ve hesaplama karmaşıklığıyla birlikte güç tüketimini azaltmışlardır. TX2 GPU platformunda bu takım ayrıca TensorRT’yi çıkarımı hızlandırmak için çıkışım optimize edici olarak kullanmışlardır.

DeepZ takımı kendi ağ yapılarını uygulamıştır. Küçük objelerin tespiti yeteneğini geliştirmek için Özellik Piramit Ağını (Feature Pyramid Network) güçlü anlamsal özelliklere sahip ince-taneli (fine-grained) özellikler ile kullanılacak hâle getirip birleştirmiştirlerdir. DeepZ takımı, eğitim aşamasındaki tek zemin doğruluk çerçevesi ile aday çerçeveler arasındaki dengesizliği azaltmak için odak kaybı işlevini kullanmıştır. Böylece tikanmaları ve dikkat dağıtıçı unsurları kısmen çözmüştür.

SDU-Legend takımı tespit doğruluğu ve sistem performansı arasında daha iyi bir denge sağlamak için hem sinir ağı hem de mimarisel seviye optimizasyonuna odaklanmıştır. YOLO v2’yi başlangıç tasarım noktası olarak seçmişlerdir. Dayanak noktaları (anchors), koordinatlar, kayıp fonksiyonundaki ölçek, yoğun boyutu ve öğrenme oranı politikası dahil olmak üzere temel eğitim parametrelerini seçmek için tasarım alanı keşfi gerçekleştirmiştirlerdir. Buna ek olarak takım, YOLO v2 ağ mimarisini 32 katmandan 27 katmana indirmiş, altörneklemeye oranını düşürerek küçük hedeflerdeki performansı güçlendirmeye çalışmıştır. Mimarisel düzeyde takım, GPU ve CPU arasındaki iş paylaşımını dengelemek için son 2 katmanı CPU üzerinde çalıştmayı tercih etmiştir. 32-bit float verisi yerine yarım veri tipi (16-bit float) kullanarak hafıza çıktısını geliştirmeye ve hesaplama maliyetini düşürmeye; bunu ise doğruluktan olabilecek minimum kayıpla gerçekleştirilmeye çalışmışlardır.



Şekil 1.12. İlk 3 GPU katılımcılarının sinir ağı yapıları [54]



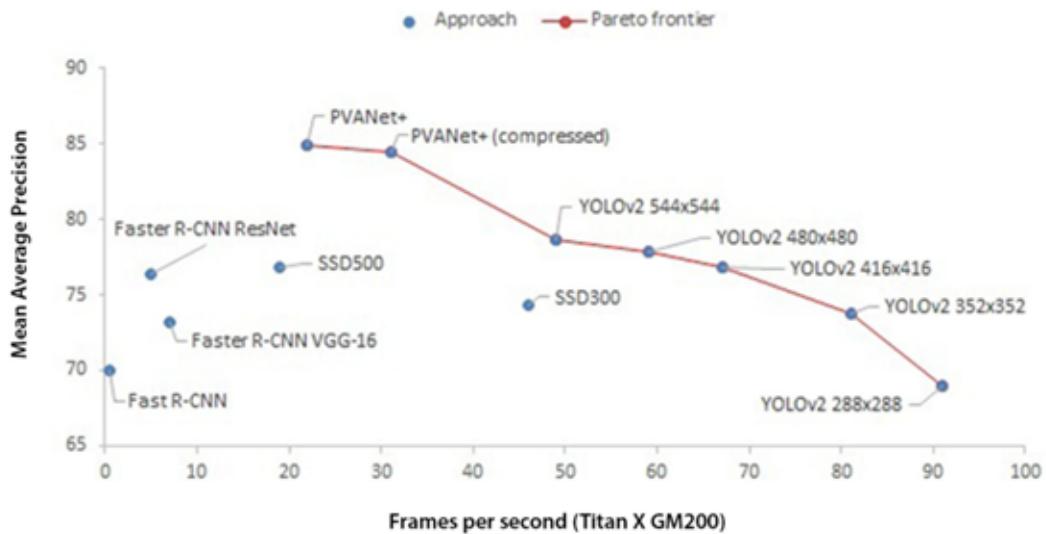
Şekil 1.13. GPU katılımcılarının genel sonuçları [54]

Tijtgat, N., Van Ranst, W., Goedeme, T., Volckaert, B., & De Turck, F. tarafından yazılan “Embedded real-time object detection for a UAV warning system.” [55] Güç kısıtlı mobil uygulamalar için nöron ağları kullanan bir GPU (NVIDIA Jetson TX2) ve YOLOv2 nesne tespit algoritması ile gerçekleştirilen bir İHA uygulaması yer almaktadır.

Bu İHA'da kullanılan metodların, framework ve algoritmaların değerlendirilmesi. Obje tespiti ve resimlerin sınıflandırılması için kullanılan algoritmaların açıklaması, seçilme sebepleri. Evrişimli sinir ağlarının kullanılmaya başlanmasından önce nesne takibinin ilk olarak AdaBoost sınıflandırıcıyla birlikte kullanılan Viola ve Jones tespit sistemi yüz tanımda, elle yazılmış modellerle de gayet iyi sonuçlar alınabildiğini göstermiştir. Diğer tespit sistemlerinden bazıları: Yönlü gradyanların histogramları (Histograms of Oriented Gradients, HoG), İçsel Kanal Özellikleri (Internal Channel Features, ICF), Toplu Kanal Özellikleri (Aggregated Channel Features, ACF) ve Deforme Edilebilir Parçalar Modeli'dir (Deformable Parts Model, DPM) yönlü gradyanların histogramlarıyla yaya ve nesne tespiti konularında gayet başarılı sonuçlar alınabildiğini göstermiştir.

Bu tespit sistemleri klasik makine öğrenme yaklaşımı kullanarak geliştirilmiş özellikler kullanır: karar ağaçları/AdaBoost (ACF, ICF, V&J) veya destek vektör makinesi (SVM) (HoG, DPM). Bu yöntemler genellikle kayan pencere yaklaşımını kullanır. Bu yaklaşım ise: görüntünün tamamı için algılama sonuçlarını alma amacıyla, görüntüdeki tüm konumlarda, farklı ölçeklerde sabit boyutlu bir pencere ortaya çıkarıp bunları değerlendirir. Bu yaklaşım gömülü sistemlere ve İHA'lara entegre edilmiş ve sıkılıkla kullanılmıştır.

Derin öğrenme algoritmalarıyla tespit işlemi ise yüksek doğrulukta sonuçlar vermesine rağmen hızdan ödün vermektektir. Son zamanlardaki gelişmelere kadar, bu algoritmaları gömülü platformlarda gerçek zamanlı olarak çalışırmak; yüksek GPU hesaplama yükü gerektirdiği için göz ardı edilmektedir. Derin öğrenme algoritmalarının hız olarak geleneksel metodlara yetişmesi bunları kullanımının önünü açmaktadır.

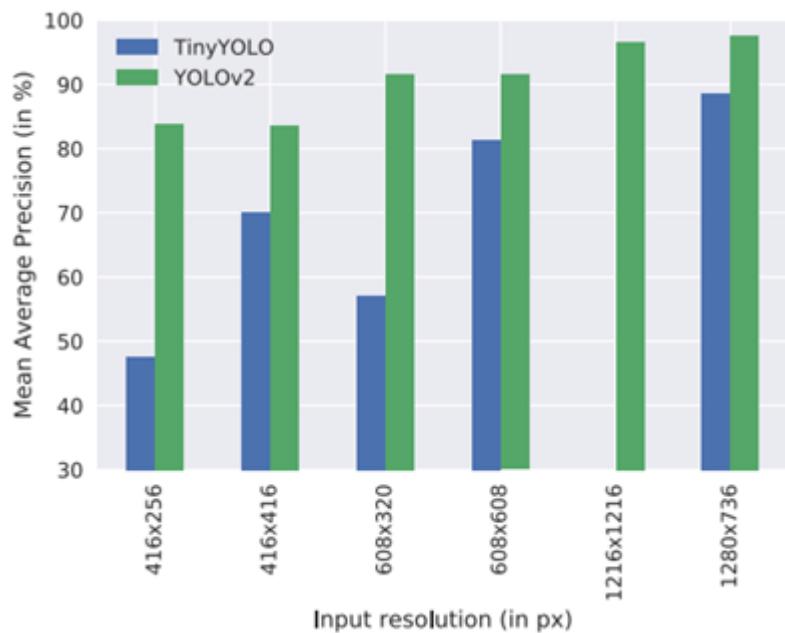


Şekil 1.14. Pascal VOC2007 verileri üzerinde test edilen farklı nesne algılama algoritmalarının karşılaştırması (eğitimler VOC2007 ve VOC2012 verilerinde yapılmıştır.) [55]

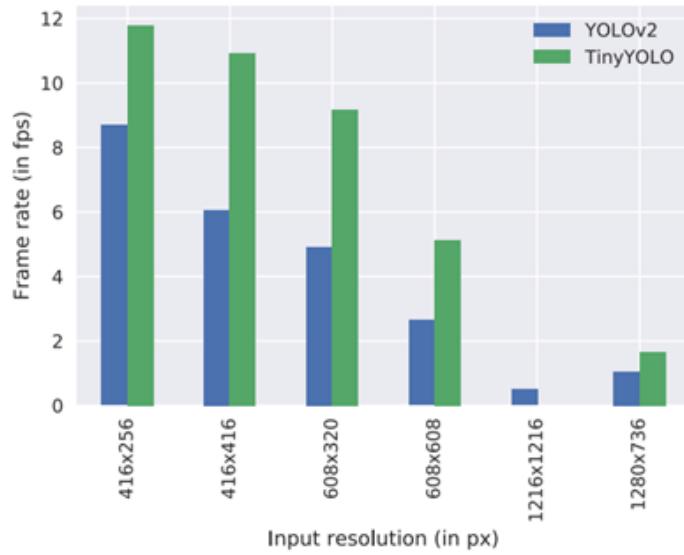
Aynı zamanda önem arz eden diğer parametre çözünürlüktür. Darknet sinir ağı YOLOv2'nin giriş çözünürlüğü varsayılan olarak 416x416'dır. Eğitime veya çıkarıma tabi her görüntü yeniden boyutlandırılır. Bu boyutların artırılması, çıkışım için gereken süreyi artıracak, ancak aynı zamanda daha büyük bir algılama doğruluğu ile sonuçlanacaktır. Yöntemin UAV123 veri setinde verdiği sonuçlara baktığımız zaman ise YOLOv2 algoritmasının diğerlerine göre üstünlüğü belirginleşmiştir. En düşük doğruluğu (% 83.67 çözünürlük: 416x256) en iyi değerle (% 97.67 çözünürlük: 1216x1216) karşılaştırdığımızda, YOLOv2 kare hızının 6 fps'den 0.497 fps'ye düşüğünü görüyoruz. Dolayısıyla doğruluk ve FPS arasında dengenin kurulması gereği görülmüştür. ACF dedektörü, 0,54 fps'lik ortalama bir kare hızına ulaşırken, YOLO dedektörleri sırasıyla 4,53 ve 9,57 fps'lik bir kare hızına ulaşır ve $2,5 \times$ hızlanır.

ACF üzerine el ile yazılmış metodlar, en etkili metodlardan olan YOLOv2 ve TinyYOLO tespit sistemlerine, alınan sonuçlar bakımından oldukça yakındır. Bu algoritmalar, derin öğrenme kısmındaki doğruluk/hız olarak el yazımı metodlara güzel bir alternatif sunmaktadır.

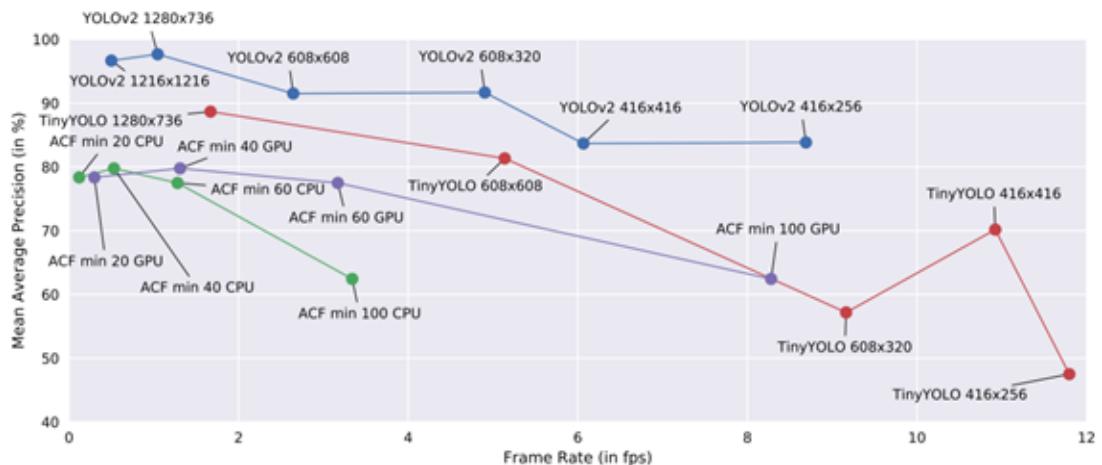
Derin evrişimli sinir ağları nesne tespitindeki çığır açan bir gelişmedir. Gerçek zamanlı kullanılacak algoritmaların çoğu tek aşamaları yaklaşımlardır. İki aşamalı yaklaşımlar hesaplama maliyeti olarak çok yüksek olduğu için şu an tercih edilmesine yaklaşılmamaktadır. Tek aşamalı yaklaşımlardan biri de SSD'dir (Single Shot MultiBox Detection). YOLO'nun (You Only Look Once) bir konfigürasyonu olan ve Redmon & Farhadi (J. Redmon and A. Farhadi. Yolo9000) [56] tarafından önerilen YOLOv2 yaklaşımı tespit hızı ve doğruluğu olarak SSD yaklaşımını geride bırakmıştır. YOLOv2, YOLO'nun daha başarılı ve ardından gelen bir konfigürasyonudur ve aynı yazarlar tarafından gerçekleştirilen açık kaynak C Darknet sinir ağını kullanır. Aynı Darknet sinir ağında çalışan bir diğer model ise TinyYOLO'dur tespit sistemidir. Daha düşük bir tespit doğruluğuna sahiptir; fakat ulaşılabilir kare oranı YOLOv2'nin erişebileceğinden 4 kat daha fazladır. Tek aşama tespit sistemleri, iki aşamaları tespit sistemleriyle kıyaslandığında, tespit doğruluğunda yarışmakta geride kalmaktadırlar. Fakat Ren vd.'nin (2017) [44] öne sürdüğü gibi, alandaki son çalışmalar, bunun değiştileceğini göstermektedir.



Şekil 1.15. Giriş çözünürlüğünün doğrulukla karşılaştırılması [44]



Şekil 1.16. Giriş çözünürlüğünün algılanan kare oranına etkisi [44]



Şekil 1.17. NVIDIA Jetson TX2'de çalışan algoritmaların, farklı çözünürlüklerdeki genel hassasiyet ortalamaları [44]

Yapılacak sistemde olması istenen kare oranı ve tespit doğruluğu gerekliliklerine göre son uygulamada kullanılacak yöntemin seçilmesi gerekmektedir. Sinir ağı temelli YOLOv2 algoritması önde gelen CPU tabanlı algoritmala göre daha yüksek kare oranları ve tespit doğruluk oranı verir.

Payal Mittal, Raman Singh, Akashdeep Sharma tarafından yazılan “Deep learning-based object detection in low-altitude UAV datasets.” [57] adlı makalede alçak irtifa

insansız hava aracı veri setlerinde derin öğrenme temelli nesne tespiti üzerine yöntemler ve açıklamalara yer verilmektedir.

İHA platformlarında nesne takibi konusundaki genel problemler: küçük objeler, tikanma (başka bir obje tarafından görüntünün kesilmesi), objelerdeki büyük ölçekli değişimler/varyasyonlar, sınıf uyuşmazlığı (nasıl çözüleceği hâlâ geçerli bir sorun). Derin öğrenme algoritmalarının obje tespitinde son zamanlarda kullanılmasının örneklerinin verilmesi: Faster RCNN, Mask RCNN, FPN, R-FCN, YOLO, SSD, RetinaNet. Düşük irtifadaki alan resimlerinde geleneksel derin öğrenme algoritmalarının istenen verimi veremediği; bu yüzden son gelişmelerle ortaya çıkan Cascade RCNN, CornerNet, CenterNet, RefineDet gibi algoritmaların kullanılmasının gerekliliği yorumu.

İki aşamalı Nesne Tespiti algoritmaları: tespit edilen objeyi 2 ayrı noktada tanımlayıp ilgi noktaları belirleyip bunları bir ağ üzerinde birbirleriyle sınıflandırarak nesne tespiti ve takibinin yapılması. Faster RCNN, Deformable R-FCN, Mask RCNN, Cascase RCNN. 1 aşamalı Nesne Tespiti algoritmaları: bunların yüksek hız ve 2 aşamalıya göre daha az hesaplama maliyeti; bellekte kaplanan yer gibi artıları dolayısıyla daha tercih edilir bir seçenek olduğu yorumu. SSD, RetinaNet, YOLO V3, RefineDet, CornerNet, M2Det, CenterNet.

Derin öğrenme temelli, daha gelişmiş Nesne Tespiti yaklaşımları: Anchorless Object Detection. CenterNet. Bu yöntemin yeni, büyük bir işlem hacmi gerektirmesi ve yavaş olması gibi bazı dezavantajları. Gerçek hayat senaryosunda görüntü tespitinde problem yaşamamak için veri seti içindeki görüntülerin; gerçek görüntülerde tespit edilmesi istenen nesneye benzerliğinin, boyutunun ve diğer özelliklerinin benzer olması gerekliliği yorumu.

Shreyamsh Kamate, Nuri Yilmazer tarafından yazılan “Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles.” [58] makalesinde nesne tespit ve takip tekniklerinin uygulamaları üzerine aratışmalar ve sonuçlar yer almaktadır.

Sabit (static) kamera görüntülerinde kullanılan algoritmaların temel olarak açıklanması, karşılaştırılması; İHA'lardaki dinamik görüntülerde bu yöntemlerin uygunluğunun değerlendirilmesi.

Hareketli nesnelerin takibi için ‘dengesiz arkaplan’ın (unstable background) bir sorun olarak bulunduğu çıkarımı. Bunu aşmak için Yönlendirilmiş Gradyen Histogramı (Histogram of Oriented Gradients; HOG) tanımlayıcıları, arkaplanı çıkartımı (background subtraction), Gauss karışım modelleri (Gaussian mixture models) kullanılarak uygulanabilir arkaplan çıkarımı (adaptive background subtraction) gibi seçeneklerin olduğu. Bu yöntemler kullanılırken uygulanan aşamaların detaylandırılması.

Sürekli Uyarlanabilir Ortalama Kayma Takibi (Continuously Adaptive Mean Shift Tracking; CAMShift) metodu. Takip için güçlü ve etkili bir algoritma olmasına rağmen İHA'larda kullanmak için uyumlu olmayabileceği çıkarımı. Renk histogramı ve objelerin dokularından yola çıkarak yapılan takibin İHA'larda gerçeklenmesinin zorluğu. Lucas-Kanade Optik Akış Takibi (Lucas-Kanade Optical Flow Tracking) metodu. Nesne üzerinde noktaların belirlenip nesnenin hızı üzerinden gerçekleşen bir işlem olduğu. Tercih edilmediği konusunda yorumlar. Sabit kamera görüntülerini üzerinde dairesel blok eşleştirme (circular block matching) gibi yöntemlerle stabilizasyon gerçekleştirildikten sonra alınan güzel sonuçların İHA'larda da alınabileceği üzerine bir sonuç önermesi.

Mustafa İlker Ekmen ve Ömer Aydoğdu tarafından yazılan “İnsansız Hava Araçları İçin Görüntü İşleme Tabanlı Otonom Iniş” [60] isimli makalede, insansız hava aracının görüntü işleme yardımıyla otomatik inişi ele alınmaktadır. Bu çalışmada havadaki bir insansız hava aracının görüntü işleme metotları kullanılarak, zemin üzerindeki renkli bir alana otonom olarak inmesi sağlanmıştır.

Alan üzerine bir daire konulmuştur. Bunun için renkli dairesel cismin kamera ile koordinatları ölçülerek bu koordinatlara göre iniş yörüngesi belirlenmiştir. Kamera ile çekilen görüntü 512 piksele ayrılmıştır. Çekilen görüntünün tam orta noktası 256x256 pikseller olarak ayarlanmıştır. Kameradan gelen anlık geri kazançlar ile

İHA kendi konumunu sürekli kontrol ederek sabit kalmaya çalışmıştır. Ayrıca çalışmada İHA için eşdeğer lineer bir transfer fonksiyonu elde edilmiştir. İnsansız hava aracının giriş sinyallerinin ve çıkış sinyallerinin değerleri incelenerek System identification ile uygun bir transfer fonksiyonu elde edilmiştir. Bu transfer fonksiyonu denetleyici tasarımda kullanılmıştır.

İHA sistemleri üzerine yerleştirilen dört adet motorun hareket ettirilmesi ile elde edilen itki kuvvetinin yardımıyla hareketini gerçekleştirirler. Dört adet motor ile üç adet konum ve üç adet yönelim durumunu denetlemesi gereken İHA, bu yapısı ile doğrusal olmayan bir modele sahiptir. Motorların hareketi ile durumlar arasında bir bağ olan ve bu durumun bozucu bir etki oluşturduğu insansız hava aracının denetlenmesi zordur.

System Identification kullanılarak PID katsayıların belirlenmesinde en önemli faktör matematiksel modeli tespit edilecek sistemin giriş ve çıkış değerleridir. Bu çalışmada görüntü işleme algoritmaları kullanılarak zeminde tespit edilen bir cismin üzerine dengeli bir şekilde İnsansız hava aracı indirilmeye çalışılmıştır. İnsansız hava araçları üzerine yapılan modelleme çalışmalarından bir tanesi P. Pounds ve arkadaşlarının geliştirdiği, ‘X-4 Flayer’ ismini verdikleri hava aracı tasarımı ve doğrusal modelinin elde edilmesidir. Bu sistemde açısal konumu korumak amacıyla elde edilen doğrusal tek giriş tek çıkışlı modelde PID denetleyici kullanılmıştır.

Matlab Simulink ortamında oluşturulan model de ilk girilen değerler sistemin giriş (Input) değerleri olacaktır. Bu değerler referans olarak kabul edilecektir. Cismin, İHA üzerinde bulunan kameraya göre hangi noktada bulunması gereği bu değerler ile ifade edilmektedir. Görüntü işleme algoritmaları ile kamerasının gösterdiği alan X ve Y ekseni boyunca 512 pikseldir. Cismin kamerasının tam orta noktasında bulunması gereği için yapılan çalışmada referans değeri olarak X ekseninde 256 ve Y ekseninde 256 piksel noktalarında cismin bulunması gerekmektedir. Referans değerlerine göre insansız hava aracının PID katsayıları olmadan iniş pozisyonuna geçerek ne kadar sapma meydana getirdiği uçuş yapılarak incelenmiştir. Aynı

zamanda PID katsayıları olmadan gerçekleştirilen uçuşun X ve Y eksenindeki piksel değerleri yaklaşık 0.1 saniye aralıklar ile kayıt altına alınmıştır.

Matlab Simulink ortamında oluşturulan modele Giriş (Input) ve Çıkış (Output) değerleri girilmiş ve grafiksel sonuçlar alınarak gerçekleştirilen uçuşun referans değerlere göre yaptığı hata incelenmiştir. System Identification toolbox kullanılarak X ve Y eksenlerine göre ayrı ayrı transfer fonksiyonları elde edilmiştir. 9 iterasyon sonucunda sistem girilen değerlere en uygun transfer fonksiyonunu ortaya çıkarmıştır.

Elde edilen transfer fonksiyonu Simulink ortamında oluşturulan matematiksel modele entegre edilerek, Matlab ortamında PID denetleyici için en uygun katsayılar elde edilmiştir. Sistem PID denetleyici olmadan yapılan uçuş ve PID denetleyici ile uçuş durumlarında çalıştırılarak giriş ve çıkış değerleri grafiksel olarak elde edilmiş ve iki durum karşılaştırılmıştır.

İniş algılama sistemi, aracın inmesi için belirlenen yeri kesin olarak belirler. Yerden bir nesneyi tanır ve nesnenin dairesel şekillere sahip olup olmadığını kontrol etmek için geometrik ve kenar algılama algoritmaları kullanır. İniş yerimizi tanımlayan dairesel şekillerin çıkarılması, yüksek kaliteli görüntü yerine kullanılabilir. Bu çalışmada ilk olarak, İHA üzerinde bulunan kamera ile cismin piksel üzerindeki konumları ölçülmektedir. Cismin piksel konumları X ve Y eksenini olarak ayrı ayrı ölçülerek İHA'nın cisme göre ileride, geride, sağda veya solda olduğu tespit edilmektedir. Daha sonra X ve Y eksenleri belirli histogram oluşturularak değerlendirilir. Geometrik hesaba dayalı Histogram, yatay ve dikey yönlerde değerlendirilir; benzer şekilde Öklid mesafesi ile de yapılabilir. Bu dizi, vektörlerin derecesini elde etmek için her piksel için ikili formatta yürütülür. Her histogram için, model özelliği M_j ve görüntü özelliği I arasındaki eşleşme derecesi D_j şu şekilde hesaplanır:

$$Dj = \sum_{i=1}^n \sqrt{I_i M_{ji}}$$

(1.1)

İkili formatın geometrik histogramından dolayı, başka bir hesaplama kenar tespiti olacaktır. Görüntüyü dönüştürmek için Öklid mesafesine özel algoritma kullanılır. Öncelikle 2B görüntüler için algoritma, görüntü çözünürlüğünü X ve Y yönlerinde ayırt eder ve kenar pikselleri tanımlanmalıdır. İkinci olarak, her kenar piksel için tam bir metrik mesafe vardır, bu nedenle metriğin her pikseli, ikili görüntünün eşleşen bir alaniyla ilişkilendirilir ve ikili format görüntüde en yakın pikselleri elde etmek için Öklid mesafesi aracılığıyla bir mesafe değeri atanır. 2D görüntüdeki iki piksel noktası P (x_1, y_1) ve Q (x_2, y_2) için klasik Öklid mesafesi L aşağıda tanımlanan denklem ile hesaplanır:

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

(1.2)

Arka plan mesafesini Öklid mesafe formülasyonu ile çıkarmak için, görüntü çözünürlüğü X ve Y boyutlarında kullanılır, bu nedenle yukarıdaki ifade,

$$L = \sqrt{(x_2 - x_1)^2 * x_{\text{çözüm}}^2 + (y_2 - y_1)^2 * y_{\text{çözüm}}^2}$$

(1.3)

olarak yazılır. Bu formüller, görüntüdeki iki alan için uygulanır. Bu değerler insansız hava aracının referans noktasına göre oluşan hata değerlerinin de tespit edilmesini sağlamıştır.

Batuhan AY ve Ersin Namlı tarafından hazırlanan “Görüntü işleme tabanlı İha ve uydu sistemleri hibrit yapay zeka modeliyle kaçak yapıların tespiti” [61] isimli makalede, imara kapalı olan bölgelerde yasak olmasına rağmen kaçak yapıların var olduğu bilinmektedir. Bu yapıların tespitini insan gücüyle yapmak ve yetkili kişilere bildirmek oldukça zor ve maddi olarak yükümlülük gerektirmektedir. Bu makalede

anlatılan çalışmada, görüntü işleme, makine öğrenme, veri madenciliği ve yapay zeka ile uydu görüntülerinden de faydalananlarak imara kapalı bölgelerdeki kaçak yapılar tespit edilmiştir.

Görüntü işleme kısmı için, görüntü işleme teknikleri ve yapay zeka teknikleri birlikte kullanılarak hibrit model ile İhadan alınan veriler analiz edilmiştir. Analiz sonucunda veriler sınıflandırılmıştır. Elde edilen veriler bir çok görüntü滤resi ile işlenmiş ve elde edilen sonuçlar doğrultusunda 4 filtre üzerinde yoğunlaşmıştır.

Bu filtre sayesinde görüntü üzerindeki ayrıntıları tespit edilebilir. Parmak izi, iris, yüz tanıma vb. görüntü işleme teknikleri gibi bir çok alanda önemli bir filtredir. Gabor滤resi fonksiyonu, Gaussian fonksiyonunu üstel fonksiyon ile çarparak hesaplamıştır. Daugman tarafından Gabor滤resinin iki boyutlu gösterimi yapılmıştır. Gabor滤resi, bilgisayar ile görme ve görüntü işleme konuları için farklı kullanımları vardır.

Renk koreogramı, görüntüdeki renklerin mekânsal korelasyonunu kodlayıp, renk koreogramını hesaplamak için kullanılan filtredir.

Bulanık renk ve doku histogramı. Elde edilen görüntülerden renk özelliklerini ayıklamak için kullanılan filtredir. Renk ve doku bilgilerini histogramda kodlar. Önemli özelliği büyük resimli veri setlerine uygundur. Histogramda renk ve doku bilgilerini bir araya getiren yeni bir düşük özelliğin çıkartılması ile alakalıdır. FCTH boyutu, resim başına 72 bayt ile sınırlıdır. Bu FCTH filtresini, büyük resim veritabanları için kullanılabilir yapar. Önerilen özellikler, deformasyonlar, gürültü, pürüzsüzleştirme vb. bozulma durumlarında bile görüntülerin doğru bir şekilde alınması için uygundur.

Elde edilen görüntünün kenar ve uç bölgelerine odaklanan filtredir. Görüntü indexleme için klasik renk histogramı bir görüntünün şekil bilgisini gözardı eder. Kenar bilgisi olan renk histogramı incelenmiştir. Renk dağılımları iki yönlü kenar, bir yönsüz kenar paterninde pikseller için bulunur, üç kenar ölçüsü, her kenar tipinin renk dağılımı temel alınarak hesaplanır. Bu üç mesafe önlemesini birleştirerek elde

edilen önerilen benzerlik önlemi, yalnızca yönlü kenar türleri kullanıma kıyas ile yanlış eşleme oranını azaltabilir.

Mustafa Zeybek ve İsmail Şanlıoğlu tarafından yazılan “Topografik Yüzey Değişimlerinin Görüntü İşleme Teknikleriyle Belirlenmesi Üzerine Bir Araştırma” [62] isimli çalışmada, topografik yüzeylerde meydana gelen hareketlerin incelenmesi için insansız hava aracı ile birlikte görüntü işleme teknikleri kullanılmıştır. Görüntü işleme için kullanılan teknikler:

IMCORR algoritması, İki görüntü ve bir dizi giriş parametresi ile görüntüler arasında küçük alt görüntü pencerelerini eşleştirmeye çalışan bir yöntemdir. IMCORR, görüntülerini ve boyutlarını, yonga olarak adlandırılan arama çerçevesi boyutunu belirleyen parametrelerin, referans yonga boyutu, grid aralığını ve çıktı dosyasının belirlenmesiyle çalışmaktadır. Ayrıca, arama yonga merkezlerinin önceden ayarlanmış aralık değerleri, tahmini hareketlerin üst limitlerine göre belirlenmelidir. Bununla birlikte IMCORR algoritmasının yer değiştirmeleri bulmaya çalıştığı alan sınırlaması yapılmadan tüm görüntüdeki alanların analiz edilmesi sağlanır. Eğer yer değiştirmelerin yerleri tam olarak bilinirse bu kesme işleminin yapılması analiz performansını artırabilmektedir.

COSICORR algoritması, Optik görüntülerin herhangi bir kaynaktan üretilmiş iki tek bantlı görüntüsü dikkate alınarak çalışmaktadır. Program normalleştirilmiş çapraz kovaryans yönteminin hızlı bir fourier dönüşümü tabanlı versiyonunu kullanmaktadır. Bu tür bir algoritmanın görüntü işlemesinde en yaygın kullanımı, iki görüntüdeki bağlama nokta çiftlerini eşleştirmek için doğru bir şekilde bulunmasından ibarettir. Bununla birlikte, görüntüler öncesinde koordinatlandırıldığı için, algoritma, görüntülerdeki belirlenen özellikler arasında çok az değişiklik olsa dahi hareketli bölümlerin değişim miktarını bulmaktadır.

Adem Kabadayı ve Murat Uysal tarafından hazırlanan “Building detection from high resolution UAV data” [63] isimli makalede, insansız hava aracından elde edilen bilgilerden türetilen ortofoto, sayısal yüzey modeli (SYM) ve sayısal arazi modeli (SAM) verilerinin otomatik bina çıkarımı için bir yaklaşım ortaya koymuştur.

Görüntülerin otomatik olarak eşleşme tekniği ile Sayısal Yüzey Modeli ve ortofoto üretilmiştir. Üretilen ortofoto'nun nesne tabanlı yöntemi ile bölütleme ve sınıflandırma yapılmıştır. Sınıflandırma işleminde çeşitli morfolojik işlemler uygulanarak binaların tespiti yapılmıştır.

Proje için eBee Plus sabit kanatlı insansız hava aracı kullanılmıştır. Resim çekimleri için Sensefly SODA kamera kullanılmıştır. Elde edilen görüntülerin çözünürlüğü 5472x3648 pikseldir. Bir pikselin arazide kapladığı alanı ifade eden konumsal çözünürlük 0.026 metre/piksel.

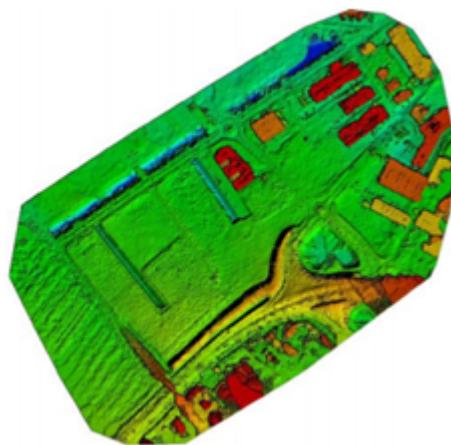
Fotogrametrik değerlendirmede araziye tesis edilen 9 yer kontrol noktası ile yapılmıştır. İnsansız hava aracı ile elde edilen veriler ve yer kontrol noktaları fotogrametik dengelemede kullanılarak oluşan model arazi koordinat sistemine dönüştürülmüştür. Üretilen veriler eCognition yazılımında nesne tabanlı sınıflandırma yöntemi kullanılarak bina çıkarımı yapılmıştır. Sayısal yüzey modeli ve sayısal arazi modeli ile normalize edilmiş sayısal yükseklik modeli (nSYM) elde edilmiştir. Üretilen görüntülere işlemleri yapılmıştır. Sınıflandırılmış görüntü üzerinde morfolojik işlemler uygulanarak binalar tespit edilmiş ve analizi yapılmıştır.

Pix4D programında, insansız hava aracı yardımıyla 106 metre yükseklikten çekilen fotoğraflar görüntü eşlemesi yapılarak işin demetleri ile blok dengelemesi sonucu sayısal arazi modeli üretilmiştir. Piksel boyutu 0.026m x 0.026m olan ortofoto oluşturulmuştur. Ortofoto nun yer örneklem aralığı 2.64 cm/Pikseldir. Üretilen veride karesel ortalama hatalar RMSEX: 1.29 cm, RMSEY: 2.94 cm, RMSEZ: 3.88 cm olarak bulunmuştur.



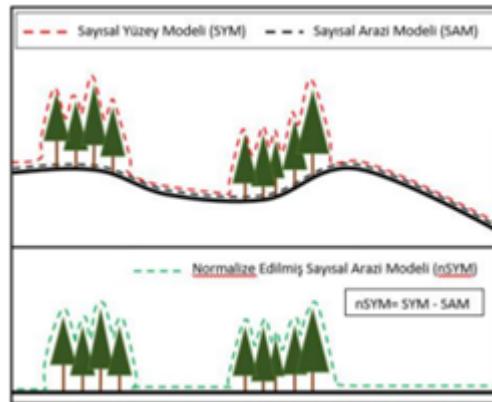
Şekil 1.18. Ortofoto görüntüsü [63]

Sayısal yükseklik modeli (SYM) beşeri ve doğal unsurları kapsamayan, dünyayı temsil eden raster gridlerden oluşan modeldir. Çalışmada Üretilen sayısal yükseklik modeli görüntüsü aşağıdadır. Üretilen sayısal yükseklik modelinin düşey doğruluğu 3.88 cm olarak tespit edilmiştir.



Şekil 1.19. SYM görünümü [63]

Normalize Edilmiş Sayısal Yükseklik Modeli Oluşturma, normalize edilmiş sayısal yüzey modeli (nSYM) üretilen sayısal yükseklik modeli ve sayısal arazi modeli farkından elde edilmektedir. $nSYM = SYM - SAM$ olarak ifade edilir. nSYM verisinde çiplak arazi yüzeyi haricindeki bitki ve insan yapımı objeleri içeren yükseklik verisi bulunmaktadır.



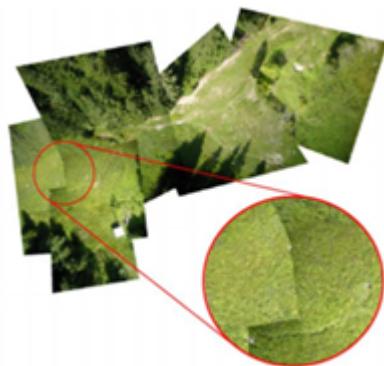
Şekil 1.20. Normalize edilmiş Sayısal Yüzey Modeli [63]

Niethammer, Rothmund, Schwaderer, Zeman ve Joswig tarafından hazırlanan “Open source image-processing tools for low-cost UAV-based landslide investigations” [64] isimli makalede, sivil yapılar için risk teşkil eden heyelan risklerini tespit etmek adına, insansız hava araçları ve uydu tabanlı uzaktan algılama teknikleri kullanılmıştır. Dört rotorlu bir insansız hava aracı sistemi kullanılmıştır. İstenen görüntüleme için uçuş yüksekliği 200 metre olarak tespit edilmiştir. Piksel başına yaklaşık 0.06 m zemin çözünürlüğü elde edilmiştir.

İnsansız hava aracı tabanlı görüntülerin daha iyi analiz edilebilmesi için ortho mosaic yöntemi kullanılmıştır. Fotogrametrik işleme ile görüntüler işlendiğinde daha iyi sonuçlar elde edilebilir. İlk adımda dijital arazi modeli (DTM) oluşturulur ve görüntüler daha sonra DTM yüzeyine yansıtılır. Son adımda, dokunun ortografik re-projection işlemi sonrasında uygulanabilir hale gelir.

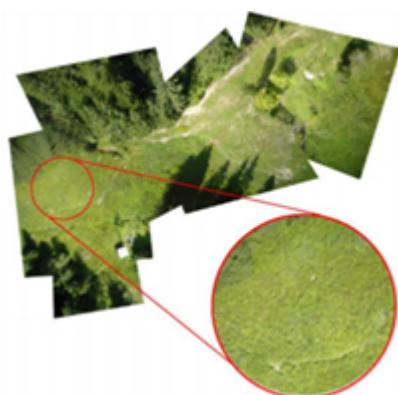
İlk adımda, optik distorsyonun düzeltilmesi gereklidir. Bunun için üçüncü derece polinom yaklaşımı uygulanabilir. Açık kaynak düzeltme aracı olan “Fulla” kullanılabilir.

İkinci adım olarak, her görüntü parametrik olmayan bir düzeltme kullanılarak düzelttilir. Doğrultulmuş resimler basitçe büyük bir mozaikle birleştirilebilir, ancak genellikle görüntü sınırları görünür kalır.



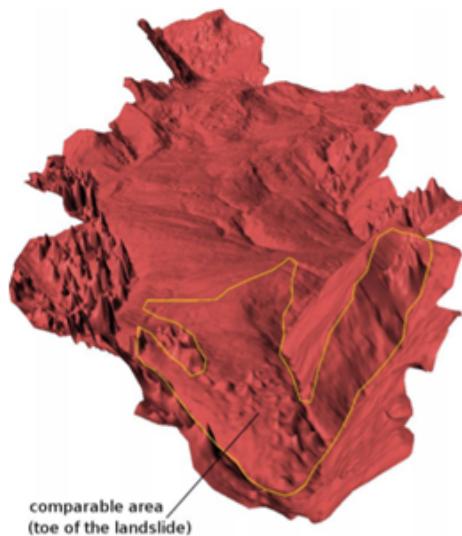
Şekil 1.21. Heumoes heyelan mozaiği görüntü sınırları [64]

Bu yöntem, daha fazla görüntü analizi için çok başarılı değildir. Görüntü harmanlama, yüksek kalite üreten etkili bir yöntem olabilir. Örtüsen görüntülerin radyometrik varyasyonları sıkılıkla insansız hava araçları ile alınan görüntülerde meydana gelir.



Şekil 1.22. Heumoes heyelanının kesintisiz mozaiği [64]

Tek görüntü işleme önemli miktarda zaman gerektirir ve düzlemede rektifiye edilmiş görüntüler kullanıldığından hassasiyet sınırlıdır. Heyelan analizi, özellikle düzensiz arazide yaklaşık görüntü dönüşümü ve düzeltilmiş görüntü içinde yanlış hizalamalar kabul edilmelidir. Ancak, Fotogrametri kullanılırken yanlış hizalamalardan kaçınılabilir. DTM işlemeyi orto-mozaik boru hattına entegre etmek, son orto-mozaikteki hataları önemli ölçüde azaltır ve üç boyutlu ölçümler DTM ile yapılabilir.



Şekil 1.23. Süper Sauze heyelanının DTM'si [64]

Hareket yaklaşımları yer kontrol noktasını dikkate almaz. Son üç boyutlu nokta bulut bilgileri ve koordinat sistemi bilgileri belirsiz kalır. Nokta bulutun hizalanması, belirsiz 3 boyutlu modelin tüm noktalarına benzerlik dönüşümü uygulanarak ayarlanabilir. Dönüşüm için yedi parametre gereklidir. Bir dizi karşılık gelen üç boyutlu noktanın lokasyonundan hesaplanabilir. Şu anda, en iyi hizalama hassasiyetleri, ön düzeltilmiş ortofotolar hesaplanarak elde edilmektedir. 4 boyutlu model içindeki bilgilerden sayısallaştırılmış noktaların hassasiyetleri daha sonra doku tarafından desteklenir. Son işleme adımda, tüm nokta bulutu daha sonra hedefe dönüştürülür.

Önceki çalışmalarda yakın mesafeli fotogrametri, MVS ve karasal lazer tarama (TLS) yaklaşımları karşılaştırılmış ve Süper Sauze heyelanının ucunda analiz edilmiştir.

Kemal Yaman, Ahmet Sarucan, Mehmet Atak ve Nizami Aktürk tarafından hazırlanan “Dinamik çizelgeleme için görüntü işleme ve arıma modelleri yardımıyla veri hazırlama” [65] isimli çalışmada, kameralar yardımıyla alınan gri seviye görüntülerini bilgisayar ortamına aktararak, görüntü segmentasyon işlemleri ile nesneler arka plandan ayrılır. Ayrılan nesneler, kullanılan metodlar ile

belirginleştirilir. Daha sonra netleştirilen görüntülerin gri seviye histogramından, ayrılan nesnelere ait alansal veriler çıkarılmaktadır.

Kameradan alınan görüntü verisinin optik elektrik mekanizma ile elektriksel sinyallere dönüştürülmesidir. Mercekte oluşan görüntü, kameranın sensörlerine odaklanır. Bu ışık elemanları üzerinde ışığın durumuna göre elektrik sinyalleri üretir. Üretilen sinyaller, bilgisayara görüntü verisi aktarılmasında kullanılan analog sinyallerdir. Bu sinyalleri üreten sistemler vakum tüp ve yarı iletken sensör gibi yapılarından oluşmaktadır. Bir diğer kullanılan teknik ise katı hal kameralarıdır. Kameralar yük bağlamalı düzen veya charge-coupled device (CCD) teknolojisi ile çalışan kamera türleridir.

Kameralardan elde edilen görüntüler, görüntü yakalama programı ile 10-15 saniye aralıklarla yakalanmış 8 bitlik gri düzey Windows BMP formatında, kameraların bulunduğu istasyona ait görüntü kütüphanesi oluşturmak üzere kayıt altına alınmıştır.

İkinci aşamada, incelenenek istasyonlara ait boş ve dolu istasyon görüntüleri bilgisayara okutulmuştur.

Üçüncü aşamada, dolu istasyon görüntülerinden ,boş istasyon görüntüsü, görüntü çıkarma teknikleri kullanılarak çıkartılmıştır. İşlemlerde kullanılan görüntülerin çözünürlükleri 352x288 pikseldir. Yani görüntü matrisleri, 352 sütün ve 288 satırdan oluşan, elemanları 0-255 arasında değişen görüntü matrisleridir.

Dördüncü aşamada ise, üçüncü adımın sonucunda elde edilen matris işlenemeyecek kadar bozuk olduğu için, kendisi ile çarpılarak bozukluk giderilip işlenecek kıvama getirilmiştir. Görüntü üzerindeki kirliliklerden arınması ve netliğin daha fazla artırılması için eşikleme, filtreleme ve histogram eşitleme teknikleri uygulanmıştır.

Son aşamada, elde edilen görüntüde nesneler arka plandan gri renk tonu değerlerinde ayrılmıştır. Son görüntünün histogramı çıkarıldığında, yalnız nesnelere ait çevresel bilgiler edinilir. Gri düzey histogramda yatay eksen 0-255 arası gri renk tonu

skalasıdır. Düşey eksen hangi renk tonundan kaçar defa tekrarlandığını, frekansı verir. Görüntü içindeki nesnenin kapalı ve ayrik bölgelerinin belirginleştirilmesinde kullanılan bir metottur. Piksellere ayrılmış görüntünün, ikili yapıdaki görüntüye kadar düzenlenmesini kapsar.

Bu çalışmada, görüntü üzerindeki yapılacak işlemler için Sigma-Scan pro görüntü analizi yazılımı kullanılmıştır. Bu yazılım ile, görüntü düzeltme, filtreleme işlemleri, elde edilen görüntüyü belirginleştirme, iz ve gölge bulma ve görüntünün ton ayarları yapılabilir. Bu yazılım, daha kompleks görüntü işleme işlemleri için Matlab 5.0 paket programını da kullanmaktadır.



Şekil 1.24. Boş ve incelenecek istasyon görüntüleri [65]

Bahadır Şin ve İzzet Kadioğlu tarafından yazılan “İnsansız Hava Aracı (İHA) ve Görüntü İşleme Teknikleri Kullanılarak Yabancı Ot Tespitinin Yapılması” [59] adlı çalışmasında, yapılan çalışmalara göre, bitkilerde yabancı otlardan kaynaklı %9,5 lik bir ürün kaybı olduğu tespit edilmiştir. Bunun üzerine insansız hava araçları üzerinden görüntü işleme yardımıyla, arazi koşullarında yabancı ot tespitlerinin yapılması amaçlanmıştır.

İnsansız hava araçlarından ve kameralardan alınacak olan veriler, farklı işlemlerden geçirilerek, mevcut olan floranın harmasını çıkarılabileceği gibi, olusabilecek ürün kayıplarının da hesaplanmasını sağlamaktadır. Yabancı otların tespiti için kullanılan kameralar genel olarak kızıl ötesi (NIR) veya NDVI (Normalized Difference Vegetation Index) özellikte olmasına dikkat edilmelidir. Bu tespit kızılötesi kameraları ile bitkilerin yakın kızılötesi bandının (NIR), kırmızı (R) banda oranının

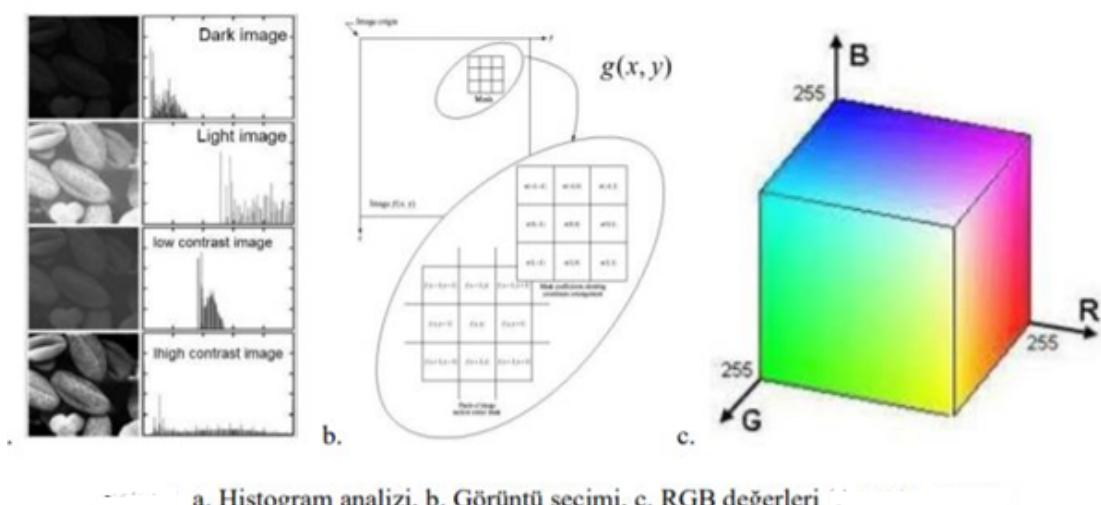
belirlenmesi ile yapılmaktadır. Kayıt altına alınan veriler daha sonra işlemlere tabi tutularak yabancı otun türü, varlığı, yoğunluğu ve ekonomik zarar seviyesi belirlenebilmektedir.

İnsansız hava araçları yardımıyla alınan görüntülerin işlenmesi ile ve GPS verilerinin yardımcılarıyla, tespit edilen yabancı otların konum bilgilerine de ulaşılabilmektedir. Mevcut veri kütüphanelerinin oluşturulmasının ardından seçilmiş olan resimlerin işlenmesine sıra gelmektedir. Bunlar için 3 farklı işleme tekniği kullanılmaktadır.

Bunlardan ilki beyaz-gri dengesi olarak geçen histogram işlemidir. Buradaki amaç dijital görüntüdeki beyaz-gri dengesinden yararlanılarak görüntüdeki şeklin belirlenmesidir.

İkinci yöntemde görüntü filtrelemesi kullanılmaktadır. Görüntü filtreleme işinde elde bulunan dijital görüntünün gridlere bölünmek suretiyle istenilen kısmının kullanılması işlenmesi amaçlanmaktadır.

Üçüncü olarak ise temel renk model baz alınmak suretiyle R-G-B (red-green-blue) değerlerinden yararlanarak dijital görüntünün tanımlanmasının yapılmasıdır.



Şekil 1.25. Histogram analizi, görüntü seçimi, RGB değerleri [59]

1.2. Yöntem

Bitirme projesi kapsamında sabit kanatlı insansız hava aracı platformu kullanılacaktır.

1.2.1. Görüntü iletim sistemi

Görüntü araçtan yer istasyonuna gönderilip orada işlenmesi planlanmaktadır. Kullanılacak yönteme dair uygulama şu şekildedir:



Şekil 1.26. İHA sistemi üzerinde kullanılan yer kontrol istasyonu

Bu sistem OpenHD olarak geçmektedir. Açık kaynaklı bir yöntem olan OpenHD, WiFi adaptörlerini kullanır, ancak bunları düşük gecikmeli veya çok uzun mesafeli video aktarımı için uygun olmayan standart WiFi modunda çalışırmaz. Bunun yerine, WiFi bağıdaştırıcısını analog video aktarım donanımına çok benzer şekilde basit bir yayına benzer şekilde yapılandırır. Bu şekilde yapılacak görüntü iletimi en az maliyet ile tamamlanması planlanmaktadır. İnsansız hava aracında kamera olarak Raspberry Pi HQ Cam kullanılması düşünülmektedir. Bunun sebebi ise, yine kamera maliyetinin ucuz kapatılmasıdır ancak ekstra olarak kullanılacak iletişim yönteminde Raspberry kullanıldığından dolayı daha verimli olacaktır.

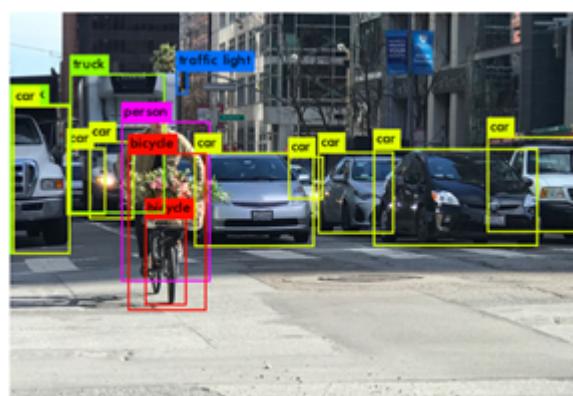
1.2.2. Görüntü işleme yöntemi

YOLOv2 [68], gerçek zamanlı işleme için hedeflenen bir nesne algılama sistemidir. YOLO çeşitleri arasında web sitesi tarafından sağlanan doğruluk ve hız karşılaştırması şekildeki gibidir:

Tablo 1.2. Gerçek zamanlı nesne tespitinde kullanılan algoritmalarının karşılaştırılması

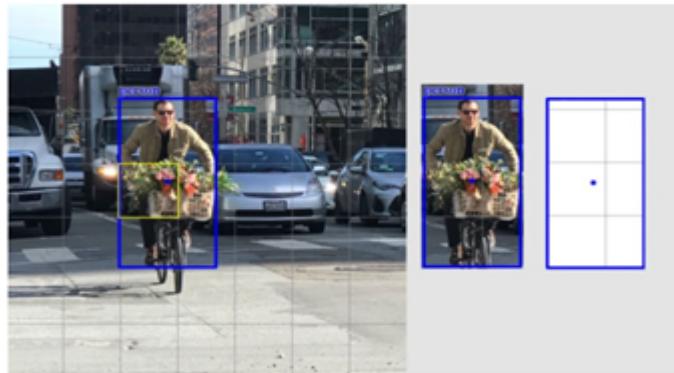
Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	-	-	7.07 Bn	200
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-416	COCO trainval	test-dev	57.9	140.69 Bn	20

YOLO tarafından tespit edilen nesnelere bir örnek aşağıdaki gibidir:



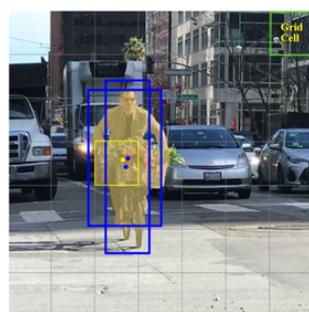
Şekil 1.27. YOLO nesne tespiti örneği

Orijinal fotoğrafı kirparak giriş görüntüsünü bir $S \times S$ ızgarasına böler. Dolayısıyla her bir hücre, bir nesneye denk gelmektedir.



Şekil 1.28. YOLO modelindeki ızgara yapısı

Her ızgara hücresi, sabit sayıda sınır kutusu öngörmektedir. Örnekte, sarı ızgara hücresi, kişinin nerede olduğunu bulmak için iki sınır kutusu tahmini (mavi kutular) yapmaktadır.

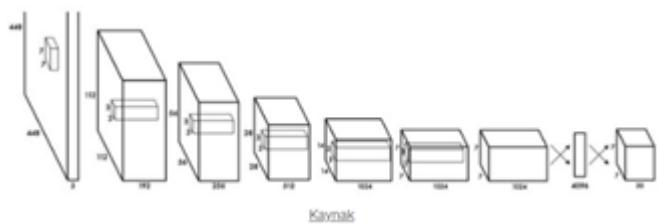


Şekil 1.29. YOLO modelinde sınırlayıcı kutu

Ancak, tek nesne kuralı algılanan nesnelerin ne kadar yakın olabileceğini sınırlamaktadır. Bunun için, YOLO'nun nesnelerin ne kadar yakın olabileceği konusunda bazı sınırlamalar yapar.

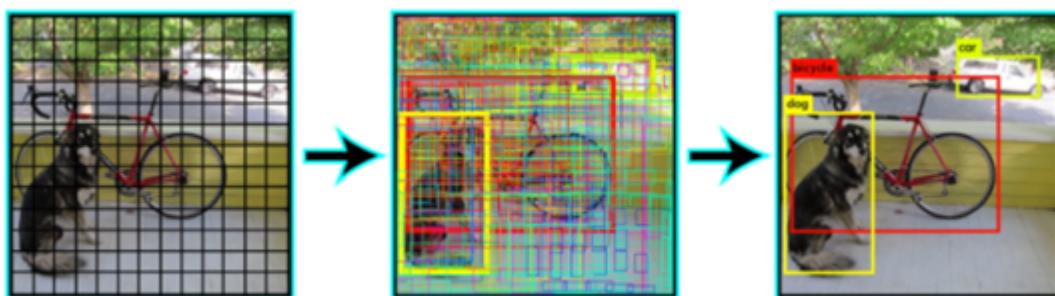
Her hücre için, B sınır kutularını tahmin etmektedir ve her kutunun bir kutu güven puanı vardır. B kutularının sayısına bakılmaksızın yalnızca bir nesneyi algılar. C koşullu sınıf olasılıklarını tahmin eder (nesne sınıfının benzerliği için sınıf başına bir tane). YOLO PASCAL VOC'yi değerlendirmek için, 7×7 ızgara ($S \times S$), 2 sınır kutusu (B) ve 20 sınıf (C) kullanır.

Her bir sınır kutusu 5 öğe içermektedir: (x, y, w, h) ve bir kutu güven puanı. Güven puanı, kutunun bir nesne içerme olasılığını ve sınır kutusunun ne kadar doğru olduğunu yansıtmaktadır. Sınırlayıcı kutu genişliğini w ve yüksekliğini h görüntü genişliği ve yüksekliğine göre normalleştirilmektedir. x ve y karşılık gelen hücreye uzaklıktır. Dolayısıyla, x, y, w ve h , 0 ile 1 arasındadır. Her hücre 20 koşullu olasılık sınıfına sahiptir. Koşullu sınıf olasılık tespit edilen nesnenin belirli bir sınıf'a ait olma olasılığıdır. Dolayısıyla, YOLO'nun tahmini $(S, S, B \times 5 + C) = (7, 7, 2 \times 5 + 20) = (7, 7, 30)$ şeklindeki bir tensörde ifade edilmektedir.



Şekil 1.30. YOLO modelinin örnek yapısı

Bu tensörü oluşturmak için bir CNN ağı oluşturulmaktadır. Uzamsal boyutu her konumda 1024 çıktı kanalından 7×7 'ye düşürmek için bir CNN ağı kullanılır. $7 \times 7 \times 2$ sınır kutusu tahminleri yapmak için iki katman kullanarak doğrusal bir regresyon gerçekleştirilir. Son tahminde bulunmak için, güven puanı 0.25'ten yüksek olanları tutulmaktadır.



Şekil 1.31. YOLO modelinde tespit edilen nesnelerin etiketlenmesi

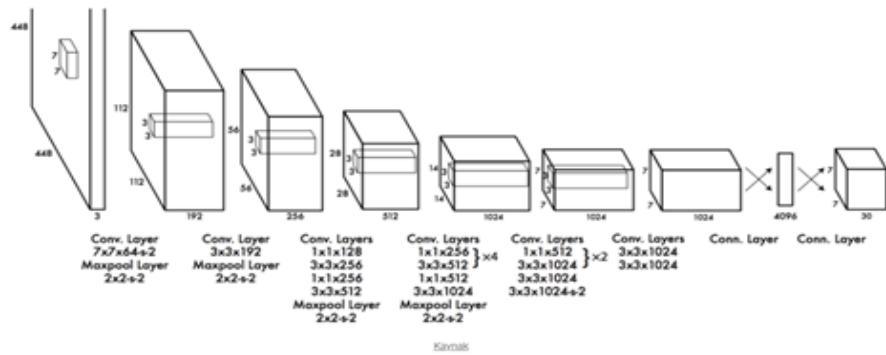
Sınıf güven puanı her tahmin kutu gibi hesaplanır:

$$\text{class confidence score} = \text{box confidence score} \times \text{conditional class probability}$$

Hem sınıflandırma hem de yerelleştirme (bir nesnenin bulunduğu yer) üzerindeki güveni ölçer . Bu puanlama ve olasılık terimlerini kolayca karıştırabiliriz. Gelecekteki referansınız için matematiksel tanımlar.

$$\text{box confidence score} = P_T(\text{object}) . \text{IoU} = \text{box confidence score} \times \text{conditional class probability}$$

1.2.2.1. Ağ tasarımı



Şekil 1.32. YOLO modelinin evrişimli sinir ağındaki yapısı

YOLOv2 [68], 24 evrişimli katmana ve ardından 2 tam bağlı katmana (FC) sahiptir. Bazı evrişim katmanları, özellik haritalarının derinliğini azaltmak için alternatif olarak 1×1 indirgeme katmanları kullanır [69]. Son evrişim katmanı için, şekilli (7, 7, 1024) bir tensör çıkarır. Tensör daha sonra düzleştirilir. Doğrusal regresyon biçimini olarak tamamen bağlantılı 2 katmanı kullanarak, $7 \times 7 \times 30$ parametrelerini çıkarır ve ardından (7, 7, 30) şeklinde yeniden şekillendirir, yani konum başına 2 sınır kutusu tahmini.

YOLO, birden çok sınırlayıcı kutu öngörür. Kaykı true-positive olarak hesaplamak için, nesneden yalnızca birinin sorumlu olmasını istiyoruz. Bu amaçla, en yüksek IoU'ya sahip olanı seçiyoruz. Bu strateji, sınırlayıcı kutu tahminleri arasında uzmanlaşmaya yol açar. Her tahmin, belirli boyutları ve en boy oranlarını tahmin etmede daha iyi hale gelir. YOLO, kaykı hesaplamak için tahminler ve temel gerçek arasındaki toplamın kare hatayı kullanır. Kayıp işlevi şunlardan oluşur: sınıflandırma kaykı, yerelleştirme kaykı (tahmin edilen sınır kutusu ile temel gerçek arasındaki

hatalar), güven kaybı. Sınıflandırma kaybı, her bir sınıf için koşullu olasılıklarının karesi alınmış hatasıdır:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

where

$\mathbb{1}_i^{\text{obj}} = 1$ if an object appears in cell i , otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i .

Yerelleştirme kaybı, tahmin edilen sınır kutusu konumları ve boyutlarındaki hataları ölçmektedir. Sadece nesneyi tespit etmekten sorumlu kutuyu saymamız gereklidir.

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \end{aligned}$$

where

$\mathbb{1}_{ij}^{\text{obj}} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Büyük kutulardaki ve küçük kutulardaki mutlak hataları eşit olarak ağırlıklandırmak istenmemektedir. Dolayısıyla büyük kutudaki 2 piksellik bir hata, küçük bir kutu için aynıdır. Bunu kısmen ele almak için YOLO, genişlik ve yükseklik yerine sınırlayıcı kutu genişliğinin ve yüksekliğinin karekökünü tahmin eder. Ek olarak, sınır kutusu doğruluğuna daha fazla vurgu yapmak için kaybı λ_{coord} ile çarpmamız gereklidir.

Güven kaybı, kutuda bir nesne tespit edilirse, güven kaybı aşağıdaki gibi hesaplanmaktadır:

$$\sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

where

\hat{C}_i is the box confidence score of the box j in cell i .

$\mathbb{1}_{ij}^{\text{obj}} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

Kutuda bir nesne algılanmazsa, güven kaybı aşağıdaki şekilde hesaplanmaktadır:

$$\lambda_{\text{noobj}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

where

$\mathbb{1}_{ij}^{\text{noobj}}$ is the complement of $\mathbb{1}_{ij}^{\text{obj}}$.

\hat{C}_i is the box confidence score of the box j in cell i .

λ_{noobj} weights down the loss when detecting background.

Kutuların çoğu herhangi bir nesne içermez. Bu bir sınıf dengesizliği sorununa neden olur, yani modeli nesneleri tespit etmekten daha sık; arka planı tespit etmesi için eğitmiş oluruz. Bunu düzeltmek için, bu kaybı bir λ_{noobj} faktörü (varsayılan olarak 5) ile ağırlıklandırıyoruz.

Nihai kayıp, yerelleştirme, güven ve sınıflandırma kayipları bir araya getirilir.

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{\text{coord}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ & + \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^S \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^S \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

Nesne tespiti sonrasında nesne izleme işlemi gerçekleştirilecektir. Ve görüntü işleme sisteminden alınan sonuç uçağa iletilecektir. Uçuş kartı iletişim protokolü olarak Mavlink protokolünü kullanmaktadır. MAVLink, insansız hava araçları ile iletişim kurmak için çok hafif bir mesajlaşma protokolüdür. Proje kapsamında Python'da yazılmış olan Dronekit kütüphanesi bu amaçla kullanılacaktır.

1.2.2. çıkışım

YOLO, aynı nesne için yinelenen algılamalar yapabilir. Bunu düzeltmek için, yinelemeleri daha düşük güvenle kaldırmak için maksimal olmayan bastırma uygular. Maksimal olmayan bastırma mAP'de %2-3 ekler.

İşte olası maksimal olmayan bastırma uygulamalarından biri:

1. Tahminleri güven puanlarına göre sıralayın.
2. En yüksek puanlardan başlayın, mevcut tahminle aynı sınıfı ve $\text{IoU} > 0,5$ 'e sahip önceki tahminler bulursak, mevcut tahminleri göz ardı edin.
3. Tüm tahminler kontrol edilene kadar 2. adımı tekrarlayın.

1.3. Uygulama

Kullanılacak veri seti seçiminde iki adet veri seti üzerinde durulmaktadır: VisDrone [66] ve UAVDT [67].

VisDrone veri seti, AISKEYE ekibi tarafından toplanmıştır. 265.228 frame ve 10.209 statik görüntüsünden oluşan 400 video bulundurur. Bu videolar Çin'deki 14 farklı şehirden çekilmiştir. Ve videolarda farklı konumlar, nesneler ve yoğunluklarda bulunmaktadır. Farklı çeşitli drone platformları (yani, farklı modellere sahip dronlar) kullanılarak, farklı senaryolarda ve çeşitli hava ve aydınlatma koşullarında toplanmıştır. Bu da çeşitlilik sunmaktadır. Bu çerçeveler, 2,6 milyondan fazla sınırlayıcı kutu veya yayalar, arabalar, bisikletler ve üç tekerlekli bisikletler gibi sık ilgi alanlarına giren hedef noktaları ile manuel olarak açıklanmaktadır./

UADVDT veri seti, yeni seviye zorlukları içeren karmaşık senaryolara odaklanan yeni bir İHA veri setidir. Toplam 10 saatlik ham videolardan seçilmiş olan, yaklaşık 80.000 temsili çerçeveye tam olarak sınırlayıcı kutularla ve ayrıca üç temel bilgisayarla görme görevi için 14 tür özniteliğe (örn. hava durumu, çarpma rakımı, kamera görüntüsü, araç kategorisi ve kapanma) açıklama eklenmiştir: nesne algılama, tek nesne izleme ve çoklu nesne izleme.

li

2. METODOLOJİ

Nesne algılama, bir görüntü veya videodaki nesnelerin tanımlanmasına ve konumlandırılmasına izin veren bir bilgisayarlı görü teknigidir. Bu tür bir tanımlama ve yerelleştirme ile birlikte nesne algılama, bir sahnedeki nesneleri saymak ve doğru bir şekilde etiketlerken kesin konumlarını belirlemek ve izlemek için kullanılabilir.

2.1 Nesne Tespitı

Nesne algılama, algılanan nesnelerin etrafına söz konusu nesnelerin belirli bir sahnede nerede olduğunu (veya nasıl hareket ettiklerini) bulmamızı sağlayan sınırlayıcı kutular çizer.

Nesne algılama genellikle görüntü tanıma ile karıştırılır, bu nedenle devam etmeden önce aralarındaki ayırmaları netleştirmemiz önemlidir. Görüntü tanıma, görüntüye etiket atar. Örneğin köpek içeren resme "köpek" etiketi verilir. Nesne tespiti ise her köpeğin etrafına bir kutu çizer ve kutuyu "köpek" olarak etiketler. Model, her bir nesnenin nerede olduğunu ve hangi etiketin uygulanması gerektiğini tahmin eder. Bu şekilde nesne algılama, bir görüntü hakkında görüntü tanımadan daha fazla bilgi sağlar.

2.2 Nesne Tespitı Türleri

Genel olarak nesne algılama, makine öğrenimine dayalı yaklaşımlara ve derin öğrenmeye dayalı yaklaşımlara bölünebilir. Daha geleneksel ML tabanlı yaklaşımlarda, nesneye ait olabilecek piksel gruplarını tanımlamak için görüntünün renk histogramı veya kenarları gibi çeşitli özelliklerine bakmak için bilgisayarla görme teknikleri kullanılır. Bu özellikler daha sonra nesnenin konumunu etiketile birlikte tahmin eden bir regresyon modelini besler.

Öte yandan derin öğrenmeye dayalı yaklaşımalar, özelliklerin ayrı ayrı tanımlanması ve çıkarılması gerekmeyen uçtan uca, denetimsiz nesne algılama gerçekleştirmek için evrişimli sinir ağlarını (CNN'ler) kullanır.

Nesne algılama, görüntü veya videodaki sahneleri anlamamıza ve analiz etmemize yardımcı olması açısından görüntü tanıma ve görüntü bölümleme gibi diğer benzer bilgisayarla görme teknikleriyle önemli bir şekilde bağlantılıdır.

Ancak önemli farklılıklar da var. Görüntü tanıma, yalnızca tanımlanmış bir nesne için bir sınıf etiketi çıkarır ve görüntü bölümleme, bir sahnenin öğelerinin piksel düzeyinde anlaşılmasını sağlar. Nesne algılamayı diğer görevlerden ayıran şey, görüntü veya video içindeki nesneleri konumlandırma konusundaki benzersiz yeteneğidir. Bu özellik, daha sonra bu nesneleri saymamızı ve ardından izlememizi sağlar.

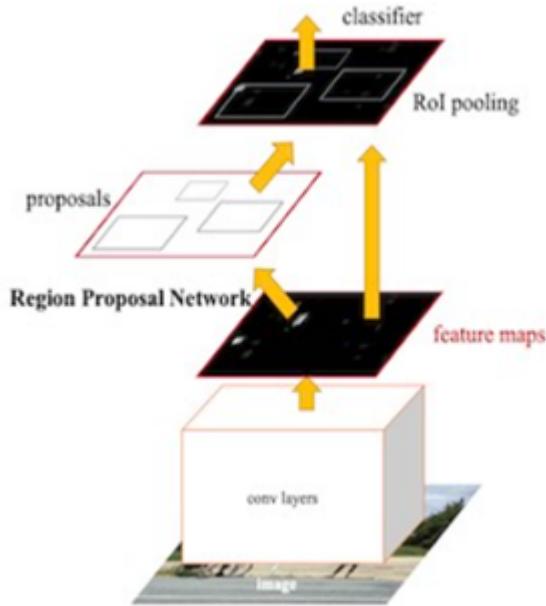
2.2.1. Nesne tespitinin çalışması

Derin öğrenme tabanlı nesne algılama modellerinin tipik olarak iki bölümünden oluşur. Bir kodlayıcı, bir görüntüyü girdi olarak alır ve nesneleri konumlandırmak ve etiketlemek için kullanılan istatistiksel özellikleri çıkarmayı öğrenen bir dizi blok ve katmandan geçer. Kodlayıcıdan gelen çıktılar daha sonra her nesne için sınırlayıcı kutuları ve etiketleri tahmin eden bir kod çözümüne aktarılır.

En basit kod çözümü, saf bir regresördür. Regresör, kodlayıcının çıkışına bağlanır ve her sınırlayıcı kutunun konumunu ve boyutunu doğrudan tahmin eder. Modelin çıktısı, nesnenin X, Y koordinat çifti ve görüntüdeki kapsamıdır. Basit olmasına rağmen, bu tür bir model sınırlıdır. Önceden kutu sayısını belirtmeniz gereklidir. Görüntüde iki köpek varsa ve model yalnızca tek bir nesneyi algılayacak şekilde tasarlanmışsa, biri etiketlenmeyecektir. Bununla birlikte her görüntüde önceden tahmin edilmesi gereken nesne sayısı biliniyorsa, saf regresör tabanlı modeller iyi bir seçenek olabilir.

Regresör yaklaşımının bir uzantısı, bir bölge teklif ağıdır. Bu kod çözümünde model, bir nesnenin bulunabileceğine inandığı bir görüntünün bölgelerini önerir. Bu bölgelere ait pikseller daha sonra bir etiket belirlemek (veya teklifi reddetmek) için bir sınıflandırma alt ağına beslenir. Daha sonra bu bölgeleri içeren pikselleri bir sınıflandırma ağı aracılığıyla çalıştırır. Bu yöntemin yararı, bir sınırlayıcı kutu

icerebilecek rastgele sayıda bölge önerebilen daha doğru ve esnek bir modeldir. Eklenen doğruluk, hesaplama verimliliği pahasına gelir.



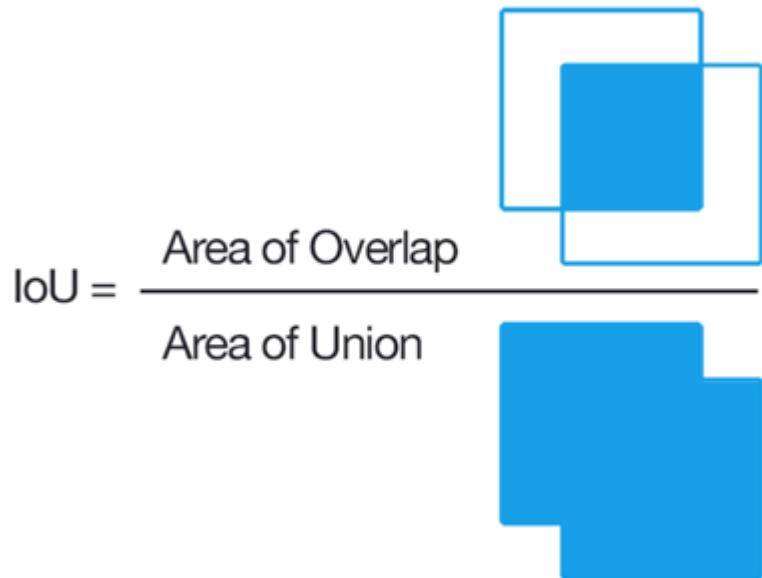
Şekil 2.1. Nesne tespiti algoritma yapısı [1]

Tek atışlı dedektörler (SSD) bir orta yol arar. Bölgeler önermek için bir alt ağ kullanmak yerine SSD'ler önceden belirlenmiş bir dizi bölgeye dayanır. Giriş görüntüsünün üzerine bir bağlantı noktası ızgarası yerleştirilir ve her bağlantı noktasında, birden çok şekil ve boyuttaki kutular bölge olarak işlev görür. Model, her çapa noktasındaki kutular için, bölgede bir nesnenin bulunup bulunmadığını dair tahmin, nesneye daha yakından uyması için kutunun konumu ve boyutunda değişiklikler yapar. Her bağlantı noktasında birden fazla kutu olduğundan ve bağlantı noktaları birbirine yakın olabileceğinden dolayı SSD'ler çakışan birçok potansiyel algılama üretir. Bu tahminlerin çoğunu ortadan kaldırmak ve en iyisini seçmek için SSD çıkışlarına son işlem uygulanmalıdır. En popüler uygulama, maksimum olmayan bastırma (non-maximum suppression) tekniğidir.

2.3 Metrikler

- Intersection over Union (IoU): Sınırlayıcı kutu tahmininin piksel seviyesinde kesin olması beklenemez. Bu nedenle, iki sınırlayıcı kutu arasındaki örtüşmenin

uzatılması için bir metriğin tanımlanması gereklidir. Union üzerinden kesişme tam olarak ne diyorsa onu yapar. İlgili sınırlayıcı kutuların kesişme alanını alır ve birleşme alanıyla böler. Bu, alanlararası örtüşme kalitesini temsil eden 0 ile 1 arasında bir puan belirtir.



Şekil 2.2. Intersection over Union [10]

- Ortalama Hassasiyet ve Ortalama Geri Çağırma: Kesinlik, tahminlerimizin ne kadar doğru olduğunu düşünürken, görüntüde bulunan tüm nesneleri tespit edip edemediğimizi açıklar. Ortalama Hassasiyet (AP) ve Ortalama Geri Çağırma (AR), nesne algılama için kullanılan iki yaygın ölçümüdür.
- Average Precision (AP) ve Mean Average Precision (mAP): Kesinlik-geri çağrıma eğrisinin altındaki alandır. AP, hem hassasiyeti hem de geri çağrımayı bir araya getirir. 0 ile 1 arasında bir değer alır. AP = 1 elde etmek için hem kesinliğin hem de geri çağrımanın 1'e eşit olması gereklidir. MAP, tüm sınıflar için hesaplanan AP'nin ortalamasıdır.

2.4. Nesne Tespiti Algoritma Çeşitleri

2.4.1. İki aşamalı nesne tespiti

İki Adımlı Nesne Tespiti, önce potansiyel olarak nesneleri içerebilecek sınırlayıcı kutuları tanımlayan ve ardından her bir sınırlamayı ayrı ayrı sınıflandıran algoritmaları içerir.

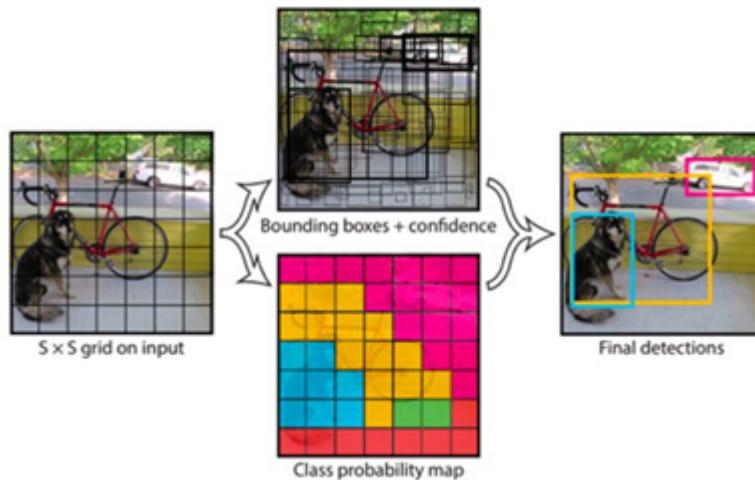
İlk adım, daha sonra ortak DL tabanlı sınıflandırma mimarilerine aktarılan bir dizi bölge sağlayan Bölge Teklif Ağı gerektirir. RCNN'lerdeki (son derece yavaş olan) hiyerarşik gruplama algoritmasından, Fast RCNN'lerde CNN'leri ve ROI havuzlamasını kullanmaya ve Faster RCNN'lerde çapaları kullanmaya (böylece boru hattını ve uçtan uca eğitimi hızlandıran) birçok farklı yöntem ve varyasyon vardır ve bu bölge teklif ağlarına (RPN'ler) sağlanmıştır.

Bu algoritmaların tek adımlı nesne algılama benzerlerinden daha iyi performans gösterdiği bilinmektedir, ancak karşılaştırıldığında daha yavaştır. Yıllar içinde önerilen çeşitli iyileştirmelerle, İki Adımlı Nesne Tespiti ağlarının gecikmesindeki mevcut darboğaz, RPN adımıdır.

2.4.2. Tek aşamalı nesne tespiti

Gerçek zamanlı nesne algılama ihtiyacı ile algılama ve sınıflandırma aşamasını birleştirmeye çalışan YOLO, YOLOv2, YOLOv3, SSD, RetinaNet gibi birçok tek aşamalı nesne algılama mimarisi önerilmiştir.

Bu algoritmaların en büyük başarılarından biri, sınırlayıcı kutu tahminlerini "geriletme" fikrini ortaya koymaktır. Her sınırlayıcı kutu birkaç değerle (örneğin, xmin, xmax, ymin ve ymax) kolayca temsil edildiğinde, algılama ve sınıflandırma adımını birleştirmek ve ardışık düzeni önemli ölçüde hızlandırmak daha kolay hale gelir.



Şekil 2.3. YOLO algoritmasının işleyiş düzeni [16]

Örneğin, YOLO, tüm görüntüyü daha küçük ızgara kutularına böler. Her ızgara hücresi için, bu ızgara hücresinden geçen her sınırlayıcı kutunun sınıf olasılığını ve x ve y koordinatlarını tahmin eder.

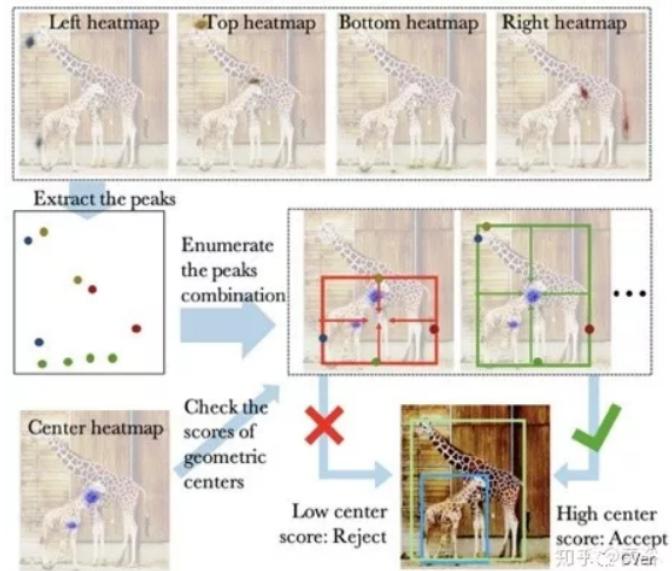
Bu değişiklikler, tek aşamalı dedektörlerin daha hızlı çalışmasını ve aynı zamanda küresel düzeyde çalışmasını sağlar. Bununla birlikte, her sınırlayıcı kutu üzerinde ayrı ayrı çalışmadıkları için, bu, yakın çevrede daha küçük nesneler veya benzer nesneler olması durumunda daha kötü performans göstergelerine neden olabilir.

2.5. Isı haritası tabanlı nesne tespiti

Isı haritası tabanlı nesne tespiti, bir anlamda, tek atış tabanlı nesne tespitinin bir uzantısı olarak düşünülebilir. Tek atış tabanlı nesne algılama algoritmaları, sınırlayıcı kutu koordinatlarını (veya ofsetlerini) doğrudan geriletmeye çalışırken, ısı haritası tabanlı nesne algılama, sınırlayıcı kutu köşelerinin / merkezinin olasılık dağılımını sağlar.

Isı haritalarında bu köşe / merkez tepe noktalarının konumlandırılmasına bağlı olarak, sonuçta ortaya çıkan sınırlayıcı kutular tahmin edilir. Her sınıf için farklı bir ısı haritası oluşturulabildiğinden, bu yöntem aynı zamanda algılama ve sınıflandırmayı da birleştirir. Isı haritası tabanlı nesne algılama şu anda yeni araştırmalara öncülük ederken, hala geleneksel tek vuruşlu nesne algılama

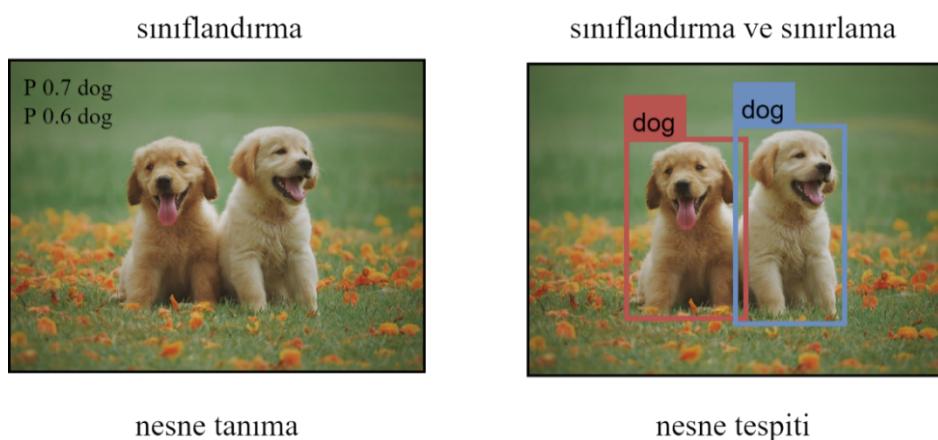
algoritmaları kadar hızlı değil. Bunun nedeni, bu algoritmaların saygın bir doğruluk elde etmek için daha karmaşık omurga mimarileri (CNN'ler) gerektirmesidir.



Şekil 2.4. Isı haritası tabanlı nesne algılama modeli [1]

3. NESNE TESPİTİİNDE KULLANILAN YÖNTEMLER

Nesne tespiti için kullanılan yöntemler önceleri, resim üzerindeki tanımlayıcı bilgilerin çıkarılması, işlenmesi ve (genellikle) tek katmanlı makine öğrenmesi algoritmalarına sokulmasıyla gerçekleşmekteydi. Tespit edilmesi istenen nesne ve yer aldığı arka plan arasında görsel bir fark oluşturulup resim üzerindeki ayırt edici bilgilere çağrıyan (cascade) modeliyle bakılmaktaydı. Çağlayan modeli resimdeki birçok bölgede negatif ve pozitif olarak ayırmalar yapıp özellik olarak çıkarılabilen tanımlamaları dönmektedir. Tek katmandan oluşan bu makine öğrenmesi algoritması ve uygulamalarından sonra Viola ve diğerleri [12] çok katmanlı çağrıyan modelini sundular. Bu çalışma tek katmanlı modelle aynı başarı oranını sağlamakta fakat işlem süresini kayda değer ölçüde azaltmaktadır. Viola'nın çalışmasının ardından Lienhart ve diğerleri [13] bu çok katmanlı modele 45 derece eğilmiş verileri de sürece katarak benzeşim oranını arttırdılar. İşlenme sonrası optimizasyon değişiklikleriyle daha performanslı bir sonuç elde ettiler. Derin öğrenme öncesi HOG (Histogram of Oriented Gradients; Yönelimli Gradyanların Histogramı) [14] ve Deforme Olabilen Parçalar Modeli (DPM, Deformable Part Models) ve getirdiği çevreleyen kutu regresyonu (bounding box regression) [15] yaklaşımlarıyla birlikte nesne tespiti üzerine gelişmeler devam etmiştir. [5]



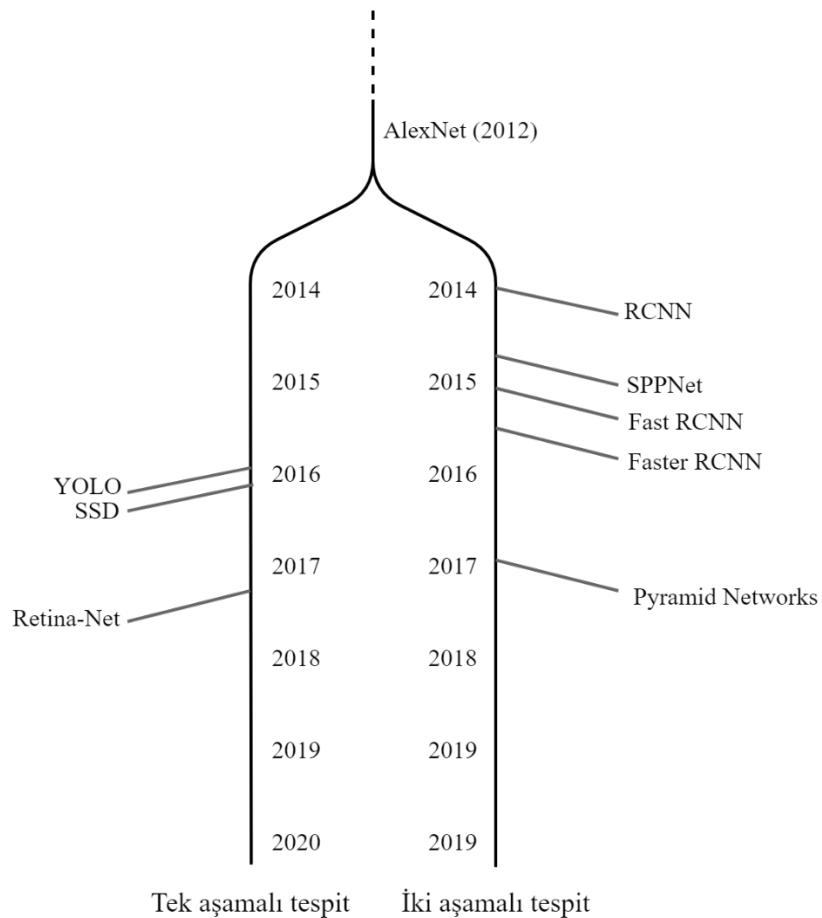
Şekil 3.1. Nesne tanıma ve nesne tespiti işlemlerinin gerçekleştirimi

3.1. CNN Modelinin Gelişimi

2011 yılının başlangıcında GPU'ların (graphics processing unit; grafik işlemci birimlerinin) hızının önemli ölçüde artmasıyla evrişimli nöron ağlarında her katmanın ön eğitime sokulması gerekliliği ortadan kalkmıştır. Model eğitmenin maliyeti ve hız konusunda olumlu bir ilerleme olarak görülebilecek bu aşamadan sonra, derin öğrenmenin diğer makine öğrenme yaklaşımlarına kıyasla daha verimli ve hızlı sonuçlar verdiği çalışmalarla kanıtlanmıştır. [5]

Bu örneklerden biri olarak gösterilebilecek AlexNet yapısı, CNN algoritması üzerine kurulan bir derin öğrenme modelidir. Bu model, çalışmanın yayınladığı yıllarda yapılan uluslararası yarışmaların çoğunda birincilikler elde etmiştir. Düzenlenmiş doğrusal birimlerin (ReLU) kullanımı sayesinde hız ve seyreltme oranlarında iyileşmeler sağlanmıştır. [17]

GPU'ların ve derin öğrenmenin öne çıkışından sonra gelişen nesne tespit modelleri temel düzeyde ikiye ayrılmaktadır: Resmin bölündüğü her bir küçük bölmeden sabit sayıda tahmin yapılan (tek aşamalı tespit) veya bir ağ yardımıyla bulunan olası nesnelerin yaptığı öneriyi takip eden başka bir ağın bu sonuçları değerlendirip nihai bir çıktı üretmesiyle oluşan bir modeldir (iki aşamalı tespit). [18]



Şekil 3.2. Derin öğrenme algoritmalarının gelişimi [5]

3.2. CNN Temelli İki Aşamalı Tespit Modelleri

3.2.1. RCNN

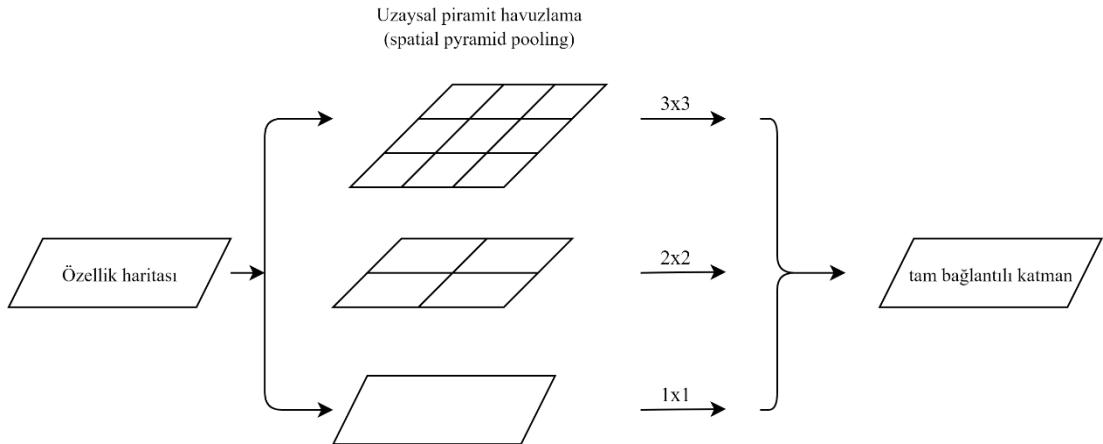
CNN algoritması üzerine yapılan modellerden RCNN (Region-based Convolutional Neural Network; Bölge-Temelli Evrişimli Nöron Ağrı) yaklaşımı 2014 yılında, nesne tespiti konusunda yeni bir dönemin kapısını açmıştır. RCNN, seçici arama ile bir dizi nesne teklifinin (aday kutularıyla) çıkışlarıyla başlar. Sonra her bir olası nesne teklifi, sabit boyutlu bir görüntüye yeniden oranlanır ve Imagenet'te eğitilmiş bir CNN modeline beslenir ve özellikleri çıkarılır. Son aşamada ise lineer destek vektör makinesi (SVM) sınıflandırması kullanılarak her bölgedeki nesneler ayrı ayrı tahmin edilir ve kategorisi belirlenir.

RCNN çoğu ölçekte büyük ilerleme kaydetse de dezavantajları da bulunmaktadır. Bu dezavantajlardan en büyüğü ise hesaplama hızı konusunda olmuştur. Resim üzerindeki bölmelerde yapılan tekliflerin sayısının çokluğu (bir resimde 2000'den fazla kutu) oldukça yavaş bir tespit hızına (GPU ile resim başına 14 saniye) sebep olmaktadır. Aynı yılın ilerleyen aylarında önerilen Mekânsal Piramit Havuz Ağları (SPPNet) ile bu yüksek hesaplama süresi konusunda iyileştirmeler yapılmıştır.

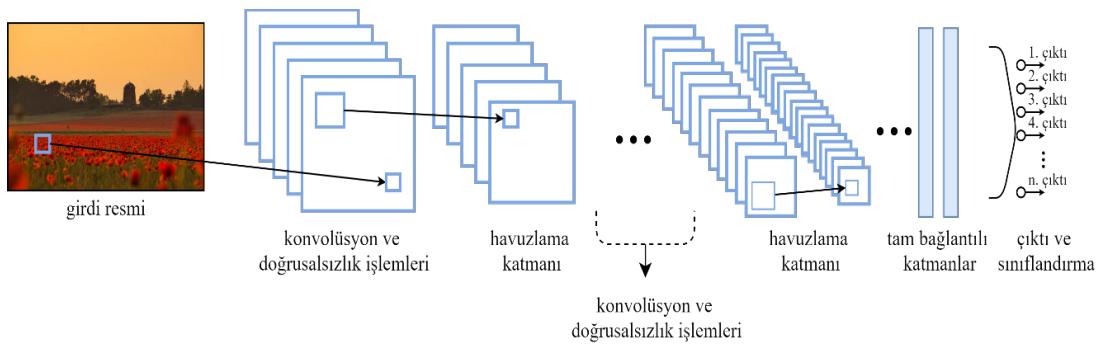
3.2.2. SPPNet

SPPNet öncesi CNN modellerinde sistemler sabit boyutlu bir giriş gerektirmektedir. Örneğin AlexNet için bu değer 224×224 'tü. SPPNet'in alana esas katkısı da bu giriş değerleri aşamasında olmuştur. Yeni önerilen mekânsal piramit havuzlama katmanı sayesinde tekrar ölçeklendirme gerekmeksiz ve ilgilenilen görüntünün/bölgelenin büyüklüğünden bağımsız olarak sabit uzunlukta bir temsil oluşturulması sağlanmıştır.

SPPNet modelinde, özellik haritaları tüm görüntüden yalnızca bir kez hesaplanır ve ardından konvolüsyon özelliklerini art arda hesaplamaktan kaçınır. Bu işlemden sonra, tespit katmanını eğitmek için, istege bağlı oluşturulan bölgelerin sabit uzunluk gösterimleri oluşturulur. Bu sabit uzunluk gösterimleri konvolüsyon katmanın tek tek tekrar çalışıp hesap yapmasını engellemiştir. Bu yenilikçi yaklaşımı getiren SPPNet modeli RCNN modelinden 20 kat daha hızlı çalışmaktadır. Bu hızı elde etmekle birlikte nesne tespit doğruluğunda olumsuz yönde bir etki de gözlemlenmemiştir. Yeni model tespit hızını etkili bir şekilde arttırmış olmasına rağmen yine de dezavantajlar taşımaktadır. Bunlardan ilki, ağıın eğitiminin hâlâ iki aşama olarak gerçekleşmesi üzerindedir. İkinci eksik noktası ise, bağlı katmanlardaki değerler bir çıkış elde etmek için incelenirken önceki tüm katmanlarda elde edilmiş veriler görmezden gelinir. (Çıkış bilgilerinin aynı zamanda diğer katmanların girişi olduğu modellerde önceki katmanlardaki veriler de dikkate alınmaktadır. Bu modelde ise özellikler tek katmanda sonuç olarak birleştirilmektedir.)



Şekil 3.3. SPPNet modelinin diğer modellerden farklı işleyen havuz katmanı [31]



Şekil 3.4. Evrişimli nöron ağlarının temel işleyiş adımları

Aynı sene sonunda geliştirilen Fast RCNN modeli bu problemlere çözümler sunmuştur.

3.2.3. Fast RCNN

Fast RCNN modeli aynı ağ altında tespit katmanı ve sınırlayıcı kutu regresyonunu (bounding box regressor) aynı anda eğitmeyi sağlar. Bu eşzamanlı eğitim işlemi Fast RCNN modeline hız ve doğruluk metrikleri özelinde olumlu sonuçlar döndürür. VOC07 veri seti üzerinde, bir önceki RCNN modeline kıyasla %20 daha başarılı sonuç vererek ortalama kesinliklerin ortalaması (mAP) kıstasında %70'lik bir başarıya ulaşmasını sağlamıştır. Bu kıstas, nesne tespiti yapan modellerin karşılaştırılmasında kullanılan en önemli değerlerdir. (Bir makine öğrenmesi modelinde kesinlik, doğru olarak tahmin ettiğimiz değerlerin gerçekten kaç adedinin doğru olduğunu göstermektedir. mAP değeri %70 olan bir modelde, görüntülerde

tespit edilen nesnelerin %70'inin gerçekten olması beklenen sonucu gösterdiğini söyleyebiliriz.)

Bu başarı oranına ek olarak nesne tespiti işlemi RCNN modeline kıyasla 200 kat daha hızlı gerçekleşmektedir. [5]

Fast RCNN, RCNN ve SPPNet'in avantajlarını başarıyla bütünlüğünü sağlayarak tespit hızı hala istenilen seviyede değildir ve gerçek zamanlı nesne tespit uygulamalarında kullanılmaya yaklaşamamaktadır. Fast RCNN de dahil olmak üzere yukarıda bahsedilen nesne tespit algoritmaları, gerçek zamanlı uygulamalarda kullanılmaya hızları dolayısıyla uygun değildir. Bunun başlıca sebebi de ilgi bölgelerinin tespit edilebilmesi için resimden çıkarılan bilgilerin katmanlara özellik (feature) olarak teklif edilmesine bağlı olmasından kaynaklanır. Yani bir bölgede nesne olup olmadığını anlayabilmek için model, alınan görüntü üzerindeki verileri katmanlara sokarak eğitilmiş verisi üzerinden bir öneri gerçekleştirmek zorundadır. Nesne olma olasılığı olan bölgelerin tespitinden sonra bu bölgelerde: ayırt edici nesne özellikleri aramaktadır.

İlgili bölgelerinin uzun süreler boyunca ağ içerisinde aranması tespit işlemini yavaşlatmaktadır ve modellerin gerçek hayat senaryolarında kullanımını sınırlamaktaydı. Bahsi geçen yaklaşımın ardından, Fast RCNN modelinin çıktığı aynı senede Faster RCNN modeli bu kısıtlamayı kaldırmayı başardı. Modelde eklenen bölge teklif ağı (RPN, Regional Proposal Network) yapısı neredeyse maliyetsiz bölge önerileri sağlamayı başardı.

Faster RCNN gerçek zamanlı nesne tespiti yapmaya uygun derin öğrenme algoritmalarının başlangıcı kabul edilir.

3.2.4. Faster RCNN

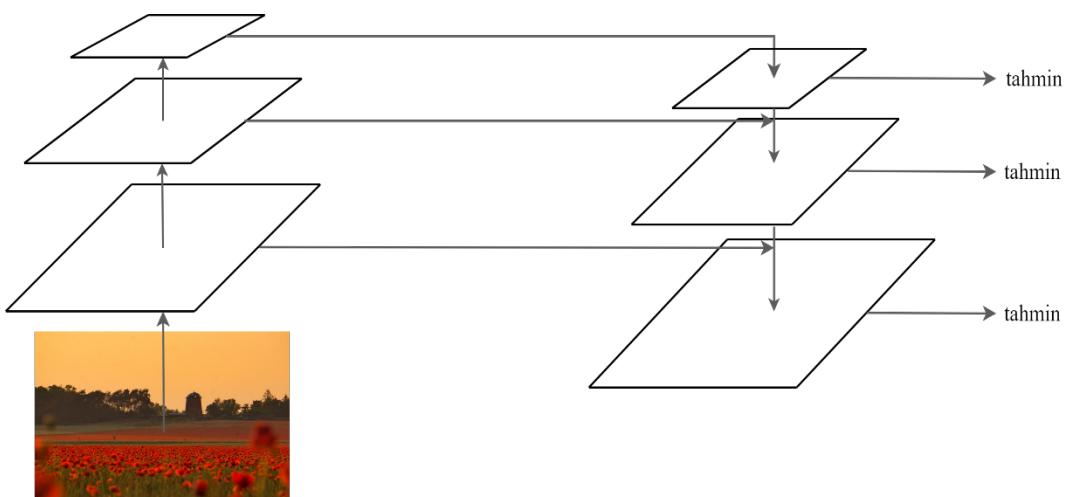
Faster RCNN modelinde RCNN modelinden farklı bir yaklaşım olarak: nesne tespiti aşamasındaki tüm ayrı bloklar, kademeli olarak uçtan uca birleşik bir çatı altında

toplannmıştır. Bu bloklar: özellik çıkarımı, öneri algılama, sınırlayıcı kutu regresyonu vb. olabilir.

Faster RCNN, Fast RCNN'in tespit hızı konusundaki kısıtlamasını aşabilmiş olmasına rağmen sonraki algılama aşamalarında hâlâ bir hesaplama fazlalığına sahiptir. Bu da modelin hızlı çalışmasına ve neredeyse gerçek zamanlı olarak kullanılabilecek noktaya gelmesine rağmen arzu edilen seviyede sonuç veremediği anlamına gelir.

3.2.5. Feature Pyramid Networks

2017 yılında geliştirilen Özellik Piramit Ağları (FPN, Feature Pyramid Networks) modeli Faster RCNN'in temellerine yönelik bir yeniden ele alımıdır. FPN'den önceki derin öğrenme bazlı tespit sistemlerinin çoğu derin katmanlardaki özellikler kategori tanıma için faydalı olmasına rağmen hesaba katılmamaktaydı. Kategori tanıma işi bir ağın yalnızca en üst katmanında gerçekleşmekteydi. [C10] FPN'de baştan aşağı kullanılan yanal bağlantılarla sahip mimari, tüm ölçeklerde üst düzey anlamsal çıkarımı garanti etmektedir. Bir CNN modeli ileri yayılma (forward propagation) işlemiyle bir özellik piramidi elde eder. FPN, çok çeşitli ölçeklerle nesneleri tespit etme konusunda büyük ilerlemeler göstermiştir. Temel bir Faster RCNN sistemi içeresine yerleştirilen FPN yapısı nesne tespitinde daha önceki sistemlerden çok daha büyük bir başarı oranı yakalamıştır.



Şekil 3.5. Özellik Piramit Ağlarında ölçeklendirilen her orijinal resim tahmin aşamasında tekrar işleme dahil edilir [21]

FPN yapısı, günümüzde kullanılan birçok tespit modelinin temel bir katmanı olarak yerini almaktadır.

3.3. CNN Temelli Tek Aşamalı Tespit Modelleri

3.3.1. You Only Look Once (YOLO)

Tek aşamalı tespit modellerinin başlangıcı YOLO ile 2015 yılında olmuştur. YOLO algoritması son derece hızlıdır. Saniye başına 155 karede işlem yapabilen YOLO'nun hızlı modeli %52,7 mAP değerine ulaşmıştır. Gelişmiş modeller ise saniye başına 45 karede %63,4 mAP değeriyile tespit doğruluğu yönünden iyileştirmeler getirmiştir. [5]

Önceki modellerde yer alan ve art arda gerçekleşen teklif algılama ve doğrulama adımları terk edilerek tüm görüntüye başlangıçta tek bir nöron ağı uygulanmasıyla bölgelerin tahmin edilmesi ve her bölge için aynı anda sınırlayıcı kutular çizilip olasılıkların bulunması işlemi gerçekleştirılmıştır. YOLO üzerine yapılan diğer gelişmeler (V2 ve V3 gibi) [22, 23] yüksek algılama hızlarında tespit doğruluğunu artıran geliştirmeler yapmıştır. Tespit hızındaki büyük gelişmelere rağmen YOLO modeli, iki aşamalı tespit modelleriyle kıyaslandığında yerleştirme doğruluğunda bir düşüş yaşamaktaydı. Bu düşüş özellikle küçük nesnelerde oldukça belirgin bir şekilde ortaya çıkmaktaydı. YOLO'nun ardından gelen SSD modeli ve YOLO'nun sonraki versiyonları bu soruna eğilmişlerdir.

3.3.2. Single Shot MultiBox Detector (SSD)

Tek Atış Çoklu Kutu Tespiti (SSD, Single Shot MultiBox Detector) [24] modeli ise YOLO'nun ardından yine aynı sene içerisinde geliştirildi. SSD ile tanıtılan birden çok referans ve birden çok çözümürlük tespit teknikleri modelin, özellikle küçük nesnelerde olmak üzere tespit doğruluğunu önceki modellere kıyasla önemli ölçüde artırmıştır. SSD ile önceki tüm tespit modelleri arasında, tespit işleminin

gerçekleşme yeri ve aşamalar konusunda farklılıklar vardır. Oncekilerde 5 farklı ölçek için ağıın farklı katmanlarında gerçekleşen algılama işlemleri SSD ve sonrasında yalnızca en üst katmanda çalışmaktadır.

3.3.3. RetinaNet

Yüksek hız ve basitliklerine rağmen, tek aşamalı tespit modelleri yıllarca iki aşamalı modellerin doğruluk oranlarının arkasından gitmek zorunda kalmışlardır. Buna sebep olan sorun 2017 yılında RetinaNet'le birlikte işaret edilip aynı modelle birlikte çözülmeye çalışılmıştır. [25]

Tek aşamalı modellerde ağıın en uç noktasında tespit işlemini yapan yoğunluk tespit katmanı bulunmaktadır. Çalışmada, modellerin eğitimi sırasında bu katmanda karşılaşılan aşırı ön ve arka plan dengesizliğinin başarı oranının düşüklüğüne sebep olduğu ortaya atıldı. Bu amaçla, standart çapraz entropi kaybını (cross entropy loss) yeniden şekillendirerek "odak kaybı" (focal loss) adlı yeni bir kayıp fonksiyonu tanıtıldı yoğunluk tespit katmanları, eğitim sırasında zorlanılan; yanlış sınıflandırılmış örnekler daha fazla odaklanarak modelin hata oranının düşmesini sağladı. Bu kayıp fonksiyonu sayesinde tek aşamalı tespit modelleri yüksek algılama hızlarını korumaya devam ederek doğruluk anlamında iki aşamalı modellerle aynı seviyeye gelebildi.

3.4. Derin Öğrenme Algoritmalarında Tespit İşleminin Hızlandırılması

RCNN modeli ve sonrasında bu modelden yola çıkarak geliştirilen diğer benzer modeller, her adımında nesne tespiti konusundaki bilgimize katkıda bulunmuştur. Modelin hızı, doğruluğu, kullanım senaryoları, modelde tespit edilebilecek nesneler, modele giren verilerin boyutu, özellikleri gibi birbirinden farklı değişkenlere çözümler sunan bu gelişmelerin ardından üzerine gidilmesi gereken sorun, gerçek zamanlı nesne tespitinin düşük işlem maliyetiyle, hızlı ve yüksek doğrulukta gerçekleşmesi konusu olmuştur.

Tespit işleminin hızlandırılması maksadıyla yapılan çalışmalar kabaca üçe ayrılmaktadır: tespit hattının hızlandırılmasına yönelik çalışmalar, tespit motorunun hızlandırılmasına yönelik çalışmalar ve sayısal hesaplamalara yönelik çalışmalar.

Tespit hattının hızlandırılması kapsamında yapılan çalışmalar, bölge teklifi, özellik çıkarımı, aktivasyon fonksiyonları yardımıyla verilerin vurgulanması ve resimdeki en önemli bilgilerin saklanarak verilerin boyutunun küçültülmesi gibi aşamalar etrafında şekillenmektedir. Oluşturulan ağdaki katmanların yapısı, sayısı, sırası ve giren verinin özellikleri gibi durumlar ilgilenilen ve değiştirilmeye çalışılan ana etmenlerdir.

Tespit motorunun hızlandırılmasına yönelik çalışmalar, modelin çalıştırıldığı yapının hesaplama gücünün arttırılmasıyla alakalı konuları kapsar. Daha yüksek işlem gücüne sahip birimlerde çalıştırılan modeller daha kısa sürede sonuç vermekte ve doğruluktan taviz vermeye gerek kalmadan bile gerçek zamanlı uygulamalarda kullanılabilecek duruma gelebilmektedirler.

Sayısal hesaplamalara yönelik çalışmalar, ağdaki katmanlar içerisinde gerçekleşen matematiksel işlemlerin, sayısal hesaplamaların, modellemelerin iyileştirilmesi, daha az işlem maliyeti gerektirecek şekilde yenilenmesi, değiştirilmesi ve geliştirilmesi gibi konuları kapsar.

Bundan sonraki yeni ve başarılı modeller bu üç konu üzerine yoğunlaşarak geliştirilmiştir. Bu modellere örnek olarak Bölge-temelli Tamamen Bağlı Evrişim Ağları (RFCN, Region-based Fully Convolutional Network) [19] ve Hafif RCNN (Light head RCNN) [20] verilebilir.

3.5. Modelin Seçimi

Nesne tespiti için kullanılan temel yaklaşımın; bu yaklaşımın avantajları, dezavantajları ve başarı oranlarının anlatılmasından sonra, bu yaklaşımı kendi uygulamalarında kullanmak isteyen kullanıcılar için hangi modelin seçilmesi gereği sorusu önem kazanmaktadır. Bu soruya genel geçer bir cevap verebilmenden

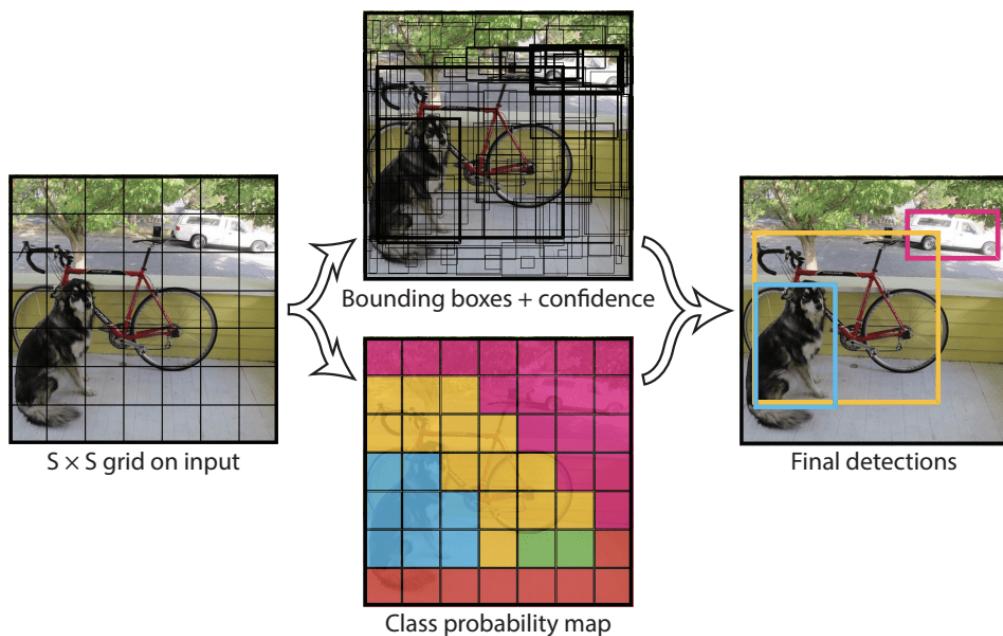
zorluğu da göz önüne alınarak farklı senaryolar için bazı çıkarımlar yapılabilmektedir.

Bu çalışmanın ve projede amaçlanan hedefin insansız hava araçlarının kamerasından alınan görüntü üzerinde gerçek zamanlı nesne tespiti yapması olduğu göz önüne alınırsa hafif ağ yapısı, gerçek zamanlı ve iki aşamalı tespit modellerine yaklaşan başarı oranıyla YOLO modeli ve bazı özelleştirilmiş versiyonları gerçek zamanlı nesne tespit uygulamalarında kullanmak üzere seçilebilir.

4. YOLO ALGORİTMALARININ KARŞILAŞTIRILMASI

Bu bölümde, YOLO algoritmasının detaylarını ve her bir varyasyona ait farklı yönlerin ve başarı oranlarının karşılaştırılması yer almaktadır.

4.1 YOLO Algoritması



Şekil 4.1. YOLO algoritmasının işleyiş düzeni [32]

YOLO (You Only Look Once) algoritması, iyi bir doğrulukla gerçek zamanlı olarak nesne algılaması yapabilen son teknoloji bir nesne dedektörüdür. Son yıllarda çok kullanılan algoritmaların başında gelir. Diğer algoritmalarдан bazıları daha iyi tahmin yapabilenler olsa da yavaşırlar. Ancak YOLO algoritmaları gerçek zamanlı nesne tespiti hızlı olmaları sebebiyle daha iyi uygular.

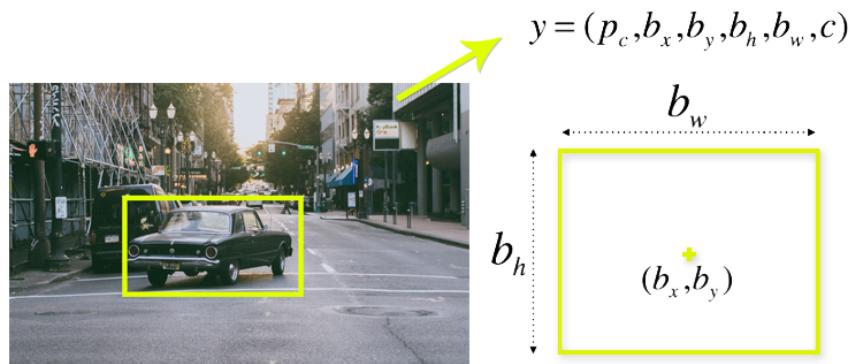
YOLO ailesi, Single Shot sınıfındadır. Dolayısıyla, görüntünün ilgi noktalarını seçmek yerine tek seferde çalıştırıldığında tüm nesnelerin sınıflarını ve sınırlayıcı

kutularını tahmin eder. Doğruluktan bir miktar ödün vererek hızda iyileştirme yaparlar.

Temelinde sınırlayıcı kutular yatar ve dört değişken tarafından tanımlanırlar:

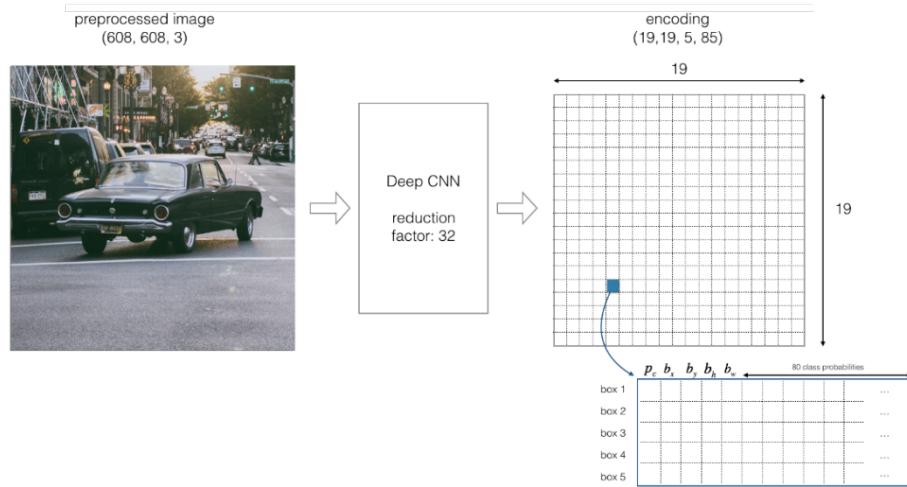
- kutunun merkezi (b_x, b_y)
- genişlik (b_w)
- yükseklik (b_h)
- nesnenin sınıfına karşılık gelen değer (c)

Bunlara ek olarak, sınırlayıcı kutuda nesne olma olasılığı olan p_c değerini tahmin etmek gereklidir.



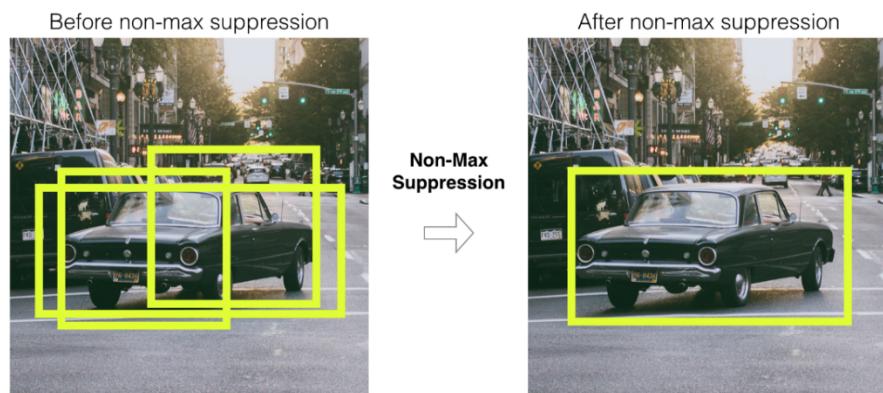
Şekil 4.2. Sınırlayıcı kutu tanımı [30]

YOLO algoritmasıyla çalışırken, görüntümüzde potansiyel nesne içerebilecek bölgeleri aramak yerine ızgaraya bölünür (genelde 19x19). Her hücre, 5 sınırlayıcı kutuyu tahmin eder. Bu nedenle birçok sınırlayıcı kutu elde edilir.



Şekil 4.3. Sınırlayıcı kutuların matrisi [30]

Bu hücrelerin ve sınırlayıcı kutuların çoğu bir nesne içermez. Bu nedenle, non-max suppression adı verilen bu süreçte düşük olasılığı olan ve en yüksek paylaşılan alana sahip sınırlayıcı kutuları kaldırırmaya yarayan p_c değeri tahmin edilir.



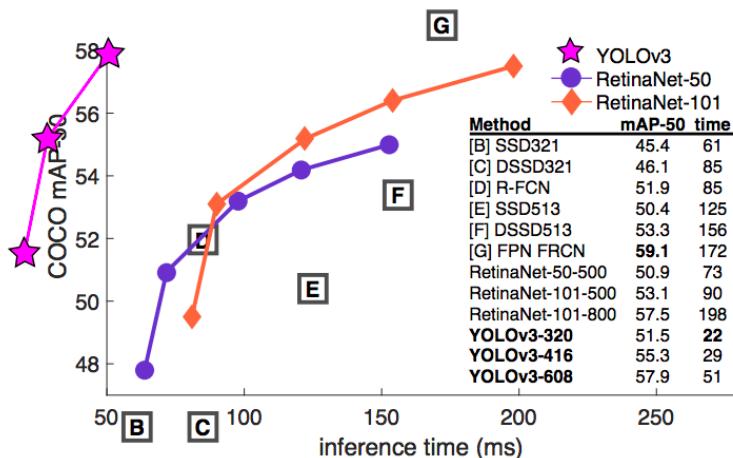
Şekil 4.4. Non-max suppression metoduyla sınırlayıcı kutuların azaltılması [30]

4.2. YOLO Sürümüleri

YOLO ailesi günümüzde altı varyasyondan oluşmaktadır. YOLOv1 2016 yılında, YOLOv2 2017 yılında, YOLOv3 2018 yılında, geri kalan üç sürüm (2020) ise yılında yayınlandı.

YOLO ilk olarak 2016 yılında tanıtıldı ve nesneleri gerçek zamanlı olarak daha yüksek doğrulukla algılama yeteneği sebebiyle nesne tespit alanında kilometre taşı oldu.

4.2.1 YOLOv3



Şekil 4.5. YOLOv3’ün diğer modellere karşılaştırılması [32]

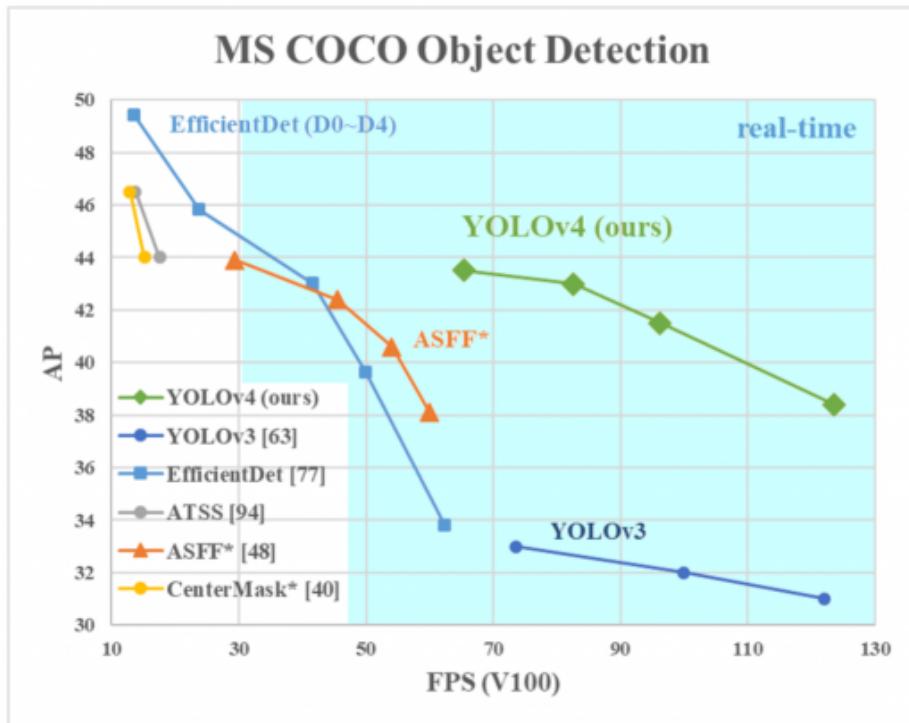
YOLOv3 [23] 2018 yılında piyasaya çıktı. Oncekilere kıyasla, herhangi bir yeniden eğitim yapılmadan model boyutunu değiştirerek hız ve doğruluk arasında kolayca değişim tokuş yapılmasını sağlar.

Bu versiyon, C ve CUDA ile yazılmış olan Darknet’e dayanmaktadır. Darknet, ağır temel mimarisini belirler ve YOLO’yu eğitmek için framework olarak kullanılır.

Tablo 4.1. YOLOv3 performans tablosu [33]

Model	size (pixels)	mAP ^{val} 0.5:0.95	mAP ^{test} 0.5:0.95	mAP ^{val} 0.5	Speed V100 (ms)		params (M)	FLOPS 640 (B)
YOLOv3-tiny	640	17.6	17.6	34.8	1.2		8.8	13.2
YOLOv3	640	43.3	43.3	63.0	4.1		61.9	156.3
YOLOv3-SPP	640	44.3	44.3	64.6	4.1		63.0	157.1

4.2.2 YOLOv4



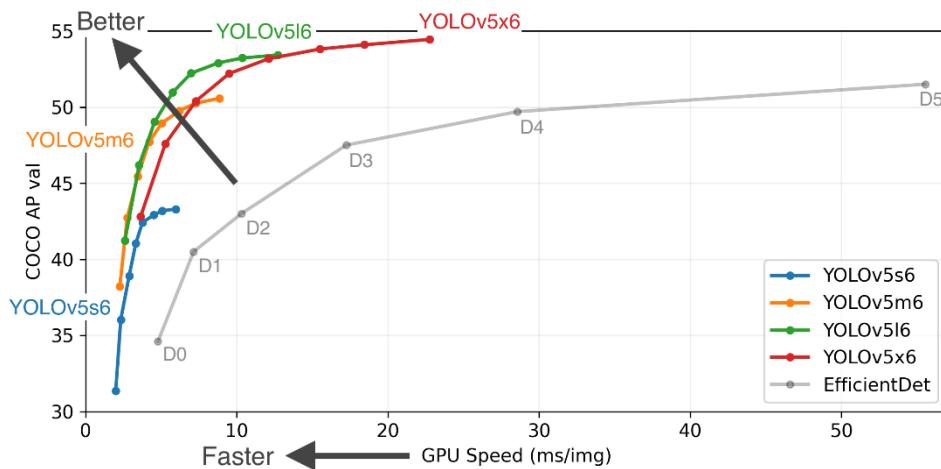
Şekil 4.6. YOLOv4’ün diğer algoritmalarla karşılaştırılması [28]

YOLOv4 2020’de çıktıktan sonra nesne algılama için en hızlı ve en doğru gerçek zamanlı model olarak kabul edildi. YOLO v4, son teknoloji ürünü BoF ve birkaç BoS'nin güçlerini bünyesinde barındırır. BoF, çıkarım süresini artırmadan dedektörün doğruluğunu artırır. Öte yandan BoS, çıkarım maliyetini küçük bir miktar artırır, ancak nesne tespitinin doğruluğunu önemli ölçüde iyileştirir. V3 varyasyonu gibi Darknet'e dayanır.

Tablo 4.2. YOLOv4 performans tablosu [34]

Model type	AP	AP50	AP75	APS	APM	APL
DarkNet (YOLOv4 paper)	0.471	0.710	0.510	0.278	0.525	0.636
Pytorch (TianXiaomo)	0.466	0.704	0.505	0.267	0.524	0.629
TensorRT FP32 + BatchedNMSPlugin	0.472	0.708	0.511	0.273	0.530	0.637
TensorRT FP16 + BatchedNMSPlugin	0.472	0.708	0.511	0.273	0.530	0.636

4.2.3 YOLOv5



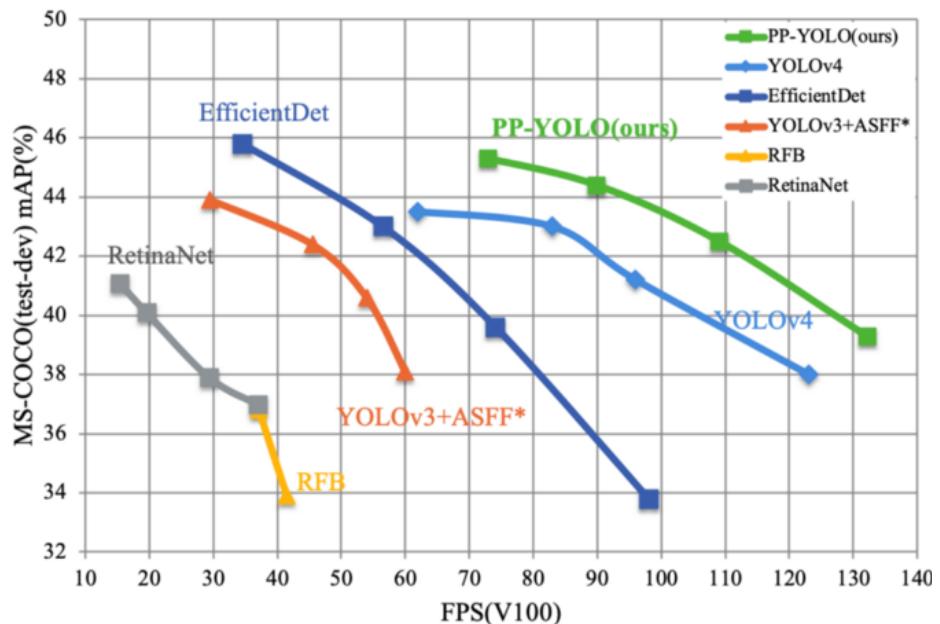
Şekil 4.7. YOLOv5'in varyasyonlarının karşılaştırılması [35]

YOLO v5, diğer tüm önceki sürümlerden farklı olarak Darknet'tin fork'u yerine bir PyTorch uygulamasıdır. YOLO v4'e benzer, YOLO v5'in bir CSP backbone ve PANNET neck vardır.

Tablo 4.3. YOLOv5 performans tablosu [35]

Model	size (pixels)	mAP ^{val} 0.5:0.95	mAP ^{test} 0.5:0.95	mAP ^{val} 0.5	Speed V100 (ms)	params (M)	FLOPS 640 (B)
YOLOv5s	640	36.7	36.7	55.4	2.0	7.3	17.0
YOLOv5m	640	44.5	44.5	63.1	2.7	21.4	51.3
YOLOv5l	640	48.2	48.2	66.9	3.8	47.0	115.4
YOLOv5x	640	50.4	50.4	68.8	6.1	87.7	218.8
YOLOv5s6	1280	43.3	43.3	61.9	4.3	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	54.4	54.4	72.0	22.4	141.8	222.9
YOLOv5x6 TTA	1280	55.0	55.0	72.0	70.8	-	-

4.2.4 PP-YOLO



Şekil 4.8. PP-YOLO'nun diğer varyasyonlarla karşılaştırılması [29]

PP-YOLO, YOLO v3 modeline dayanmaktadır. Yeni bir algılama modeli olmaktan ziyade, gerçek senaryoları doğrudan uygulanabilen, nispeten dengeli ve verimli olan bir nesne algılayıcısı uygular. Platformun boyutunu azaltmaya yardımcı olan ve yüksek performanslı dağıtımı mümkün kıلان veri artırma yöntemler sağlar.

Tablo 4.4. PP YOLO performans tablosu [29]

YOLOv3 + ASFF* [26]	Darknet-53	320	60	-	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF* [26]	Darknet-53	416	54	-	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv3 + ASFF* [26]	Darknet-53	608	45.5	-	42.4%	63.0%	47.4%	25.5%	45.7%	52.3%
YOLOv3 + ASFF* [26]	Darknet-53	800	29.4	-	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%
YOLOv4 [1]	CSPDarknet-53	416	96	164.0*	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4 [1]	CSPDarknet-53	512	83	138.4*	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4 [1]	CSPDarknet-53	608	62	105.5*	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
PP-YOLO	ResNet50-vd-dcn	320	132.2	242.2	39.3%	59.3%	42.7%	16.7%	41.4%	57.8%
PP-YOLO	ResNet50-vd-dcn	416	109.6	215.4	42.5%	62.8%	46.5%	21.2%	45.2%	58.2%
PP-YOLO	ResNet50-vd-dcn	512	89.9	188.4	44.4%	64.6%	48.8%	24.4%	47.1%	58.2%
PP-YOLO	ResNet50-vd-dcn	608	72.9	155.6	45.2%	65.2%	49.9%	26.3%	47.8%	57.2%

4.3. Gerçek Zamanlı Nesne Tespit Algoritma ve Sürümelerinin Performans ve Başarı Oranlarının Karşılaştırılması

Çalışmanın bir önceki başlıklarında incelenen nesne tespiti algoritmalarından YOLO ve YOLO'nun farklı sürümlerine ek olarak gerçek zamanlı olarak kullanılabilen bazı diğer nesne tespit algoritmaları kullanılanlara farklı hedefler için çözümler sunulmaktadır. Gerçek zamanlı nesne tespiti algoritmalarının seçimi projeden projeye, uygulamadan uygulamaya ve istenen kriterlere göre değişiklik göstermektedir. Bazı algoritmaların resim çözünürlüğü, FPS ve backbone metrikleri açısından aşağıda karşılaştırma olarak sunulmaktadır.

SSD ve R-FCN algoritmalarının COCO veri seti üzerinde canlı nesne tespitiyle ilgili FPS, backbone ve mAP değerleri aşağıdaki tabloda verilmiştir:

Tablo 4.5. SSD ve R-FCN algoritmalarının performansı

Metot	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
SSD	ResNet-101	512	59	31.20%	50.40%	33.30%	10.20%	34.50%	49.80%
R-FCN	ResNet-101	512	12	31.5%	53.20%	-	14.30%	35.50%	44.20%

R-FCN ve SSD algoritmaları karşılaştırıldığında; R-FCN algoritmalarının doğruluk oranının bir miktar fazla olmasına rağmen FPS değerleri arasındaki fark azımsanamayacak derecededir.

RetinaNet algoritması COCO veri seti üzerinde canlı nesne tespitiyle ilgili FPS, backbone ve mAP değerleri aşağıdaki tabloda verilmiştir:

Tablo 4.6. RetinaNet algoritmasının performansı

Metot	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
RetinaNet	ResNet-50	500	13.9	32.50%	50.90%	34.80%	13.90%	35.80%	46.70%
	ResNet-101	500	11.1	34.40%	53.10%	36.80%	14.70%	38.50%	49.10%
	ResNet-50	800	6.5	35.70%	55.00%	38.50%	18.90%	38.90%	46.30%
	ResNet-101	800	5.1	37.80%	57.50%	40.80%	20.20%	41.10%	49.20%

RetinaNet algoritmasında farklı backbone'lara sahip ve farklı boyutlardaki resimler denenmiştir. ResNet-50, FPS değeri olarak ResNet-101'den başarılıken doğruluk değeri için tam tersi geçerlidir. Aynı şekilde resim boyutu arttıkça doğruluğun arttığını ancak FPS değerinin yarı yarıya düşüğünü gözlemlemekteyiz.

EfficientDet algoritması COCO veri seti üzerinde canlı nesne tespitiyle ilgili FPS, backbone ve mAP değerleri aşağıdaki tabloda verilmiştir:

Tablo 4.7. EfficientDet algoritmasının performansı

Metot	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
EfficientDet	Efficient-B0	512	98	33.80%	52.20%	35.80%	12.00%	38.30%	51.20%
	Efficient-B1	640	74.1	39.60%	58.60%	42.30%	17.90%	44.30%	56.00%
	Efficient-B2	768	56.5	43.00%	62.30%	46.20%	22.50%	47.00%	58.40%
	Efficient-B3	896	34.5	-	65.00%	49.30%	26.60%	49.40%	59.80%

YOLO sürümlerinin ve diğer nesne tespit modellerinin gerçek zamanlı görüntüler üzerinden nesne tespiti yapmasıyla ilgili COCO veri seti üzerinde FPS, kullanılan arka yapı ve mAP değerleri aşağıdaki tabloda verilmiştir:

Tablo 4.8. YOLO algoritmalarının performansı

Metot	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
YOLOv3	Darknet-53	320	45	28.20%	51.50%	29.70%	11.90%	30.60%	43.40 %
	Darknet-53	416	35	31.00%	55.30%	32.30%	15.20%	33.20%	42.80 %
	Darknet-53	608	20	33.00%	57.90%	34.40%	18.30%	35.40%	41.90 %
YOLOv4	CSPDarknet-53	416	38	41.20%	62.80%	44.30%	20.40%	44.40%	56.00 %
	CSPDarknet-53	512	31	43.00%	64.90%	46.50%	24.30%	46.10%	55.20 %
	CSPDarknet-53	608	23	43.50%	65.70%	47.30%	26.70%	46.70%	53.30 %
	Pytorch	416	65	40.40%	61.50%	43.60%	19.50%	43.80%	55.20 %
	TensorRT FP32	416	103	41.20%	62.50%	44.50%	20.00%	44.60%	56.40 %
	TensorRT FP16	416	162	41.20%	62.50%	44.50%	20.00%	44.60%	56.30 %
PP-YOLO	ResNet-50	320	132.2	39.30%	59.30%	42.70%	16.70%	41.40%	57.80 %
	ResNet-50	416	109.6	42.50%	62.80%	46.50%	21.20%	45.20%	58.20 %
	ResNet-50	512	89.9	44.40%	64.60%	48.80%	24.40%	47.10%	58.20 %
	ResNet-50	608	72.9	45.20%	65.20%	49.90%	26.30%	47.80%	57.20 %

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada; nesne tespit işleminin gerçekleştirilmesi, algoritmaların detayları ve farklılıklarıyla birlikte YOLO algoritmasının farklı sürümleri karşılaştırılmıştır.

Evrişimli sinir ağlarının (CNN) parametre paylaşımı ve boyutsallık azaltma özelliklerinden dolayı ileri beslemeli ağlara göre daha iyi bir sonuç verdiği görülmüştür. CNN'deki parametre paylaşımı nedeniyle, parametre sayısı azaltılır ve dolayısıyla hesaplamalar azaltılmış olur. Bu sebeplerden ötürü de görüntü verisetlerinin sınıflandırılması konusunda sıkılıkla kullanılmaktadır.

Evrişimli sinir ağı modellerine baktığımızda ise tek aşamalı dedektör ve iki aşamalı dedektör algoritmaları karşımıza çıkmaktadır. Tek aşamalı dedektör, sinir ağından yalnızca tek bir geçiş gerektirir ve tek seferde tüm sınırlayıcı kutuları tahmin eder. İki aşamalı dedektör ise bölge önerileri (potansiyel olarak nesne içeren alanları) oluşturur ve ardından bu bölgelerin her biri için ayrı bir tahmin yapar. İki aşamalı dedektörler iyi sonuç verse de birçok aşama gerektirdiğinden dolayı yavaş çalışmaktadır. Anlık görüntülerde nesne tespitinin düzgün yapılabilmesi için başarı oranından taviz verilerek hızın ön plana çıkarılması gerektiğinden dolayı tek aşamalı dedektörler kullanılmaktadır.

Popüler nesne tespit algoritmalarının başında R-CNN, R-FCN, SSD ve YOLO gelmektedir. R-CNN, evrişimli sinir ağları ile bölge önerilerinin kombinasyonunu oluşturur ve az miktarda açıklamalı algılama verisiyle yüksek kapasiteli bir modelin eğitilmesine yardımcı olur. R-FCN, hesaplamaları tüm görüntüde paylaştırarak tamamen evrişimli hale getirir. SSD, çeşitli boyutlardaki nesneleri doğal olarak işlemek için birden çok özellik haritasının tahminleri birleştirir. YOLO, görüntünün ilgi noktalarını seçmek yerine tek seferde çalıştırıldığında tüm nesnelerin sınıflarını ve sınırlayıcı kutularını tahmin eder. Bu algoritmaların hızı ve doğruluğu arasındaki

alışveriş göz önüne alındığında gerçek zamanlı görüntüler için YOLO'nun en iyi seçenek olduğu sonucuna varılmaktadır.

YOLO bir çok alt sürüme sahiptir. V3 sürümü, eğitim yapılmadan model boyutunu değiştirerek hız ve doğruluk arasında kolayca değişim tokuş yapılabilmesi özelliğini getirir. V4 sürümü, son teknoloji ürünü BoF ve birkaç BoS'nin güçlerini ekler ve nesne tespitinin doğruluğunu önemli ölçüde iyileştirir. V5 sürümü, algoritmanın PyTorch kütüphanesine entegre edilmiş ve CSP backbone ve PA-NET neck barındıran halidir. PP-YOLO, platformun boyutunu azaltmaya yardımcı olan ve yüksek performanslı dağıtımını mümkün kıلان veri artırma yöntemler sağlar.

Tablo 5.1. Nesne tespit algoritmalarının performanslarının karşılaştırılması

Metot	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
SSD	ResNet-101	512	59	31.20%	50.40%	33.30%	10.20 %	34.50 %	49.80 %
R-FCN	ResNet-101	512	12	31.50%	53.20%	-	14.30 %	35.50 %	44.20 %
RetinaNet	ResNet-50	500	13.9	32.50%	50.90%	34.80%	13.90 %	35.80 %	46.70 %
	ResNet-101	500	11.1	34.40%	53.10%	36.80%	14.70 %	38.50 %	49.10 %
	ResNet-50	800	6.5	35.70%	55.00%	38.50%	18.90 %	38.90 %	46.30 %
	ResNet-101	800	5.1	37.80%	57.50%	40.80%	20.20 %	41.10 %	49.20 %
EfficientDet	Efficient-B0	512	98	33.80%	52.20%	35.80%	12.00 %	38.30 %	51.20 %
	Efficient-B1	640	74.1	39.60%	58.60%	42.30%	17.90 %	44.30 %	56.00 %
	Efficient-B2	768	56.5	43.00%	62.30%	46.20%	22.50 %	47.00 %	58.40 %
	Efficient-B3	896	34.5	-	65.00%	49.30%	26.60 %	49.40 %	59.80 %
YOLOv3	Darknet-53	320	45	28.20%	51.50%	29.70%	11.90 %	30.60 %	43.40 %
	Darknet-53	416	35	31.00%	55.30%	32.30%	15.20 %	33.20 %	42.80 %
	Darknet-53	608	20	33.00%	57.90%	34.40%	18.30 %	35.40 %	41.90 %
YOLOv4	CSPDarknet-53	416	38	41.20%	62.80%	44.30%	20.40 %	44.40 %	56.00 %
	CSPDarknet-53	512	31	43.00%	64.90%	46.50%	24.30 %	46.10 %	55.20 %

	CSPDarknet-5 3	608	23	43.50%	65.70%	47.30%	26.70 %	46.70 %	53.30 %
	Pytorch	416	65	40.40%	61.50%	43.60%	19.50 %	43.80 %	55.20 %
	TensorRT FP32	416	103	41.20%	62.50%	44.50%	20.00 %	44.60 %	56.40 %
	TensorRT FP16	416	162	41.20%	62.50%	44.50%	20.00 %	44.60 %	56.30 %
PP-YOLO	ResNet-50	320	132. 2	39.30%	59.30%	42.70%	16.70 %	41.40 %	57.80 %
	ResNet-50	416	109. 6	42.50%	62.80%	46.50%	21.20 %	45.20 %	58.20 %
	ResNet-50	512	89.9	44.40%	64.60%	48.80%	24.40 %	47.10 %	58.20 %
	ResNet-50	608	72.9	45.20%	65.20%	49.90%	26.30 %	47.80 %	57.20 %

AP: birincil değerlendirme metriği, AP_S: küçük nesneler (alan < 32²) için AP,
AP_M: orta nesneler (32² < alan < 96²) için AP , AP_L: orta nesneler (alan > 96²) için AP

İlgili tabloya göz atıldığında, öteki nesne tespit algoritmalarına kıyasla ResNet-50 backbone ve 608x608 piksel boyutlu resimleri kullanan, 72.9 FPS değeri ve 45.20% mAp değeri elde edilerek diğerlerine kıyasla daha başarılı olan PP-YOLO'nun en iyi model olduğuna karar verilmiştir.

Karşılaştırma öncesinde modellerin, NVIDIA Jetson TX2 platformu kullanılarak insansız hava aracı veri seti ile eğitilmesi ve değerlendirilmesinin yapılması planlanmıştır. Ancak platform tabanlı sıkıntılardan ötürü değerlendirmenin yapılacağı mAp değeri elde edilememiş, bazı modellerin çalıştırılmasında sıkıntılar yaşanmıştır. Bu sebeple daha önce yapılan çalışmalarдан elde edilen sonuçlar ele alınarak değerlendirmeler gerçekleştirılmıştır.

KAYNAKLAR

- [1] <https://towardsdatascience.com/object-detection-simplified-e07aa3830954> (Ziyaret tarihi: 12 Nisan 2021).
- [2] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11), 3212-3232.
- [3] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2), 261-318.
- [4] <https://www.mathworks.com/discovery/object-detection.html> (Ziyaret tarihi: 29 Nisan 2021).
- [5] Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.
- [6] Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. *Procedia computer science*, 132, 1706-1717.
- [7] https://www.researchgate.net/publication/338253407_A_STUDY_ON_OBJECT_DETECTION (Ziyaret tarihi: 24 Nisan 2021).
- [8] <https://towardsdatascience.com/understanding-object-detection-9ba089154df8> (Ziyaret tarihi: 29 Nisan 2021).
- [9] <https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html> (Ziyaret tarihi: 18 Nisan 2021).
- [10] <https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/> (Ziyaret tarihi: 20 Nisan 2021).
- [11] <https://www.fritz.ai/objectdetection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them> (Ziyaret tarihi: 2 Mayıs 2021).
- [12] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.
- [13] Lienhart, Rainer, and Jochen Maydt. "An extended set of haar-like features for rapid object detection." Proceedings. international conference on image processing. Vol. 1. IEEE, 2002.

- [14] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 886–893.
- [15] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8
- [16]<https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/> (Ziyaret tarihi: 8 Mayıs 2021).
- [17] <https://www.dataversity.net/brief-history-deep-learning/> (Ziyaret tarihi: 12 Nisan 2021).
- [18] <https://www.jeremyjordan.me/object-detection-one-stage/> (Ziyaret tarihi: 2 Mayıs 2021).
- [19] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in Advances in neural information processing systems, 2016, pp. 379–387
- [20] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, “Light-head r-cnn: In defense of two-stage object detector,” arXiv preprint arXiv:1711.07264, 2017.
- [21] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection.” in CVPR, vol. 1, no. 2, 2017, p. 4.
- [22] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
- [23] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
- [24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37.
- [25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” IEEE transactions on pattern analysis and machine intelligence, 2018.
- [26] Day R.A., Bilimsel Bir Makale Nasıl Yazılır ve Yayımlanır, Çeviri: Gülay Aşkar Altay, Tubitak, <http://journals.tubitak.gov.tr/kitap/maknasyaz/> (Ziyaret tarihi: 10 Nisan 2012).

- [27] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [28] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
- [29] Long, Xiang, et al. "PP-YOLO: An effective and efficient implementation of object detector." arXiv preprint arXiv:2007.12099 (2020).
- [30] <https://appsilon.com/object-detection-yolo-algorithm/> (Ziyaret tarihi: 6 Mayıs 2021).
- [31] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.
- [32] <https://pjreddie.com/darknet/yolo/> (Ziyaret tarihi: 6 Mayıs 2021).
- [33] <https://github.com/ultralytics/yolov3> (Ziyaret tarihi: 6 Mayıs 2021).
- [34] <https://github.com/Tianxiaomo/pytorch-YOLOv4> (Ziyaret tarihi: 6 Mayıs 2021).
- [35] <https://github.com/ultralytics/yolov5> (Ziyaret tarihi: 6 Mayıs 2021).
- [36] Albaba, B. M., & Ozer, S. (2020). SyNet: An Ensemble Network for Object Detection in UAV Images. arXiv preprint arXiv:2012.12991.
- [37] Zhang, S. (2005, March). Object tracking in unmanned aerial vehicle (uav) videos using a combined approach. In Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. (Vol. 2, pp. ii-681). IEEE.
- [38] Teutsch, M., & Krüger, W. (2012, September). Detection, segmentation, and tracking of moving objects in UAV videos. In 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance (pp. 313-318). IEEE.
- [39] Coşkun, M., & Ünal, S. (2016). Implementation of tracking of a moving object based on camshift approach with a UAV. Procedia Technology, 22, 556-561.
- [40] Ibrahim, A. W. N., Ching, P. W., Seet, G. G., Lau, W. M., & Czajewski, W. (2010, November). Moving objects detection and tracking framework for UAV-based surveillance. In 2010 Fourth Pacific-Rim Symposium on Image and Video Technology (pp. 456-461). IEEE.

- [41] Abughalieh, K. M., Sababha, B. H., & Rawashdeh, N. A. (2019). A video-based object detection and tracking system for weight sensitive UAVs. *Multimedia Tools and Applications*, 78(7), 9149-9167.
- [42] Cazzato, D., Cimarelli, C., Sanchez-Lopez, J. L., Voos, H., & Leo, M. (2020). A Survey of Computer Vision Methods for 2D Object Detection from Unmanned Aerial Vehicles. *Journal of Imaging*, 6(8), 78.
- [43] Merkulova, I. Y., Shavetov, S. V., Borisov, O. I., & Gromov, V. S. (2019). Object detection and tracking basics: Student education. *IFAC-PapersOnLine*, 52(9), 79-84.
- [44] 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.
- [45] Kim, S., and Casper, R. (2013). Applications of Convolution in Image Processing with MATLAB. University of Washington
- [46] Jain, R., Kasturi, R., and Schunck, B.G. (1995). Machine vision.
- [47] El-Sayed, M.A. (2012). Edges Detection of Images: Algorithms of Edges Detection for Digital Image. LAP LAMBERT Academic Publishing.
- [48] Viola, P., Jones, M. J., and Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *Proceedings Ninth IEEE International Conference on Computer Vision*.
- [49] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2), 154–171.
- [50] Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. (2016). Monocular 3D Object Detection for Autonomous Driving. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2147–2156.
- [51] Ma, C., Huang, J.-B., Yang, X., and Yang, M.-H. (2015). Hierarchical Convolutional Features for Visual Tracking. *2015 IEEE International Conference on Computer Vision (ICCV)*, 3074–30.
- [52] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, Qi Tian, "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking", European Conference on Computer Vision (ECCV), 2018.
- [53] Jalled, F., & Voronkov, I. (2016). Object detection using image processing. arXiv preprint arXiv:1611.07791.

- [54] Xu, X., Zhang, X., Yu, B., Hu, X. S., Rowen, C., Hu, J., & Shi, Y. (2019). Dacsdc low power object detection challenge for uav applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [55] Tijtgat, N., Van Ranst, W., Goedeme, T., Volckaert, B., & De Turck, F. (2017). Embedded real-time object detection for a UAV warning system. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 2110-2118).
- [56] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).
- [57] Mittal, P., Sharma, A., & Singh, R. (2020). Deep learning-based object detection in low-altitude UAV datasets: A survey. *Image and Vision Computing*, 104046.
- [58] Kamate, S., & Yilmazer, N. (2015). Application of object detection and tracking techniques for unmanned aerial vehicles. *Procedia Computer Science*, 61, 436-441.
- [59] Bahadır, Ş. İ. N., & Kadioğlu, İ. (2019). İnsansız Hava Aracı (İHA) ve Görüntü İşleme Teknikleri Kullanılarak Yabancı Ot Tespitinin Yapılması. *Türkiye Herboloji Dergisi*, 22(2), 211-217.
- [60] Ekmen, M. İ., & Aydoğdu, Ö. İnsansız Hava Araçları İçin Görüntü İşleme Tabanlı Otonom İniş. *Avrupa Bilim ve Teknoloji Dergisi*, 297-303.
- [61] Batuhan, A. Y., & NAMLI, E. GÖRÜNTÜ İŞLEME TABANLI İHA VE UYDU SİSTEMLERİ HİBRİT YAPAY ZEKÂ MODELİYLE KAÇAK YAPILARIN TESPİTİ. *Yönetim Bilişim Sistemleri Dergisi*, 3(2), 114-129.
- [62] Zeybek, M., & Şanlıoğlu, İ. (2019). Topografik Yüzey Değişimlerinin Görüntü İşleme Teknikleriyle Belirlenmesi Üzerine Bir Araştırma. *Doğal Afetler ve Çevre Dergisi*, 5(2), 350-367.
- [63] KABADAYI, A., & UYSAL, M. (2020). Building Detection From High Resolution UAV Data, 2(2), 43-48.
- [64] Niethammer, U., Rothmund, S., Schwaderer, U., Zeman, J., & Joswig, M. (2011). Open source image-processing tools for low-cost UAV-based landslide investigations. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(1), C22.
- [65] YAMAN, K., SARUCAN, A., Mehmet, A. T. A. K., & AKTÜRK, N. (2001). Dinamik çizelgeleme için görüntü işleme ve arıma modelleri yardımıyla veri hazırlama. *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, 16(1).
- [66] Zhu, P., Wen, L., Du, D., Bian, X., Hu, Q., & Ling, H. (2020). Vision Meets Drones: Past, Present and Future. *arXiv preprint arXiv:2001.06303*.

- [67] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, Qi Tian, " The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking", European Conference on Computer Vision (ECCV), 2018.
- [68] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).
- [69] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788). 779-788)

ÖZGEÇMİŞ

Cengizhan PARLAK

1998 yılında Edirne'de doğdu. Ortaokul ve lise öğrenimini Keşan'da tamamladı. 2017 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı.

Enes BAŞPINAR

1999 yılında Bursa'da doğdu. İlk, orta ve lise öğrenimini doğduğu şehirde tamamladı. 2017 yılında Kocaeli Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne başladı.

Mehmet Ertuğrul EVRENSEL

1999 yılında Erzincan'da doğdu. İlk, orta ve lise öğrenimini Erzincan'da tamamladı. 2017 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'ne kaydoldu.