



**ARDIŞIK SAYISAL GÖRÜNTÜLERDE ÇOK SAYIDA HAREKETLİ
NESNENİN TESPİTİ, İZLENMESİ, TANIMLANMASI VE
SINIFLANDIRILMASI (NTİTS)**

Kemal BOZKURT

**DOKTORA TEZİ
ELEKTRİK EĞİTİMİ ANA BİLİM DALI**

**GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

HAZİRAN 2021

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmada;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmada yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Kemal BOZKURT
29/06/2021

ARDIŞIK SAYISAL GÖRÜNTÜLERDE ÇOK SAYIDA HAREKETLİ NESNENİN TESPİTİ, İZLENMESİ, TANIMLANMASI VE SINIFLANDIRILMASI (NTİTS)

(Doktora Tezi)

Kemal BOZKURT

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2021

ÖZET

Gelişen teknolojiye bağlı olarak çok yüksek çözünürlüklü sayısal kameralar ucuzlamış, erişimi kolaylaşmış ve kullanım alanları her geçen gün yaygınlaşmaya başlamıştır. Ancak çok yüksek çözünürlüklü ardışık sayısal görüntüler çok büyük miktarda veri içerdiğinden sayısal ortamda işlenip faydalı bilginin ortaya çıkartılması yüksek işlemci hızına ve büyük hesaplama kapasitesine ihtiyaç doğurmaktadır. Yüksek işlemci hızı ve büyük hesaplama kapasitesi sağlansa bile yüksek çözünürlüklü ardışık sayısal görüntülerin gerçek zamanlı olarak özelliklerinin tespit edilmesi, faydalı bilginin çıkartılması aynı zaman diliminde değerlendirilmesi, kullanılan algoritmalarla göre çoğunlukla mümkün olmamaktadır. Ardışık sayısal görüntülerden çok sayıda hareketli nesnenin gerçek zamanlı tespiti, izlenmesi, tanımlanması ve sınıflandırılması ise paralel işlem yapabilen çok farklı işlemciler ve onlara uygun algoritmalar gerektirmektedir. Bu tez çalışmasında, Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması (NTİTS) için biyomimetik yaklaşımı temel alan iki aşamalı bir yöntem geliştirilmiştir. Yöntemin birinci aşamasında ilgi alanı olan hareketli nesnelerin bulunduğu alanlar tespit edilerek hareketsiz nesnelerden ve arka zeminden ayrıştırılmaktadır. Hareketli alanlar tespit edilirken İnsan, maymun ve bazı başka gelişmiş canlıların gözünün çalışma prensibinden esinlenilmiştir. Bu canlıların gözleri görüş alandaki her noktayı aynı çözünürlükte görmez. İlgi alanı olan nesneleri çok yüksek çözünürlüklü görürken ilgi alanından uzaklaştıkça netlik azalır, objeler flulaşır. Böylece ilgi alanı dışındaki alanların ayrıntılarıyla beyin meşgul edilmez. Doğadaki bu özellik kullanılarak çok yüksek çözünürlüklü görüntülerin tüm piksellerini işlemeye kaynak ayırmaktansa sadece hareketli nesneleri oluşturan alanların analizi yapılmaktadır. Her türlü canlı, cansız, endüktif, kapasitif nesneyi zeminden, arka plandan ve hareketsiz olan diğer nesnelerden ayırtırmak için optik akış (OA) yöntemi uygulanmaktadır. Yöntemin ikinci aşamasında, hareketli nesnelere ait özellikler çıkartılarak tanımlanma, izleme ve sınıflandırma işlemi yapılmaktadır. Nesne tanımlama ve sınıflandırma işlemlerinde Derin Sinir Ağları (DSA) kullanılmaktadır. Kodlar, Grafik İşlemci Birimleri (GPU)'nde, Merkezi İşlemci Birimleri (CPU)'ne kıyasla çok daha hızlı işlenebildiği için geliştirilen yöntemde GPU'lar kullanılmaktadır. NTİTS yönteminden elde edilen sonuçlar, çok yüksek çözünürlüklü ardışık sayısal görüntülerde çok sayıda hareketli nesnelerin tespiti, izlenmesi tanımlanması ve sınıflandırılması işleminin yüksek doğruluk oranı ile gerçekleştirilebildiğini göstermiştir.

Bilim Kodu	: 91440
Anahtar Kelimeler	: Optik akış, yapay zeka, derin sinir ağları, derin öğrenme, hareket tespiti
Sayfa Adedi	: 153
Danışman	: Prof. Dr. Çetin ELMAS
2. Danışman	: Prof. Dr. Uğur GÜVENÇ

DETECTION, TRACKING, IDENTIFICATION AND CLASSIFICATION OF MULTIPLE MOVING OBJECTS IN SEQUENTIAL DIGITAL IMAGES (NTITS)

(Ph. D. Thesis)

Kemal BOZKURT

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2021

ABSTRACT

Due to the developing technology, very high-resolution digital cameras have become more affordable, easier to access, and the areas of their utilization have become widespread day by day. As sequential digital images with very high resolution contain a vast amount of data, however, processing and revealing useful information in a digital environment creates a need for high processor speed and large computing capacity. Even when high processor speed and large computing capacity get to be provided; depending on the algorithms that have been employed, it is often not possible to detect the properties of high-resolution sequential digital images in real-time, extract useful information, and assess them at the same time period. Real-time detection, tracking, identification, and classification of multiple moving objects in sequential digital images require quite special processors with parallel processing capability and proper algorithms for them. In this thesis, a two-step method based on the biomimetic approach was developed for the Detection, Tracking, Identification, and Classification of Multiple Moving Objects in Sequential Digital Images (NTITS). The areas where the focused moving objects are, get detected and separated from the static objects and the background, at the first stage of the method. It was inspired by the principle of how the eyes of humans, monkeys, and some other advanced creatures work when detecting moving areas. The eyes of these creatures do not see every point at sight at the same resolution. While the focused objects are seen with very high resolutions, the sharpness decreases, and the objects become blurred as one moves away from the focused area. In this way, the brain remains not engaged with the details of other areas outside of the focused area. Instead of allocating resources for processing all the pixels in very high-resolution images, only the areas with moving objects are analyzed with the help of this feature in nature. The optical flow (OF) method is applied to separate all variations of animate, inanimate, inductive, capacitive objects from the surface, background, and other static objects. At the second stage of the method, the properties of moving objects are extracted, and the identification, tracking, and classification process take place. Deep Neural Networks (DNN) are employed in object identification and classification processes. As the codes can be processed eminently faster on Graphics Processing Units (GPUs) compared to Central Processing Units (CPUs), GPUs are preferred in the developed method. The results from the NTITS method demonstrated that the detection, tracking, and classification of multiple moving objects in very high resolution sequential digital images can be completed with a high accuracy rate.

Science Code : 91440
Key Words : Optic flow, artificial intelligence, deep neural network, deep learning, motion detection
Page Number : 153
Supervisor : Prof. Dr. Çetin ELMAS
Co Supervisor : Prof. Dr. Uğur GÜVENÇ

TEŞEKKÜR

Çalışmalarım boyunca yardımlarını esirgemeyen, çalışmalarım da daima beni yönlendiren ve tecrübelerini paylaşan, ufkumu açan ve her zaman destek olan danışman hocam Prof. Dr. Çetin ELMAS'a ve Prof. Dr. Uğur GÜVENÇ'e yine kıymetli tecrübelerinden faydalandığım tez izleme komitesi hocalarım Prof. Dr. Recep DEMİRCİ ve Prof. Dr. Mustafa ALKAN'a çalışmam boyunca her anlamda maddi ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan Annem'e ve Babam'a teşekkürü bir borç bilirim.



İÇİNDEKİLER

	Sayfa
ÖZET	iv
ABSTRACT	v
TEŞEKKÜR.....	vi
İÇİNDEKİLER	vii
ÇİZELGELERİN LİSTESİ.....	xi
ŞEKİLLERİN LİSTESİ	xii
RESİMLERİN LİSTESİ	xiv
SİMGELER VE KISALTMALAR.....	xvi
1. GİRİŞ.....	1
2. GÖRÜNTÜ İŞLEME VE HAREKETLİ NESNE TESPİTİ	5
2.1. Görüntü İşleme İle İlgili Temel Kavramlar	5
2.1.1. Sayısal görüntü işleme	12
2.1.2. Görüntü işleme ile çözünürlük arasındaki ilişki.....	13
2.1.3. Gerçek zamanlı görüntü işlemede karşılaşılan sorunlar.....	13
2.1.4. Görüntü işlemenin kullanıldığı alanlar	14
2.2. Görüntü İşleme Yöntemleri.....	15
2.3. Hareketli Nesnelerin Tespiti	16
2.3.1. Optik akış (Optic flow)	18
2.3.2. İki boyutlu hareket kısıtlama denklemi.....	21
2.3.3. Açıklık ya da belirsizlik problemi (Aperture problem).....	24
2.4. Fark Teknikleri.....	28
2.4.1. Akış lablasyenleri hesabı.....	30
2.4.2. Hataları minimize etme	31

	Sayfa
2.4.3. Yinelemeli çözüm (Iterative solution)	33
3. DERİN ÖĞRENME	35
3.1. Evrişimli Sinir Ağları (Convolutional Neural Network)	43
3.1.1. Evrişim katmanı (Convolution layer).....	44
3.1.2. Etkinleştirme katmanı (Activation layer).....	49
3.1.3. Havuzlama katmanı (Pooling layer).....	53
3.1.4. Düzleştirme katmanı (yapay sinir ağları için giriş katmanı).....	54
3.1.5. Sinir ağı katmanı (SAK).....	55
3.1.6. Seyreltme katmanı (Dropout layer).....	57
3.1.7. Sınıflandırma katmanı.....	58
3.1.8. Benzerlik katsayısı	59
3.2. ESA Mimarileri.....	63
3.2.1. LeNet mimarisi.....	64
3.2.2. AlexNet	64
3.2.3. ZFNet mimarisi	65
3.2.4. VGGNet mimarisi	66
3.2.5. GoogleLeNet mimarisi.....	67
3.2.6. ResNet mimarisi.....	69
3.3. Derin Ağlarda Kullanılan Programlama Dilleri	70
3.3.1. Python programlama dili.....	70
3.3.2. C++ programlama dili.....	71
3.3.3. Java programlama dili	71
3.3.4. Scala programlama dili	72
3.3.5. R programlama dili	72

	Sayfa
3.3.6. Lua programlama dili.....	72
3.4. Derin Ağlarda Kullanılan Derin Öğrenme Kütüphaneleri	73
3.4.1. Caffé kütüphanesi.....	73
3.4.2. Keras kütüphanesi	74
3.4.3. Tensorflow kütüphanesi.....	74
3.4.4. Theano kütüphanesi	74
3.4.5. Torch kütüphanesi.....	75
3.4.6. Digits kütüphanesi.....	75
3.4.7. Deeplearning4j kütüphanesi.....	76
4. ARDIŞIK SAYISAL GÖRÜNTÜLERDE BULUNAN ÇOK SAYIDA HAREKETLİ NESNENİN TESPİTİ, İZLENMESİ, TANIMLANMASI VE SINIFLANDIRILMASI	79
4.1. Doğayı Taklit, Canlıyı Taklit (Biomimicry, Biomimetic)	82
4.2. Tek Atış Çoklu Kutu Algılama (Single Shot Multibox Detector) (TAÇKA)	96
4.3. NİTİTS Yönteminin Yazılımının Geliştirilmesi	107
4.3.1. Ubuntu 16.04 işletim sistemi.....	108
4.3.2. Kullanılan programlama dili Python.....	108
4.3.3. OpenCV kütüphanesi	109
4.3.4. NumPY kütüphanesi	109
4.3.5. Matplotlib kütüphanesi.....	110
4.4. Geliştirilen NTİTS Yönteminin Çalıştırılmasında Kullanılan Donanım	110
4.4.1. Jetson TX2 geliştirme kiti	110
4.4.2. Kamera	113
4.4.3. Klavye	114
4.4.4. Fare.....	114

	Sayfa
4.4.5. Ekran	115
5. GELİŞTİRİLEN NTİTS YÖNTEMİNİN ÇALIŞTIRILMASINDAN ELDE EDİLEN SONUÇLARIN İRDELENMESİ	117
6. SONUÇ VE ÖNERİLER	137
6.1. Sonuçlar.....	137
6.2. Gelecekte Yapılabilecek Uygulamalara Dair Öneriler.....	137
KAYNAKLAR	141
ÖZGEÇMİŞ	153

ÇİZELGELERİN LİSTESİ

Çizelge	Sayfa
Çizelge 2.1. Sayısal görüntü çözünürlükleri	9
Çizelge 2.2. Lablasyenlerin hesaplanmasında kullanılan maske	31
Çizelge 3.1. 6x6 boyutlarındaki matris ile 4x4 boyutlarındaki matrisin evrişim sonucu	47
Çizelge 3.2. Sıfır dolgu işlemi	49
Çizelge 3.3. VGGNet-ConvNet mimarisini oluşturan katmanlar	67
Çizelge 3.4. 34 katmanlı ResNet mimarisi	70
Çizelge 3.5. DÖ kütüphanelerinin özelliklerinin birbirleri ile karşılaştırılması	77
Çizelge 4.1. Pascal Voc 2007 veri seti kullanılarak elde edilen sonuçlar.....	106
Çizelge 4.2. JetsonTM TX2 Kitinin teknik özellikleri	113

ŞEKİLLERİN LİSTESİ

Şekil	Sayfa
Şekil 2.1. Analog ve sayısal sinyaller	5
Şekil 2.2. Sayısal görüntünün elde edilmesi	7
Şekil 2.3. PPI ile inch arasındaki ilişki	11
Şekil 2.4. Ardışık üç tane sayısal görüntü karesinde insan kafası silueti görülmektedir. a) ve b)'de birinci ve ikinci görüntü kareleri ve bu iki görüntü karesi arasındaki hareketin yönü ve şiddeti d)'de görülmektedir. b) ve c)'de ikinci ve üçüncü görüntü kareleri ve bu iki görüntü kareleri arasındaki hareketin yönü ve şiddeti e)'de görülmektedir	20
Şekil 2.5. Bir konumdan, diğer konuma hareket eden noktanın zamana bağlı olarak şiddet değişmezliği kabulü ile hız bileşenlerinin gösterilmesi	22
Şekil 2.6. a) hareketli çubuğun ortası delik kartonda hareketi, b) parlaklık değişiminin kısıt çizgisi	24
Şekil 2.7. a) dönen ışıklı direk berber işareti, b) berber işareti hareket yönü, c) hareket alanı d) optik akış alanı	25
Şekil 2.8. a) Sabit ışık kaynağı altında, kendi eksenini etrafında hareket eden küre b) hareketli ışık kaynağı altında, sabit duran küre	28
Şekil 2.9. x, y ve t yönlerindeki şiddet türevlerinin hesaplanmasında kullanılan küp....	29
Şekil 2.10. Hatayı minimize eden akış hızı değeri, bileşen doğrusuna dik olarak akış hızının lokal ortalamasından çizilen doğru	33
Şekil 3.1. Yapay zeka, makine öğrenmesi ve derin öğrenme ilişkileri.....	37
Şekil 3.2. Yapay zekanın gelişim sürecindeki kilometre taşları	40
Şekil 3.3. Derin öğrenmenin başarısı ve veri miktarı arasındaki ilişki.....	41
Şekil 3.4. a) giriş görüntüsü matrisi, b) evrişim operatörü ve c) filtre.....	45
Şekil 3.5. Sayısal görüntüye filtre uygulanması	45
Şekil 3.5. (devam) Sayısal görüntüye filtre uygulanması	46
Şekil 3.5. (devam) Sayısal görüntüye filtre uygulanması	47
Şekil 3.6. Yapay sinir ağlarında sıklıkla kullanılan etkinleştirme fonksiyonları	53
Şekil 3.7. Enbüyüğü havuzlama ve ortalamayı havuzlama işlemleri	54
Şekil 3.8. Tam bağımlı katmana giriş için düzleştirilen verilerin elde edilmesi.....	54

Şekil	Sayfa
Şekil 3.9. ESA’larda SAK’ta tam bağlı katmanın sinirlere bağlantısı.....	56
Şekil 3.10. ESA’larda SAK’ta kısmi bağlı katmanın sinirlere bağlantısı.....	57
Şekil 3.11. Seyreltilmiş sinir ağı modeli a) iki gizli katmanlı standart sinir ağı yapısı b) seyreltilmiş sinir ağı yapısı bazı sinirlerin bağlantıları ihmal edilmiş.....	58
Şekil 3.12. ESA’ların yapısında bulunan katmanların gösterimi.....	59
Şekil 3.13. Dice benzerlik katsayısının şekillerle matematiksel olarak ifadesi	62
Şekil 3.14. Jaccard benzerlik katsayısının şekillerle matematiksel olarak ifadesi.....	63
Şekil 3.15. LeNet mimarisini oluşturan katmanlar	64
Şekil 3.16. AlexNet mimarisini oluşturan katmanlar.....	65
Şekil 3.17. ZFNet mimarisinin yapısı	66
Şekil 3.18. Tüm katmanları ile birlikte GoogLeNet mimarisi	68
Şekil 3.19. ResNet mimarisini oluşturan 34 katmanlı yapı	69
Şekil 4.1. İnsan gözünün yatay eksenindeki görüş açılarına göre görüş netlikleri	87
Şekil 4.2. İnsan gözünün dikey eksenindeki görüş açılarına göre görüş netlikleri	89
Şekil 4.3. Geliştirilen özgün NTİTS yönteminin blok şeması	91
Şekil 4.4. Ardışık sayısal görüntülerdeki hareketli nesneler.....	92
Şekil 4.5. NTİTS yöntemini işlem akış diyagramı	95
Şekil 4.6. TAÇKA mimarisini oluşturan yapılar	99
Şekil 4.7. a) nesne tespit kutuları ile giriş görüntüsü b) özellik kutu haritası c) özellik kutu haritası.....	101
Şekil 4.8. TAÇKA’nın farklı büyüklüklerdeki nesnelerin VOC2007 test seti üzerindeki elde edilen sınırlayıcı kutuların ile nesnelerin en boy oranları sonuçları	105
Şekil 4.9. TAÇKA’nın VOC2007 veri seti üzerinde hız ve doğruluk oranları bakımından karşılaştırılması görülmektedir.....	107

RESİMLERİN LİSTESİ

Resim	Sayfa
Resim 2.1. Çingene kızı mozaïği	8
Resim 2.2. Katı olmayan doku hareketi.....	26
Resim 2.3. Ekran sabir olmasına rağmen hareketi gösterir	27
Resim 3.1. a) Normal, b) Aynalama, c) Döndürme, d) Kırpma, e) Renk değıştirme, f) Gürültü ekleme, g) Bilgi kaybı, h) Renk zıtlığı	42
Resim 3.2. Trafik işaretinin yeşil çerçeve ile gerçek sınırları, kırmızı çerçeve ile nesne tespit algoritmasından elde edilen sınırları gösterilmiştir	60
Resim 4.1. Doğadan esinlenilen tasarımlar görölmektedir.	83
Resim 4.2. Varsayılan kutuların, kesin referans kutuları ile eşleştirilmesi.....	103
Resim 4.3. Seçilen sınırlayıcı kutunun nesneler üzerinde gösterimi	104
Resim 4.4. Jetson TX2 geliştirme kitinin paket içeriğı.....	111
Resim 4.5. Jetson TX2 modülü.....	112
Resim 4.6. NTİTS yönteminde kullanılan donanımlar	116
Resim 5.1. a),b) sayısal kameradan alınan 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntü görölmektedir, c) 3840x2160 piksel 4K çözünürlüğündeki sayısal görüntünün yeniden boyutlandırılarak 300x300 piksel çözünürlüğündeki görüntü.....	119
Resim 5.2. 3840x2160 piksel 4K çözünürlüğündeki hareket eden nesnelerin OA'dan elde edilen hız vektörleri.....	120
Resim 5.3. 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün hareketli nesneler nesnelerin olmadığı durumda optik akış vektörleri	121
Resim 5.4. 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün hareket eden ve hareket etmeyen nesnelerin olduğu bölgeye yakından bakıldığında optik akış vektörleri	122
Resim 5.5. 1920x1080 çözünürlükteki sayısal görüntülerde OA algoritmasından elde edilen hareketli nesnelere ait hız vektörlerin karşılık geldiğı piksellere ait görüntü	123
Resim 5.6. Yolda yürüyen insanların OA yöntemi ile tespit edilmesi.....	124

Resim	Sayfa
Resim 5.7. 3840x2160 piksel 4K çözünürlüğündeki sayısal görüntüde hareketli nesnelerin hareketsiz nesnelerden ayrıştırılıp tespit edilip tanımlanması ve sınıflandırılması gösterilmiştir	125
Resim 5.8. Yolda yürüyen insanlar NTİTS yöntemi ile tespit ediliyor	126
Resim 5.9. NTİTS ile analizi sonucunda görüntüdeki insanın tanımlanması.....	126
Resim 5.10. Bir mağazadaki insanların NTİTS yöntemi ile tespit edilmesi.....	127
Resim 5.11. Hareketli nesnelerin hareketsiz nesnelerden ve arka plandan ayrıştırılması.....	128
Resim 5.12. Yürümekte olan kadın.....	129
Resim 5.13. Geliştirilen NTİTS yönteminin çalıştırılmasıyla otoyolda seyahat halinde giden otomobillerin tespiti tanımlanması ve sınıflandırılmasının yapılması	129
Resim 5.14. Yolun kenarında yürüyen insanlar.....	130
Resim 5.15. Vücudunun büyük bir kısmı olmayan hareketli insanın tanımlanması	131
Resim 5.16. Sadece omuz bölgesi ve baş kısmı olan hareketli insanın ve elinde tuttuğu su şişesinin tanımlanması.....	132
Resim 5.17. Ardışık sayısal görüntülerde iç içe geçmiş hareketli nesnelerin tanımlanması	133
Resim 5.18. Otoparkta park etmeye çalışan otomobil	134

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler

Açıklamalar

D	Dice katsayısı
I_x, I_y	x ve y yönündeki uzaysal kısmi türevler
I_t	Zamansal kısmi türev
J	Jaccard katsayısı
m	Metre
p	Ekleme
u, v	x ve y yönündeki hız bileşenleri
$\bar{u}$ ve $\bar{v}$	x ve y yönündeki ortalama hız bileşenleri
α	Yumuşatma değeri
Δ	Laplasien operatörü

Kısaltmalar

Açıklamalar

2B	2-Boyutlu
3B	3-Boyutlu
CPU	Merkezi işlemci birimi
DIGITS	NVIDIA Derin Öğrenme GPU Eğitim Sistemi
DL4J	Deeplearning4j
DÖ	Derin Öğrenme
DÖA	Derin Öğrenme Algoritmaları
DSA	Derin Sinir Ağları
ESA	Evrişimli Sinir Ağı
EF	Etkinleştirme Fonksiyonu
FPS	Saniyedeki kare sayısı (Frame Per Second)
Full HD	Çok Yüksek Çözünürlük (Full High Definition)

Kısaltmalar**Açıklamalar**

GPU	Grafik işlemci birimi
HD	Yüksek Çözünürlük (High Definition) Görüntü
HQ	Yüksek Kaliteli (High Quality) Görüntü
ILSVRC	Büyük Ölçekli Görsel Tanıma Yarışması (Large Scale Visual Recognition Challenge)
ImageNet	Büyük Ölçekli Görsel Tanıma Yarışmasını
JPL	Jet Propulsion Laboratuvarı
MÖ	Makine Öğrenmesi
MP	Mega Piksel
NTITS	Çok Yüksek Çözünürlüklü Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Gerçek Zamanlı Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması
OA	Optik Akış
OpenCV	Açık Kaynak Bilgisayar Görmesi (Open Source Computer Vision)
Pascal VOC	Pascal Görsel Nesne Sınıfları
PPI	Per Pixel Inch
PReLU	Parametrik Doğrultulmuş Lineer Birim (Parametric Rectified Linear Unit) fonksiyonu
ReLU	Doğrultulmuş Lineer Birim (Rectified Linear Unit) fonksiyonu
SAK	Sinir ağı katmanı
SD	Standart Çözünürlük (Standard Definition) Görüntü
TAÇKA	Tek Atış Çoklu Kutu Algılama (Single Shot Multibox Detector)
Tanh EF	Tanjant Hiperbolik(Tanh) Etkinleştirme Fonksiyonu
UHD	Aşırı Yüksek Çözünürlük (Ultra High Definition) 4K çözünürlük
YSA	Yapay Sinir Ağı
YTD	Yüksek Dereceden Terimler
YZ	Yapay Zeka

1. GİRİŞ

Hızlı gelişen teknolojiye bağlı olarak yüksek çözünürlüklü ardışık sayısal görüntülerin elde edildiği sayısal kameralar ucuzlamış, erişimi kolaylaşmış kullanım alanları her geçen gün yaygınlaşmaya başlamıştır. İlk başlarda özel olarak yetiştirilmiş personel kontrolünde yapılan pek çok görüntü işleme probleminin çözümünün sayısal ortamda otomatik olarak yapılması gerekliliği ortaya çıkarmıştır. Çünkü sayısal görüntülerdeki verilerin analizi için istihdam edilen nitelikli personel yetişmiş iş gücü kayıplarına neden olmakta, insan kaynaklı hataları arttırmakta hem de bazı durumlarda özel olarak eğitilmiş personeller bile yetersiz kalmaktadır. Bu nedenlerden dolayı sayısal görüntülerin otomatik olarak işlenmesi zorunlu ihtiyaç haline gelmiştir. Ancak çok yüksek çözünürlüklü ardışık sayısal görüntüler çok büyük miktarda veri içerdiğinden sayısal ortamda işlenip faydalı bilginin ortaya çıkartılması yüksek işlemci hızına ve büyük hesaplama kapasitesine ihtiyaç doğurmaktadır. Bu ihtiyaçları karşılamak için bilgisayarla görme alanının konusu içinde yer alan sayısal görüntülerden faydalı bilginin çıkartılması ile ilgili çalışmaları her geçen gün artarak devam etmektedir. Yüksek işlemci hız ve büyük hesaplama kapasitesi sağlansa bile yüksek çözünürlüklü sayısal kameralardan alınan ardışık sayısal görüntülerin gerçek zamanlı olarak işlenmesi, özelliklerinin tespit edilmesi, faydalı bilginin çıkartılması ve sonucun aynı zaman diliminde değerlendirilmesi kullanılan algoritmalara göre çoğunlukla mümkün olmamaktadır. Bir işlemin yapılmasından daha önemlisi, işlemin ne kadar kısa sürede yapıldığıdır. Bazı durumlarda bir işin geç yapılması ile hiç yapılmaması arasında bir fark olmayabilir. Bu nedenle çok yüksek çözünürlüklü sayısal görüntülerin gerçek zamanlı olarak işlenmesi günümüzün en önemli görüntü işleme problemlerindendir.

Bu tez çalışmasının konusu bilgisayarla görme alanı içindedir. Tez çalışması kapsamında Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması (NTİTS) için biyomimetik yaklaşımı temel alan iki aşamalı bir yöntem geliştirilmiştir. Geliştirilen iki aşamalı NTİTS yönteminin birinci aşamasında ilgi alanı hareket olduğu için sayısal kameradan alınan ardışık sayısal görüntülerdeki çok sayıda hareketli nesneyi, hareketsiz nesnelerden ve arka zeminden ayırıştırarak hareketli nesnelerin bulunması gerekmektedir. Ardışık sayısal görüntülerdeki hareketli alanlar tespit edilirken İnsan, maymun ve bazı başka gelişmiş canlıların gözünün çalışma prensibinde esinlenilmiştir. Bu canlıların gözleri görüş alandaki her noktayı aynı

özünörlükte görmez. İlgi alanı olan nesneleri çok yüksek özünörlöklö görürken ilgi alanından uzaklaştıka görüntö netliğı azalır, objeler flulaşır. Böylece ilgi alanın dışındaki alanların ayrıntıları ile beyin meşgul edilmemiş olur. Doğadaki bu özellik kullanılarak çok yüksek özünörlöklö ardışık sayısal görüntülerdeki görüntülerin tüm piksellerini işlemeye kaynak ayırmaktansa sadece hareketli nesneleri oluşturan alanlara ait piksellerin olduğı bölgeler tespit edilerek ayrıştırma işlemi yapılmaktadır. Böylelikle gereksiz iş yükünden kurtulunmuş olunur. Her türlü canlı, cansız, endüktif, kapasitif nesneyi arka plandan, zeminden ve hareketsiz olan diğér nesnelerden ayrıştırarak hareketli nesneye ait bölgeyi bulmak için hareketli nesnelere karşılık gelen piksellerin ışıklılık değérlerindeki değışimleri tespit edilerek anlamlandırılmalı ayrıştırılmalı ve gruplandırılmalıdır. Gruplandırılan bu pikseller hareketli nesnelere karşılık geldiğinden hareketli nesnelere ait görüntüler elde edilir. Elde edilen hareketli nesnelere ait piksel grupları sayısal kameradan alınan görüntülerin boyutlarından daima daha küçüktür. Tespit edilen hareketli nesnelere ait piksel grupları sayesinde hareketli nesneler tespit edilmiş olur. Ardışık sayısal görüntülerdeki nesne hareket ettiğinde bu piksel grupları yer değıştirir. Bu nedenle öncelikle kullanılacak olan hareket tespit yöntemi ardışık sayısal görüntülerdeki hareketli nesneleri etkin olarak tespit edebilmelidir. Hareketli nesnenin, hareketlerine bağılı olarak, nesnelerin boyutlarındaki değışimlere cevap verebilmelidir. Ayrıca hareket tespit algoritmasının sabit ve hareketli kamera uygulamalarında da çalışabilmelidir. Çok yüksek özünörlöklö ardışık sayısal görüntülerde ki hareketli küçük nesnelerin tespitini de yapabilmelidir. İlgi alanı algoritmaları sayesinde işlemcilerin iş yükü azalır, işlem hızları artar. Çok yüksek özünörlöklö görüntülerin gerçek zamanlı işlenebilmesi mümkün hale gelir. İlgi alanı algoritmaları, sistem gereksinimleri azaldığı için kurulum maliyeti ve enerji sarfiyatları da azalır. Azalan enerji gereksinimleri fosil yakıtlardan elde edilen elektrik kullanımını azalttığı için CO2 salınımını azalttığı için ilgi alanı algoritmaları çevre dostu algoritmalarıdır.

Geliştirilen NTITS yönteminin ikinci aşamasında, birinci aşamada OA yöntemiyle elde edilen hareketli nesnelere ait görüntülerdeki nesnelerin özellikleri çıkartılarak, bu hareketli nesnelerin tanımlanması, izlenmesi ve sınıflandırılması işlemleri yapılmaktadır. Nesne tanımlama ve sınıflandırma işlemlerinde Yapay Zeka'nın (YZ) bir alt dalı olan Derin Sinir Ağları (DSA) kullanılmaktadır. DSA'lar büyük verileri tek bir sinir ağı katmanını yerine pek çok katmanda, hesaplamaları tek bir seferde yapabilen hatta tanımlanması gereken parametreleri bile kendisi keşfedebilen, belki de daha iyi parametreler ile değérlendirmelerde bulunabilen bir sistemdir. DSA'lar insan algısına yakın bir doğruluk

oranı ile nesnelerin tanımlamasını ve sınıflandırmasını yapabilmektedir. Grafik İşlemci Birimleri (GPU)'nde algoritmaların kodları Merkezi İşlemci Birimleri (CPU)'ne kıyasla 10-20 kata kadar daha hızlı işlenebilmektedir[1]. Geliştirilen yöntemde gerçek zamanlı işlem yapabilmek için DSA algoritması, GPU'larda çalıştırılmaktadır.

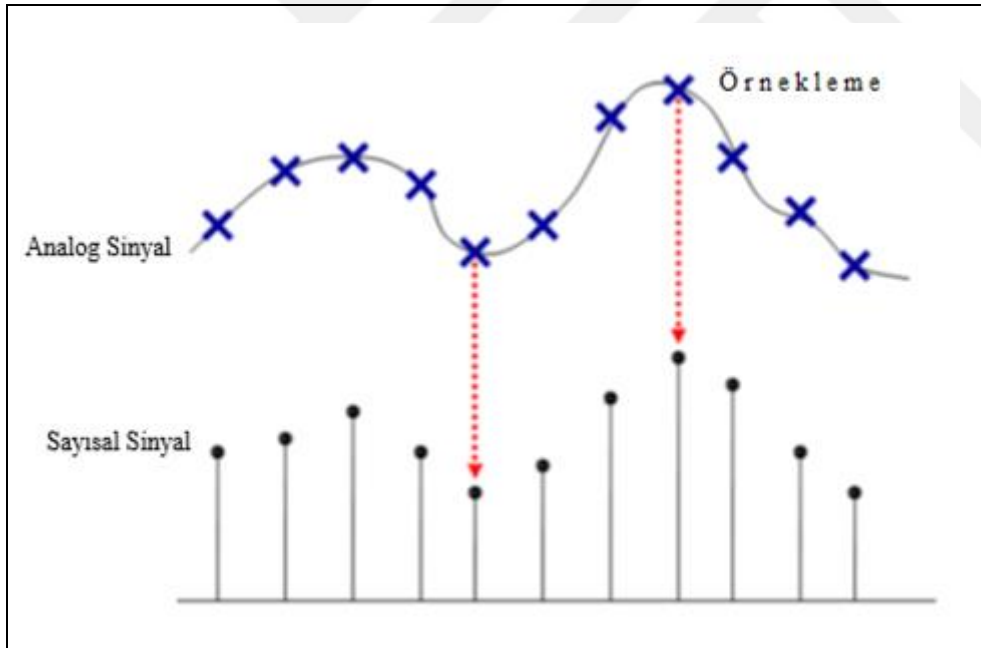
Bu tez çalışması kapsamında geliştirilen NTİTS yönteminin çalıştırılmasından elde edilen sonuçlar, çok yüksek çözünürlüklü ardışık sayısal görüntülerde çok sayıda hareket edebilen her türlü canlı, cansız, endüktif, kapasitif nesnenin, arka plandan, zeminden ve hareketsiz olan diğer nesnelerden aynı türde olsa bile ayrıştırılarak, hareketli nesnenin boyutlarındaki değişimlere de uyarlanabilir bir şekilde hareketli nesnelerin tespiti, izlenmesi tanımlanması ve sınıflandırılması işlemlerinin yüksek doğruluk oranı ile gerçekleştirilebildiğini göstermiştir.

2. GÖRÜNTÜ İŞLEME VE HAREKETLİ NESNE TESPİTİ

Hareketli nesne tespiti, ardışık sayısal görüntülerin işlenerek hareket halindeki bir nesnenin hareket etmeyen diğer nesnelerden ve arka plandan ayrıştırılarak bulunması yöntemini ifade etmektedir. Konunun daha iyi anlaşılmasını için bu bölümde görüntü işleme ile ilgili temel kavramlar, görüntü işleme yöntemleri ve ardışık sayısal görüntülerden hareketli nesnelerin tespit teknikleri hakkında bilgi verilmesinin uygun olacağı değerlendirilmektedir.

2.1. Görüntü İşleme İle İlgili Temel Kavramlar

Görüntüler, 3-Boyutlu (3B) gerçek dünyadaki bir sahnenin, 2-Boyutlu (2B) bir yüzey üzerine düşürülmüş haritası olarak tanımlanabilir. Görüntüler, analog ve sayısal görüntüler olarak iki çeşittir.



Şekil 2.1. Analog ve sayısal sinyaller

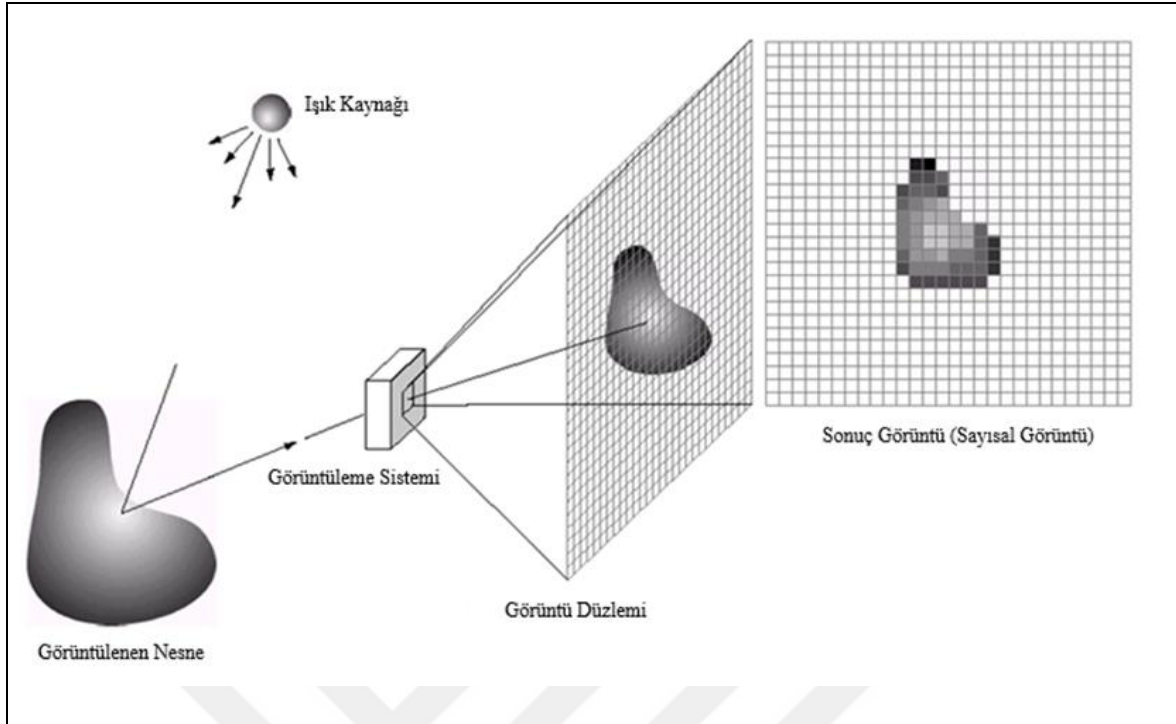
Hem zamana hem de genliğe göre sürekli olan işaretler analog işaretler olarak adlandırılır[2]. Gerçek dünyada algılanan, zamana göre sonsuz değerler alabilen tüm fiziksel algılar analog sinyaldir. Analog sinyaller belirli bir zaman dilimi içinde sonsuz değerler alır. Analog görüntülerde bir resme ne kadar yaklaşılsa yaklaşılsın görüntüyü oluşturan renkler sürekli olarak devam eder, yani analog görüntüler sonsuz çözünürlükte dirler. Analog

görüntüye sonsuz çözünürlüklü görüntüler de denilebilir. Bir ressam tarafından bir tuvale yapılan resim ya da film şeritlerinin kullanıldığı analog fotoğraf makinelerinde şeritlerin üzeri ışığa duyarlı kimyasallar ile kaplanır ve 3B'li ortamdan alınan ışık film üzerine düşürülür ve negatif üzerine işlenir bu yol ile elde edilen fotoğraflar sonsuz çözünürlüğe sahip görüntülerdir.

Gerçek yaşamda algılanan tüm fiziksel algıların, belirli zaman aralıklarında örneklenerek elektronik ortamda işlem yapılabilecek şekle dönüştürülmüş haline sayısal sinyal denilir. Bu tez konusunun çalışma alanına giren sayısal görüntülerin her biri birer sayısal sinyaldir. Yukarıdaki Şekil 2.1'de analog ve sayısal sinyaller gösterilmiştir. Şekil 2.1'de de görüldüğü üzere sayısal sinyaller analog sinyallerin belirli zaman aralıkları ile örneklenmiş halleridir.

Sayısal görüntü, 3B gerçek dünyadaki bir sahnenin kamera benzeri cihazlarla, bulunduğu 3B ortamdaki sahneden doğal ya da yapay ışık kaynaklarından gelen ışık ışınlarının 3B'li ortamdaki nesnelere çarpıp yansıyan ışığın bir mercekten geçirilerek elektronik ortamda örneklenerek işlem yapılabilecek şekilde sayısal sinyallere dönüştürülmüş haritası olarak tanımlanmaktadır[3]. Şekil 2.2'de sayısal görüntünün oluşumu gösterilmektedir.

Sayısal görüntü 2B'li ışık yoğunluk fonksiyonu olarak tanımlanabilir. Burada x ve y resmin uzay koordinatlarını ifade eder, f ise (x, y) koordinatlarındaki genliği yani ışıklılık şiddetini belirtmektedir[4, 5]. Sayısal görüntü, 2B'li ortamda x, y koordinatlarında bir diziden oluşur. Renkli sayısal görüntülerde her bir rengi içeren piksel üç ana rengin (R, G ve B) kombinasyonları olarak temsil edilir, yani renkli bir sayısal görüntü de bir piksel $f_R(x, y)$, $f_G(x, y)$ ve $f_B(x, y)$, bileşenlerinden meydana gelir[6]. Renkli sayısal görüntüler 3B'li matrislerden oluşur.



Şekil 2.2. Sayısal görüntünün elde edilmesi[3]

Varoluşundan bu yana insanlar görüntülerle çalışmıştır ve çalışmaya devam etmektedir. Bu konuda geçmişten günümüze aktarılan birçok kültür mirası vardır. Bunların en önemlileri mozaiklerdir. Mozaik, farklı renklerdeki taş, cam, tuğla, metal, seramik, çini gibi malzemeleri kübik, üçgen prizma veya dörtgen şekillerinde küçük parçalar haline getirip bu parçaları birleştirerek yer ya da duvarları resim ya da motiflerle süsleme sanatına verilen addır. Mozaikteki bir figürü ya da motifi oluşturmak için çok sayıda şekillendirilmiş değişik renklerdeki küçük parçalar bir araya getirilir. Mozaiklerle bir figür ya da motif elde etmek zor ve emek isteyen bir işlemdir. Mozaığın bilinen en eski örneği Milattan Önce 7250-6750 yılları arasında Diyarbakır Ergani ilçesi yakınlarındaki Çayönü Tepesi'nde Neolitik yerleşiminin olduğu yerde bulunan ve dini amaçlar için kullanıldığı düşünülen bu yapıların tabanlarında Terazzo adı verilen döşeme türü mozaikler bulunmuştur. Milattan önceki döneme iat olan bu yapının görselleri piksel kavramının kökeni oluşturmaktadır. Resim 2.1'de bu şekilde elde edilmiş Zeugma antik kentinden ortaya çıkarılan ve günümüzde Gaziantep Arkeoloji Müzesi'nde sergilenen "Çingene Kızı" figür mozaïği bulunmaktadır. Resim 2.1'de dikkatlice incelenecek olursa farklı renklerdeki parçaların bir araya getirilmesi ile figürün meydana getirildiği görülecektir.



Resim 2.1. Çingene kızı mozaïği

Mozaikteki figürü ya da motifini oluşturan parçaların her bir elemanı, sayısal görüntüde sayısal görüntüyü oluşturan en küçük parçaya piksele karşılık gelir. Mozaikteki çok sayıda parçanın bir araya gelmesi ile figürü ya da motif oluşur. Sayısal görüntüler ise piksellerin bir araya gelmesiyle oluşur.

Piksel (Pixel): Piksel kelimesi *Picture element* sözcüklerinin birleştirilmesiyle oluşmuştur, sayısal görüntünün en küçük birim elemanını ifade eder. Piksel sayısal görüntünün en küçük elemanıdır. Sayısal görüntünün satır ve sütunlarının kesiştiği noktalara piksel denilmektedir[3, 5]. Sayısal görüntünün en küçük yapı taşı olan piksellerin aksi belirtilmedikçe en boy oranı eşit olarak hesaplanır. Pikseller renkli görüntülerde R,G,B renklerinin her biri için değer tutarlar ve görülen farklı renkler bu değerlerin birleşiminden meydana gelir.

Parlaklık (intensity): x ve y uzaysal boyutlar olmak üzere $I(x,y)$, x ve y koordinatlarındaki pikselin parlaklık değerini gösterir. Pikselin tuttuğu elektriksel sinyalin şiddetinin ifadesidir.

Çözünürlük (Resolution): Çözünürlük, sayısal bir görüntünün kaç piksel ile temsil edildiğinin ifadesidir. Sayısal görüntü algılayıcısının, görüntülenen gerçek dünyadaki 3B'li

sahneyi örnekleme aralığı olarak ifade edilebilir. Sayısal görüntülerde, örneklemenin çokluğu gerçek dünya ile ilgili sayısal ortama alınan bilgilerin çokluğu yani çözünürlüğü demektir. Aynı boyutlardaki iki sayısal görüntünün netlikleri, barındırdıkları piksel sayılarıyla yani çözünürlükleri ile doğru orantılı olarak artar ya da azalır. Bir sayısal görüntü ne kadar fazla piksel içeriyorsa görüntü o kadar nettir belirgindir ayrıntılar daha fazladır. Sayısal görüntüleri meydana getiren toplam piksel dizilimleri o sayısal görüntünün çözünürlüğü olarak ifade edilir. Aşağıdaki Çizelge 2.1’de günümüzde yaygın olarak kullanılan sayısal görüntü türlerinin çözünürlükleri gösterilmiştir.

Çizelge 2.1. Sayısal görüntü çözünürlükleri

Format	Etiket	Çözünürlük
SD	360p	480 x 360
SD	480p	640 x 480
SD	480p	720 x 480
HQ	576p	720 x 576
HD – 1MP	720p	1280 x 720
Full HD – 2MP	1080p	1920 x 1080
2K (QHD) - 4MP	1440p	2560 x 1440
4K UHD TV - 8MP	2160p	3840 x 2160
Cinema 4K	2160p	4096 x 2160
8K UHD TV	4320p	8192 x 4320
16K	8640p	15360x8640

SD görüntü İngilizce’de Standard Definition kelimelerinin baş harfleri ile ifade edilen standart çözünürlük anlamına kullanılmaktadır. 360p ve 480p olmak üzere iki çözünürlükte SD görüntü olarak ifade edilir. SD 480p ve 720x480 olan çözünürlük geniş ekran olarak bilinir. Teknolojinin getirdiği yenilikler sayesinde SD çözünürlük artık günümüzde kullanılan en düşük çözünürlükteki görüntü olmuştur. SD görüntüler küçük ekranlarda görüntü kalitesi farkı gözle çok belli olmasa da büyük ekranlarda çok daha düşük kalitede olduğu görünür. Düşük internet hızının olduğu yerlerde yaygın olarak kullanılmaya devam etmektedir.

HQ görüntü İngilizce’de High Quality kelimelerinin baş harfleri ile ifade edilen Yüksek Kaliteli görüntü anlamındadır. SD görüntüler ile karşılaştırıldığında görüntü kalitesi iyi olsa da HD görüntü ve üzerindeki görüntü türlerine göre çözünürlüğü düşük kalmaktadır. 720x576 çözünürlük HQ türü çözünürlük olarak anılmaktadır.

HD görüntü İngilizce 'de High Definition kelimelerinin baş harfleri ile ifade edilen Yüksek Çözünürlük anlamına gelmektedir. HD çözünürlük 720p olarak adlandırılır. HD çözünürlük, sayısal görüntünün yatay ekseninde 1 280, dikey ekseninde 720 piksel olmak üzere toplam 921 600 adet pikselden meydana gelir. 720p görüntüye yüksek çözünürlüklü görüntü denilir.

Full HD görüntü İngilizce'de Full High Definition demektir. Full ve High Definition kelimelerinin baş harfleri ile ifade edilen Çok Yüksek Çözünürlük anlamına gelmektedir. Full HD çözünürlük 1 080p olarak adlandırılır. Çok yüksek çözünürlük-Full HD çözünürlük, yatay ekseninde 1 920 adet piksel, dikey ekseninde 1 080 adet piksel olmak üzere toplam 2 073 600 tane pikselden meydana gelir.

K İngilizcede bin anlamında kullanılır. K terimi bin anlamında ilk kez 1945 yılında Arthur E. Fitzgerald tarafından Temel Elektrik Mühendisliği kitabında kullanılmıştır[7]. Günümüzde 1K-2K-4K-8K-16K çözünürlük olarak adlandırılan sayısal görüntüler vardır. K yatay olarak bir satırın barındırdığı piksel sayısını ifade eder. "K" standardı, Paramount Pictures, Sony Pictures Entertainment, 20th Century Fox, Universal Studios, The Walt Disney Company ve Warner Bros gibi büyük sinema film stüdyolarının 2002 yılında oluşturduğu Digital Cinema Initiatives LLC (DCI) birliği tarafından, K çözünürlük standardı olarak 2005 yılında belirlenmiştir[8]. 2K görüntü bir satırda 2 048 piksele sahip olduğu için 2K denilmiştir. 4K görüntünün bir satırında 3 840 adet piksel vardır. 3,84 K yerine yukarıya yuvarlanarak 4K denilmiştir.

QHD görüntü İngilizce'de Quad High Definition kelimelerinin baş harfleri ile ifade edilen piksel sayısı $2\,560 \times 1\,440$ olan, çözünürlük türüdür. QHD görüntüye 2K çözünürlüklü görüntüde denilmektedir

UHD görüntü İngilizce 'de Ultra High Definition kelimelerinin baş harfleri ile ifade edilen Aşırı Yüksek Çözünürlük anlamına gelmektedir. UHD görüntüye 4K çözünürlüğe sahip görüntüye denir. $4K = 4\,096 \times 2\,160 = 8\,847\,360$ piksele sahiptir.

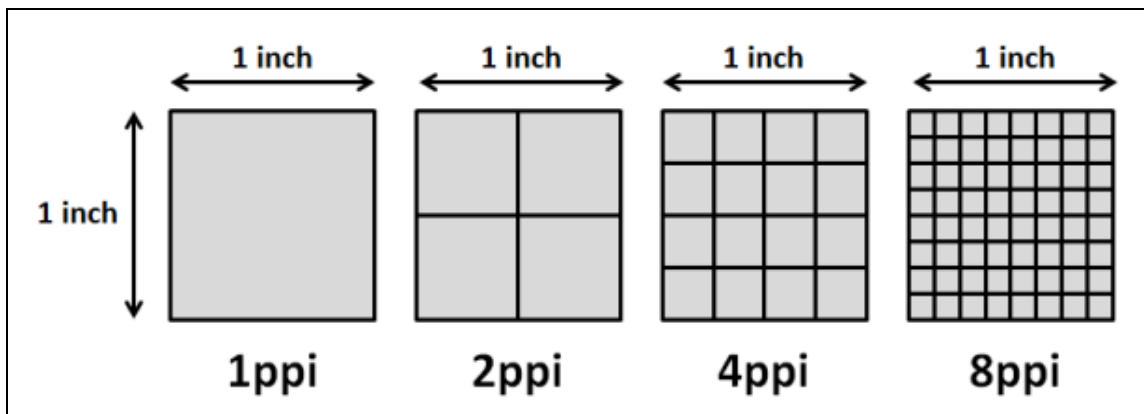
8K çözünürlüğe sahip sayısal görüntü $8\,192 \times 4\,320 = 35\,389\,440$ piksele sahiptir. 8K görüntü HD görüntünün yaklaşık 16 katı kadar daha fazla piksel içermektedir.

Çözünürlük sayısal bir görüntüyü oluşturan yatay ve dikeydeki tüm piksellerin toplam sayısına denir. Örneğin FullHD bir görüntü 2 Mega Piksel (MP) 1 920yatay ve 1 080 dikey piksellerden oluşan çözünürlüğe sahiptir toplam 2 073 600 adet piksel bulunmaktadır. Mega Piksel (MP) 1 milyon pikseli ifade eder. FullHD görüntüye 1 080p görüntüde denilmektedir. 1 080p'deki "p" harfi progressive yani ilerleyici demektir. İlerleyici görüntü ile daha akıcı daha hızlı görüntü hareketlerinde donma bulanıklaşma, buğulanma gibi olumsuzluklar olmaz. Sayısal görüntüyü oluşturan piksellerin sayısı arttıkça her zaman detaylı bir sayısal görüntü elde edilir.

Görüntü çözünürlükleri, teknolojinin gelişimine bağlı olarak artmaktadır. İlerleyen zaman dilimlerinde görüntü çözünürlüklerinin daha yüksek değerlere ulaşması ile görüntülerin hızlı işlenmesinin gerekliliği çok daha iyi anlaşılabacaktır.

PPI (pixels per inch): Görüntüdeki piksellerin yoğunluğudur. Bir inch'lik alana düşen piksel sayısı pixels per inch (PPI) olarak ifade edilir. Bir inch, 2,54cm'dir. PPI, taranan her inç karede hem yatay hem de dikey olarak piksel sayısını açıklar. Örneğin, 100 PPI demek bir inch'lik uzunluğa 100 adet pikselin bir kenarının toplam uzunluğuna eşit demektir. Bir inç karede ise $100 \times 100 = 10000$ adet pikselin olduğunu belirtir[9].

Şekil 2.2'de PPI ile inch arasındaki ilişki görülmektedir. PPI değeri yükseldikçe görüntüdeki piksel yoğunluğu artacağı için görüntünün çözünürlüğü yani detayları artar. 400PPI görüntünün 72PPI görüntüye göre fazla piksele sahip olduğu, görüntünün çözünürlüğünün yüksek olduğu ve görüntüye ait çok daha fazla bilgi içerdiği anlaşılır.



Şekil 2.3. PPI ile inch arasındaki ilişki

Morfolojik işlemler: İkili (binary) görüntülerdeki filtreleme, kenar bulma, inceltme, gürültü giderme vb. gibi görüntü işleme yöntemlerinin genel adıdır.

Saniyedeki kare sayısı (Frame Per Second (FPS)): 1 saniyelik video görüntüsündeki toplam görüntü kare sayısıdır. Yani bir saniyelik sayısal video görüntüsünü oluşturan toplam görüntü sayısını ifade eder. Günümüzde 24fps, 30fps ve 60fps sayısal görüntüler yaygın olarak kullanılmaktadır.

2.1.1. Sayısal görüntü işleme

Ölçülmüş ya da kaydedilmiş olan sayısal görüntülerden faydalı bilginin çıkartılmasına yönelik sayısal ortamda, işlemciler üzerinde yazılım programları ile yapılan çalışmaları kapsamaktadır. Bir başka deyişle sayısal görüntüden faydalı bilgilerin çıkartılması için yapılan tüm işlemleri kapsamaktadır.

Görüntü işleme, sayısal kameralar yardımı ile 3B’li gerçek ortamdaki verilerin yakalanarak 3B’li gerçek ortamdaki verileri ölçme ve değerlendirme işlemlerinden geçirip, başka bir elektronik cihaza görüntülenebilir hale getirilmesi ya da bir elektronik ortamdan başka bir elektronik ortama aktarılmasına yönelik bir çalışma olan görüntü işleme “sinyal işlemeden” farklı bir işlemdir[3, 5]. Görüntü işleme, sayısal ortama alınan görüntülerde belirli bir amaç için yapılan, iyileştirme bölütleme, ayrıştırma, yabancılaştırma, keskinleştirme, bulanıklaştırma vb. gibi yapılan tüm işlemleri kapsamaktadır.

Tarihteki ilk sayısal görüntü 1920’li yılların başında Londra ve New York arasında Bartlane kablolu görüntü aktarım sistemi kullanılarak denizaltı kablolarıyla gazete görüntüleri iletilmiştir[10-13]. Özelleştirilmiş ekipmanlarla görüntü, kablolu iletimi için sayısal olarak kodlandı ve ardından alıcı tarafta alınan kodlar yeniden yapılandırdı[12, 13]. Bu ilk sayısal görüntünün, görüntü kalitesinin iyileştirilmesi, baskı üreticilerinin seçimi ve yoğunluk seviyelerinin dağıtımları gibi problemler görüntü işlemenin ilk problemleri olmuştur[11, 12]. Bartlane sistemi ile Avrupa’da savaşın başladığı 1939 yılına kadar önemli haber olayları içeren yaklaşık 500 fotoğrafın başarılı bir şekilde iletilmesini sağlamıştır[10]. Görüntü işleme, NASA’nın, esasen Jet Propulsion Laboratuvarı (JPL)’de yapılan çalışmalar yoluyla, bu alana açıkça liderlik ettiği bir alandır[14, 15]. 1960’lı yıllara gelindiğinde PPL, Massachusetts Teknoloji Enstitüsü, Bell Labs, Maryland Üniversitesi ve diğer araştırma

birimlerinde görüntü işleme ile ilgili; uydu görüntüsü, tele-foto, tıbbi görüntüleme, görüntülü telefon, karakter tanıma ve fotoğraf iyileştirme uygulamaları ile ilgili çalışmalar yapılmıştır[16]. 1960'larda, işlem gücü yüksek olan bilgisayarların geliştirilmesiyle sayısal görüntü işleme doğdu[3, 13]. 1964 yılında Ranger 7, ABD uzay aracı tarafından çekilen Ayın Uydu Görüntüleri, Kaliforniya'daki Jet Propulsion laboratuvarında işlendi[3, 17]. 1960'tan itibaren dijital görüntü işleme tekniklerinin kullanımı oldukça yaygınlaştı[13]. 1970'lerin başında tomografi cihazının icadı, tıbbi tanıda görüntü işleme uygulamalarının en önemli olaylardan biri olmuştur[3]. 1979 yılında Allan M. Cormack ve Godfrey N. Hounsfield geliştirdikleri bilgisayar destekli tomografi cihazını geliştirmelerindeki başarılarından dolayı Nobel Fizyoloji veya Tıp Ödülü, almaya hak kazanmışlardır[18].

2.1.2. Görüntü işleme ile çözünürlük arasındaki ilişki

Görüntü işleme; ölçülmüş ya da kaydedilmiş olan sayısal görüntülerden faydalı bilginin çıkartılmasına yönelik sayısal ortamda yazılım programları ile yapılan işlemleri kapsamaktadır. Çözünürlük ise bir sayısal görüntüyü oluşturan toplam piksel sayısıdır. Görüntüyü oluşturan piksel sayısı ne kadar çok ise görüntünün alındığı ortam ile ilgili o kadar çok bilgi içerir. Sayısal görüntüye ait ne kadar çok bilgi var ise görüntü işleme teknikleri ile görüntüden çok daha çok faydalı bilgi elde edilir. Dolayısı ile yüksek çözünürlüklü sayısal görüntülerin işlenmesinden elde edilen sonuçlar çok daha sağlıklı ve çok daha başarılı olur. Ancak yüksek çözünürlüklü görüntüler daha çok veri içerdiklerinden dolayı bu görüntülerin işlenmesi daha yüksek işlemci gücü ve daha çok zamana gereksinim duyarlar. Bu nedenle bu konuya olan ihtiyaçların artmasıyla bilim insanları artan veri miktarlarını en az sistem gereksinimleri ile en kısa sürede çözmek için yeni yöntemler geliştirmeye çalışmaktadır.

2.1.3. Gerçek zamanlı görüntü işlemede karşılaşılan sorunlar

Gerçek zaman; içinde bulunulan zamanı, yaşanmakta olan anı, yani şimdi ki zamanı ifade eder. Gerçek zamanlı görüntü işleme ise sayısal kameradan alınan, ardışık sayısal görüntülerin anlık olarak alınıp, işlenmesi ve faydalı bilginin çıkartılması ve sonucunun kullanıcıya aynı zaman diliminde bildirmesidir. Yani gerçek zamanlı görüntü işlemede; anlık olarak görüntüler alınıp, işlenerek sonucunun kullanıcıya gösterilmesidir.

Bir görüntü karesindeki veri miktarı görüntünün çözünürlüğüne bağlıdır. Örneğin sayısal kameradan alınan görüntü 1920×1080 piksel renkli görüntü olsun. Bu renkli görüntü karesi $1920 \times 1080 \times 3 = 6\,220\,800$ pikselden oluşur. Her bir piksel 1 byte'dan oluşur. Yani 1920×1080 piksellik sayısal bir görüntü 6 220 800 byte lık bir veri demektir. Bir sayısal kameradan 1 sn'de 24 adet görüntü karesi alınıyorsa $6\,220\,800 \times 24 = 149\,299\,200$ byte lık bir veri bir saniyede elde edilir. Bu kadar büyüklükteki verinin sayısal ortamda işlenip faydalı bilginin ortaya çıkartılması yüksek hesaplama kabiliyeti olan sistemleri gerektirmektedir. Görüntü işleme yönteminde kullanılan algoritmaların zorluğuna bağlı olarak çok yüksek çözünürlüklü görüntülerin bazı durumlarda gerçek zamanlı olarak işlenebilmesi mümkün olmamaktadır. Bu nedenlerden dolayı çok yüksek çözünürlüklü ardışık görüntüler gerçek zamanlı işlenmesi zordur.

2.1.4. Görüntü işlemenin kullanıldığı alanlar

Günümüzde görüntü işleme pek çok alanda kullanılmaktadır. Her geçen gün teknolojinin sunduğu imkânlar sayesinde görüntü işlemenin kullanım alanı giderek genişlemektedir,

- Tıp, tümör, doku analizi, damar gibi yapıların analizi, kemiklerin yapısı, tomografi, ultrason, x-ray görüntüler, biyomedikal görüntüler
- Askeri uygulamalarda (denizaltı sonic dalga taramaları), termal görüntüleme, sualtı görüntüleme, radar, hava, uydu görüntüleri gibi uygulamalarda
- Güvenlik, yüz tanıma, hareket algılama, kriminal laboratuvarlar
- Robotik, trafik,
- Astronomi, radar, uydu görüntüleri, gezegenlerin hareketleri, bulut hareketleri
- Gazetecilik ve fotoğrafçılık endüstrisi uygulamalarında
- Tarım ve hayvancılık (sığır eti kalite tayini) ,
- Jeodezi, yeryüzü haritalarının çıkartılması, yeryüzü doğal yapılarının incelenmesi
- Fizik, sanat, biyomedikal alanlarda
- Uzaktan algılama uygulamalarında
- Uydu görüntülerinden nüfus yoğunluğu, yerleşim alanları, çevre kirliliği gibi uygulamalarda
- Çevresel şartların tanımlanmasında
- Uydulardan alınan sayısal görüntülerden hava gözlemleri ve hava durumu tahmini vb. gibi uygulamalarda
- Güvenlik sistemlerinde yüz tanımada, parmak izi tanıma ve retina tanımada

- Duygu Analizi
- Eski ya da hasar görmüş fotoğrafların onarılması, film sektörü vb. gibi alanda oldukça yaygın olarak kullanılmaktadır.

Görüntü işleme yapılırken bilgisayar grafiği, hesaplamalı fotoğrafçılık, görsel algılama, görüntü kodlama, sinyal işleme, istatistik bilgi teorisi, derin öğrenme, bilgisayarla görme, yapay zeka, makine öğrenmesi vb. gibi disiplinler arası çalışmalardan da faydalanılmaktadır.

2.2. Görüntü İşleme Yöntemleri

Sayısal görüntülerin işlenmesi, görüntü onarımı(image restoration), bir şekilde bozulmuş bir görüntüyü kurtarmaya, onarmaya çalışan bir işlemdir. Görüntü geliştirme ya da iyileştirmede(Image enhancement) görüntü çoğunlukla buluşsal tekniklerle görüntüyü daha iyi hale getirir, böylelikle görüntüden daha sağlıklı yararlı bilgiler çıkartılabilir. Görüntü geliştirme orijinal veya gözlenen görüntüde bulunmayan yeni özellikler oluşturan görüntüden görüntüye görsel olarak daha hoş görüntüler oluşturmak için kullanılır[19]. Tıbbi görüntüleme bağlamında, görüntüleri iyileştirmek klinik veya bilimsel uygulamalarda analizleri kolaylaştırmak için kullanılır[19]. Görüntü iyileştirme teknikleri, bozulmanın geri döndürülebilmesi için istatistiksel veya matematiksel bir tanımının bulunduğu bozuk görüntüleri işlemeyi amaçlar. Görüntü analizi(Image analysis), bir görüntünün tercihen otomatik olarak işlenmesine görüntü içindeki nesnelerin ölçülmesine gruplandırılmasına veya sınıflandırılmasını içerir, böylece istenilen bilgiler otomatik olarak görüntüden elde edilerek kullanıcıya sunulur. Görüntü analizi örnekleri, görüntü bölümleme, kenar çıkarma ve doku ve hareket analizleri yapar[6, 20]. Görüntü sentezi, diğer görüntülerden veya görüntü olmayan verilerden yeni görüntüler oluşturulur, örneğin, x-ışını bilgisayarlı tomografi de olduğu gibi kestirim verilerinden eksenel veya “dilim” tomografik görüntülerin yeniden yapılandırılmasıdır[20]. Görüntü sıkıştırma(Image compression), görüntüyü tanımlamak için gereken veri miktarını azaltır. Görüntü dosyaları görüntünün çözünürlüğüne göre büyük dosya boyutları gerektirir. Özellikle video görüntüleri gereksiz veya tekrarlayan bilgiler içermeye eğilimindedir. Görüntü sıkıştırma işlemi ile dosya boyutu küçültülür, böylelikle görüntü daha kısa sürede daha verimli bir şekilde, elektronik olarak depolanabilir veya taşınabilir[20].

Tıbbi görüntü işleme, tıbbi görüntülerin çeşitli görüntü işleme algoritmaları yardımıyla kontrast arttırma, gürültü azaltma, kenar keskinleştirme, kenar algılama, bölümlendirme vb. gibi tekniklerle hastalık tespiti sürecini otomatik veya yarı otomatik hale getirerek hekimlere yardımcı olur[21].

2.3. Hareketli Nesnelerin Tespiti

Ardışık sayısal görüntülerden hareketli nesne tespiti; hareket halindeki bir nesnenin hareket etmeyen diğer nesnelerden ve arka plandan ayrıştırılarak tespit edilmesidir. Sayısal görüntülerdeki hareketli nesnelerin tespit edilmesi bilgisayarla görme konusunun çalışma alanıdır. Hareketli nesne tespiti, zeki görüntü işleme istemleri için çok önemli bir araştırma konusudur[22]. Tez çalışmasının amacı kapsamında yapılan çalışmalarda, yüksek çözünürlüklü sayısal kameradan alınan ardışık sayısal görüntülerde hareket edebilen her türlü canlı, cansız, endüktif, kapasitif varlığı arka plandan ve hareketsiz diğer nesnelerden ayrıştırarak, hareketli nesnenin boyutlarındaki değişimlere de uyarlanabilir bir şekilde tespit etmektir. Sayısal görüntülerdeki hareket tespiti görüntü işleme konusundaki pek çok problemin temelini oluşturmaktadır[23]. Herhangi bir sayısal kameradan alınan sayısal görüntü $m \times n$ boyutlarında bir tam sayı matrisini ifade eder[5]. Sayısal kameralardan alınan sayısal görüntüler $m \times n$ boyutlarındaki görüntü matrisinin her bir elamanı, bir pikseli ifade eder[5]. Pikseller görüntünün en küçük birimidir. Piksellerin her biri bir değer tutar[5]. Bu değerlerin karşılığında görüntü(ler) oluşur[5]. Gerçek dünyada 3B'li ortamda meydana gelen hareketler, değişimler sayısal görüntülerde piksellerin değerlerindeki değişimlere karşılık gelmektedir. Bu sebeple ardışık sayısal görüntülerdeki hareketleri bulmak için görüntüyü oluşturan piksellerin ışıklılık değerlerindeki değişimlerden faydalanılır.

Sayısal görüntülerdeki hareketin tespiti için geliştirilmiş temel yaklaşımlar; Arka Plan Çıkarma, Fark Alma, Zamansal Görüntü Farkları ve OA olarak dört forma ayrılabilir[24, 25]. Sabit kamera görüntüleri kullanılarak yapılan hareketli nesnenin belirlenmesinde arka plan çıkarma, fark alma (çerçeve farkı) ve zamansal görüntü farkları gibi geçici bilgi tabanlı yöntemlerdir[24].

Ardışık sayısal görüntülerden, hareketli nesne tespit yöntemleri:

Arka plan çıkarma (background subtraction): bu yöntemde sabit kameradan alınan boş bir sahnenin görüntüsü referans olarak kabul edilir. Daha sonra sayısal kameradan alınan, görüntüler referans görüntüden piksel piksel çıkartılarak sahneye giren ya da sahnede hareket eden hareketli nesne ya da nesneler tespit edilir[22]. Referans görüntü yabancı nesne içermeyen sabit alanı içerdiği için arka plan olarak kabul edilir. Sonradan görüntüye giren nesneler bu arka alandan ayrıştırılmış olur. Bu nedenle bu yönteme arka plan silme yöntemi denilmiştir. Sürekli olarak referans görüntü ile sonradan alınan görüntü farklarının alınması temeline dayanmaktadır. Arka plan görüntüsünün yenilenerek yapıldığı uygulamaları vardır[26]. Arka plan rengi ile ön plandaki nesneler aynı renklere sahip olabilir. Rüzgarda salınan ağaç akan nehir arka plan tamamen hareketsiz değildir. Ortamın ışık değişimlerine karşı çok duyarlıdır. Bulutların hareketi ile ortamın ışıklılığı değişimi yöntemin çalışmasını olumsuz etkiler.

Fark alma ya da görüntü farkı (frame differencing): sayısal görüntülerdeki hareketli nesnelerin tespiti, iki ardışık sayısal görüntü arasındaki farkın hesaplanmasıyla belirlenir[22]. Hesaplama ardışık sayısal görüntülerdeki ikinci görüntüden, birinci görüntü çıkartılır. İki görüntü karesi arasındaki fark değişim olan piksellere karşılık gelir. Hesaplanması basit ve uygulanması kolaydır. Çeşitli dinamik ortamlar için, iyi bir uyarlanabilirliğe sahip olabilir, ancak genellikle hareketli nesnenin tam olarak elde etmek zordur[27].

Zamansal görüntü farkları (temporal differencing): zaman serisi görüntüsünü temel alan birkaç ardışık görüntü kullanır. Arka plan çıkarmaya çok benzerdir. Hareketli nesneleri fark alma yöntemiyle belli bir eşik değeri ile tespit ederek hareketli nesneyi tespit eder. Bu yöntem, hareketli nesne algılama diğer algoritmalarına kıyasla basit ve kolaydır. Bununla birlikte, dinamik sahnelerle karşı çok savunmasızdır, çok fazla gürültü içerir genellikle bazı hareketli nesnelerin tüm ilgili piksellerini tespit etmede başarısız olur[22, 28]. Bu yöntem nesnelerin algılanması, karmaşık sahnelerin hesaplanması için geçerli değildir ve gerçek zamanlı uygulamalarda kullanılamamaktadır[28].

Optik akış (optic flow) : ardışık sayısal görüntülerdeki hareketli alanları tespit etmek için zamanla değişen hareketlerin hız vektörlerini hesaplar[29]. Hareketli kamera uygulamalarında daha iyi sonuç verir[30], ancak OA algoritması karmaşık hesaplama ve ayrıca gerçek zamanlı uygulamalarda özel donanım desteğine ihtiyacı duyar[28].

Ardışık sayısal görüntülerde hareket eden nesneyi tanımlamak, sınıflandırmak ve takip etmek için ilk adım olarak hareketin tespit edilmesi gerekmektedir. Tespit edilecek hareketli nesnelerin boyutları hareket halinde iken hareketli nesnenin özelliklerine göre değişebilir. Kullanılacak hareket tespit yöntemi hareketli nesnelerin boyutlarındaki değişimlere uyarlanabilir şekilde hareketli nesneleri tespit etmelidir. Bu nedenle hareket tespit algoritmaları incelenmiş ve OA yönteminin kullanılması kararlaştırılmıştır. Çok yüksek çözünürlüklü kameradan alınan ardışık sayısal görüntüler, OA hareket tespit algoritması ile ardışık sayısal görüntülerdeki değişimlerin olduğu piksellerden nesnelerin hız vektörleri tespit edilir ve gruplandırılır. Gerçek dünyada harekete karşılık gelen bu piksel gruplarından hareketli bölgeler tespit edilip, kesilip çıkartılarak hareketli nesnelerin tespiti yapılmıştır.

2.3.1. Optik akış (Optic flow)

Bilgisayarla görmede, hareket tespitinde önemli bir yeri olan OA kavramı ilk defa Amerikalı psikolog Gibson tarafından 1950’de dile getirilmiştir[31]. OA en basit bir şekilde tarif edilecek olursa görsel bilgi akışı olarak tarif edilebilir.

OA, görüntüdeki piksellerin ışıklılık değerlerinin, hareketlerin, görünür hızlarının dağılımlarıdır[29]. OA, nesnelerin hareketleri ve görünüşleri ile ilgilidir[29]. 2B’li sayısal görüntü dizilerinin işlenmesindeki asıl problem OA’nın veya görüntü hızının ölçülme zorluğudur. OA, 2B’li sayısal görüntülerdeki hareket alanına yaklaşık bir hesaplama yapmaktadır[32].

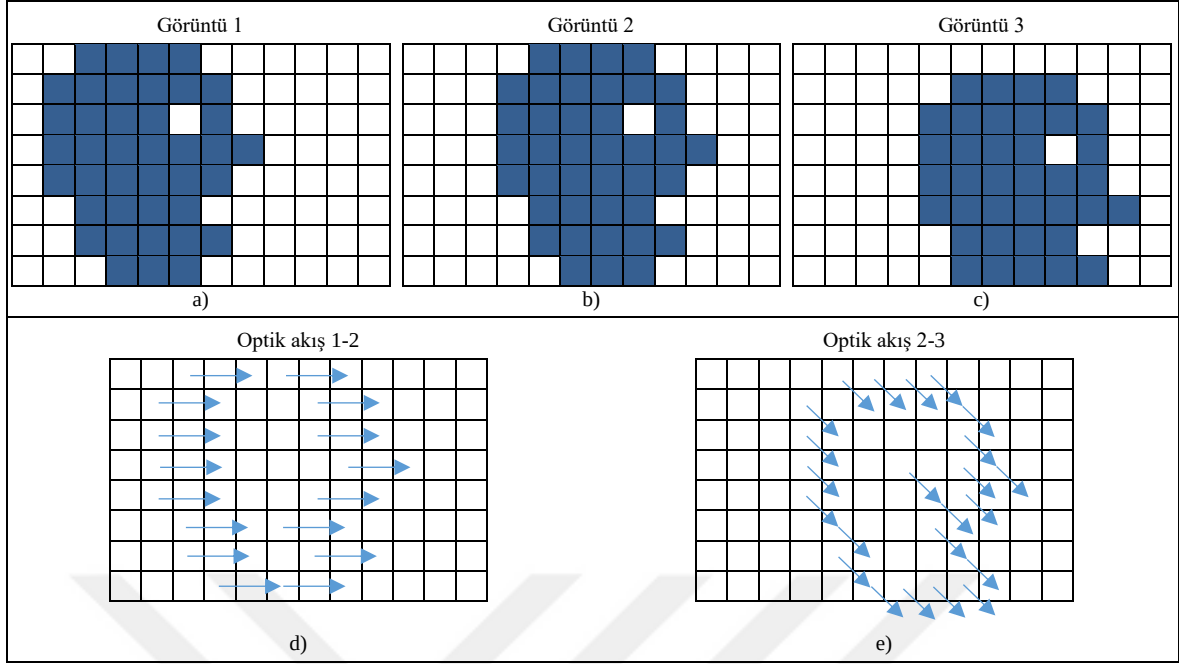
Görüntülerdeki hareketin analizi için üç boyutlu (3B) gerçek dünyadaki hareketin sayısal görüntülerde, 3B’li gerçek dünyadaki ortamın görünür görüntüsünün bulunduğu sahnenin, görüntüsünün 2B’li ortamdaki izdüşümünden yararlanarak, görüntünün anlık hız bilgisini tüm pikselleri kullanarak çıkaran ve hızlarını vektörel olarak ifade eden bu yöntem OA metodu denir. OA metodu günümüzde bilgisayarla görme uygulamaları, yapay zeka, çarpışmadan kaçınma ve robotik gibi alanlarda sıklıkla kullanılmaktadır. OA, görüntülerdeki piksellerin ışıklılık değerlerinin görüntüleri, görüntülerdeki nesnelerin görünür hızlarının vektörel olarak ifadesidir.

Bilgisayarla görme konusunda sayısal kameralardan alınan 3B’li gerçek dünyadan alınan görüntünün 2B’li düzleme izdüşümünden elde edilen sayısal görüntüler $m \times n$ boyutlarında

ardışık sayı dizeleri olarak ortaya çıkar. 3B'li gerçek dünyaya karşılık gelen 2B'li bu sayı dizelerinin her bir elemanın tuttuğu değer, bir pikselin değerine karşılık gelmektedir. Gerçek ortamda sayısal kameranın görüntü aldığı ortamda bir nesne hareket ettiğinde 2B'li sayı dizisinde de hareket eden nesneye karşılık gelen bölgeyi ifade eden piksellerin değerlerinde değişimler olur. OA yöntemi piksel değerlerindeki bu değişimlerden yola çıkarak nesnelerin hareketinin, yönünü ve hızını bulmaya çalışır. Görüntü dizilerinin işlenmesindeki temel problem OA'nın veya görüntü hızının ölçülmesi zorluğudur. OA, iki boyutlu hareket alanına yaklaşık bir hesaplama yapmaktadır[32, 33].

OA'yı daha iyi anlayabilmek için örneğin, bir araç ile seyahat halinde giderken iken yolda kenarlarda ağaçlar, dağlar, tepeler, trafik levhaları vb. gibi pek çok nesneler görülür. Yol boyunca cisme doğru hareket edildiğinde uzaktaki nesneler sürekli olarak büyür, nesneye doğru hareket devam ettikçe etrafta bulunan nesnelerin kenarları görüş alanının dışına doğru hareket eğiliminde olur. Araç hareketine devam edip nesneler geçildikten sonra arkaya bakıldığında ya da dikiz aynasından arkada kalan nesneler izlendiğinde, nesnelerin sürekli küçüldüğü ve kenarların büzüştüğü gözlemlenir. "Nesnenin görüntüsünün kenarlarının büyüyüp küçülmesiyle 3B'li uzayda bulunan nesnelerin 2B'li bir düzleme izdüşümü alınmaktadır"[34]. OA tekniği, sayısal görüntülerdeki nesnelerin hızın şiddetini ve istikametini bulmaya çalışmaktadır[34].

Aşağıdaki, Şekil 2.3 a,b,c'de üç görüntü karesinde de bir insan kafası silueti görülmektedir. İnsan kafası siluetinin üç görüntü karesi boyunca ki hareketlerini göstermektedir. OA tekniği birinci ve ikinci görüntüler arasındaki, ikinci ve üçüncü görüntüler arasındaki insan kafası siluetinin sınır piksellerini yer değişimlerinin hızını ve yönünü vektörel olarak Şekil 2.3. d,e'de göstermektedir. Şekil 2.3 a'dan b'ye düz iki kare düz (soldan sağa doğru iki kare) ilerlerken Şekil 2.3 d'de ilerlemenin yönü ve şiddeti vektörel olarak ok işareti ile gösterilmiştir. Yine Şekil 3.1 b'den c'ye bir kare ileri(soldan sağa doğru bir kare) yönde, bir karede aşağı yönde ilerlemiştir. Şekil 2.3 d'de bu ilerlemenin yönü ve şiddeti vektörel olarak ok işaretleri ile gösterilmiştir. Bu şekildeki görüntülerden tespit edilen nesnelerin hareketlerinin vektörel olarak ifade edilip gösterilmesine OA denilmektedir.



Şekil 2.4. Ardışık üç tane sayısal görüntü karesinde insan kafası silueti görülmektedir. a) ve b)'de birinci ve ikinci görüntü kareleri ve bu iki görüntü karesi arasındaki hareketin yönü ve şiddeti d)'de görülmektedir. b) ve c)'de ikinci ve üçüncü görüntü kareleri ve bu iki görüntü kareleri arasındaki hareketin yönü ve şiddeti e)'de görülmektedir[35]

OA hesaplaması için bilim adamları tarafından birçok teknik geliştirilmiştir. Genellikle bu teknikler geliştiren kişi ya da kişilerin adı ya da adlarıyla anılmıştır[35]. OA'nın hesaplamasında kullanılan bu algoritmaları Baron ve arkadaşları incelemiş ve sınıflandırılmışlardır[32]. Baron ve arkadaşları bu algoritmaları dört grupta incelemişlerdir[32].

Optik akış teknikleri

- 1) Fark teknikleri
 - a) Horn schunck
 - b) Lucas Kanade
 - c) Nagel
 - d) Uras
- 2) Alan tabanlı teknikler
 - a) Anandan
 - b) Singh
- 3) Enerji tabanlı teknikler
 - a) Heeger

4) Faz tabanlı teknikler

- a) Waxman
- b) Fleet Jepson

Bu OA çözüm yöntemleri arasında uygulama ve performans farklılıkları vardır. Ancak yine de birçok bakımdan birbirlerine denk olarak düşünülebilir.

Bu tez çalışması boyunca yapılan çalışmalarda fark teknikleri arasında yer alan genel fark tekniği temeline dayanan Horn Schunck OA yöntemi kullanılmıştır.

2.3.2. İki boyutlu hareket kısıtlama denklemi

Bilgisayarla görmede görüntü 3B'li gerçek dünyadan 2B'li düzleme izdüşümü ile aktarılan sayısal görüntülerin, 2B'li bu düzlemde t anındaki $r = (x, y)$ noktasındaki görüntünün bir pikselinin parlaklığı $I(x, y, t)$ ile ifade edilir. Görüntüdeki belirli bir noktanın parlaklığının zamanla sabit olduğu varsayımı yapılır.

$$\frac{dI}{dt} = 0 \quad (2.1)$$

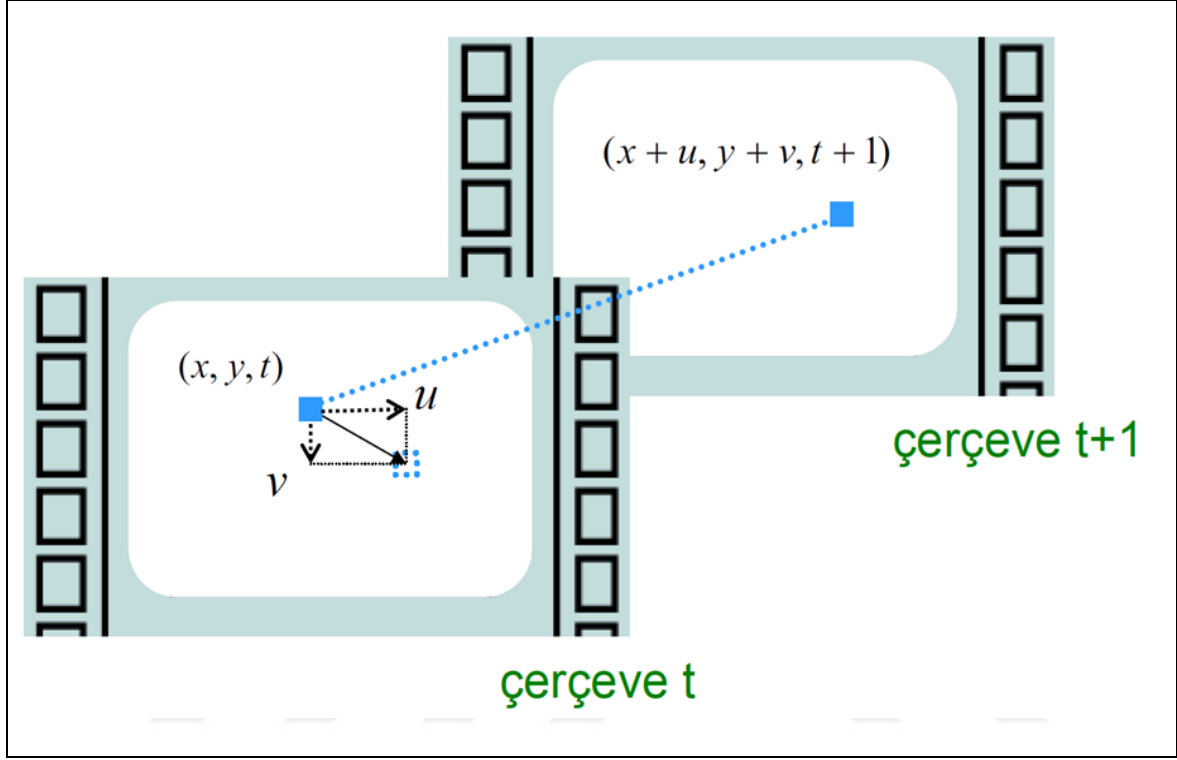
$m \times n$ komşuluğundaki t anındaki $r = (x, y)$ noktasındaki pikselin $I(x, y, t)$ olduğunu ve dx, dy zamanda dt ile ifade edilir. Aşağıda denklem (2.2)'deki gibi yazabilir.

$$I(x, y, t) - I(x + dx, y + dy, t + dt) = 0 \quad (2.2)$$

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \quad (2.3)$$

Denklem (2.2)'den $I(x, y, t)$ eşitliğin diğer tarafına aktararak denklem (2.3) elde edilmiştir. Hareketin tespit edilebilmesi için şiddetinin zamanla değişiminin olmadığı kabulü yapılır. Şiddet değişmezliği kabulü ile hareket kısıt denkleminin temelini oluşturur. Aksi halde hareket tespiti mümkün olmaz.

Görüntülerdeki bir noktanın t zamanından $t + 1$ zaman sonraki hareketi ile yer değişiminin hız bileşenlerini şiddet değişmezliği kabulü ile aşağıdaki, Şekil 2.4’de hız vektörleri ile gösterilmiştir.



Şekil 2.5. Bir konumdan, diğer konuma hareket eden noktanın zamana bağlı olarak şiddet değişmezliği kabulü ile hız bileşenlerinin gösterilmesi

Denklem (2.4)’daki $I(x, y, t)$ için 1. dereceden Taylor serileri açılımını kullanarak aşağıdaki denklemi (2.4) elde edilmiştir.

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{dI}{dx} dx + \frac{dI}{dy} dy + \frac{dI}{dt} dt + YDT \quad (2.4)$$

Denklem (2.4)’de Y.D.T. ile ifade edilen *Yüksek Dereceden Terimlerin* kısaltılmasını ifade etmektedir. Bu YDT’ler Taylor Serileri açılımı ile elde edilir. YTD’ler küçük olduğu için hesaplamayı etkilemeyeceğinden göz ardı edilerek ihmal edilir. Denklem (2.4) kullanılarak aşağıdaki denklem (2.5) elde edilmiştir. $t + 1$ anındaki şiddet değişmezliği kabulü ile, belirli bir $r = (x, y)$ noktasındaki görüntüsünün parlaklığı $I(x + dx, y + dy, t + dt)$ ile ifade edilmektedir.

$$I(x+dx, y+dy, t+dt) = I(x, y, t) \quad (2.5)$$

$$\frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt = 0 \quad (2.6)$$

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} \frac{dt}{dt} = 0 \quad (2.7)$$

Buradan $v_x = \frac{dx}{dt}$ ve $v_y = \frac{dy}{dt}$ şeklinde ifade edilir. Denklem (2.7)'den eşitlik (2.8) elde edilir.

$$\frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} = 0 \quad (2.8)$$

Gerekli sadeleştirme işlemlerinden sonra

$$\frac{\partial I(x, y)}{\partial t} = - \left(\frac{dx}{dt} \frac{dy}{dt} \right) \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} \quad (2.9)$$

Kısmi türevler $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ ve $I_t = \frac{\partial I}{\partial t}$ şeklinde ifade edilir.

$$I_t = -(v_x, v_y) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \quad (2.10)$$

OA'nın hesaplanmasında (x ve y)'nin hız bileşenleri olan (v_x, v_y) ile zamana göre şiddet türevleri (I_x, I_y, I_t) arasındaki ilişki kısaca tekrar denklem (2.11)'deki gibi yazılabilir.

$$(I_x, I_y) \cdot (v_x, v_y) = -I_t \quad (2.11)$$

veya

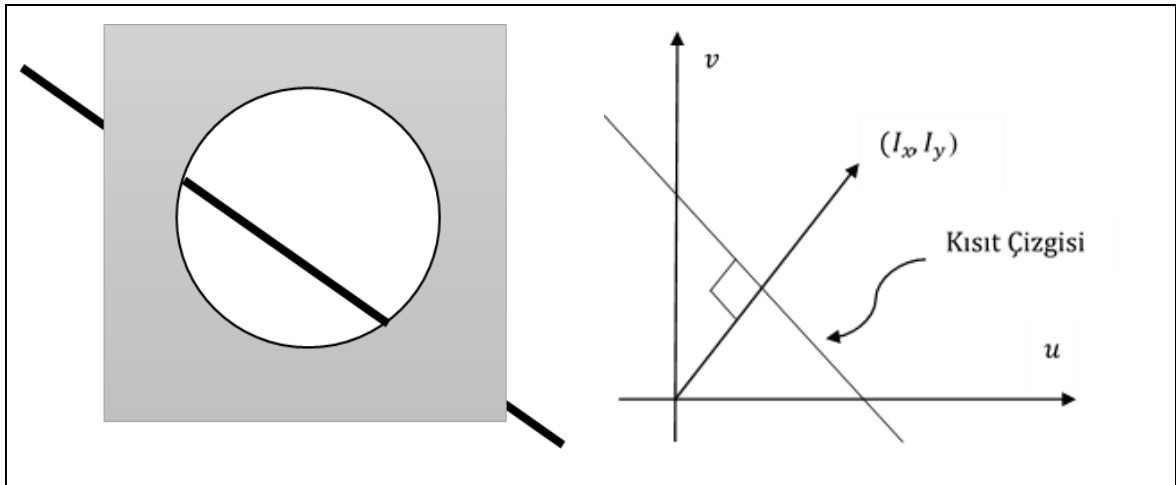
$$\nabla I \cdot \vec{v} = -I_t \quad (2.12)$$

Denklem (2.12)'de, $r = (x, y)$ pikselinin t anındaki hızıdır veya bir başka deyişle OA'dır. Denklem (2.12) "*İki Boyutlu Hareket Bileşen Denklemi (İBHKD)*"dir. Bu denklem iki bilinmeyenli bir doğru denklemidir. İki bilinmeyenli bu denklemin çözümünden denklem (2.7) elde edilir. Doğrultusu bu doğru üzerinde olan bir (x, y) noktasından pek çok hız bileşeni elde edilebilir. Bu sebeple İBHKD'den elde edilen doğru hızın tam yerini bulmaya yeterli değildir. Bu probleme OA'ın "belirsizlik problemi" denir.

2.3.3. Açıklık ya da belirsizlik problemi (Aperture problem)

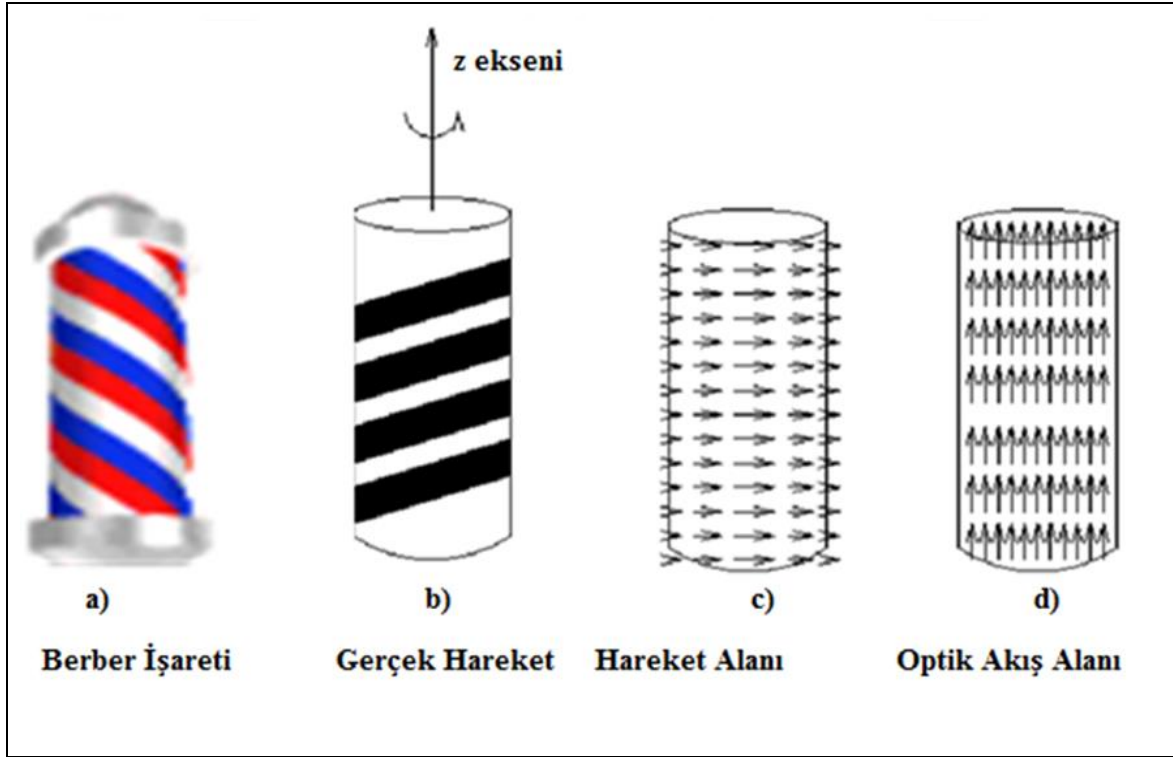
OA' komşu noktalardan bağımsız ek bileşenler olmadan görüntü üzerindeki bir noktadaki hızı hesaplayamaz. Çünkü görüntü üzerindeki bir noktadaki hızın iki bileşeni vardır. Ancak görüntü düzlemindeki bir noktada meydana gelen harekete bağlı görüntünün parlaklık değerlerindeki değişimin tek bileşeni vardır. Dolayısı ile OA'yı hesaplayabilmek için ek bileşenlere ihtiyaç vardır. İşte bu problem OA'nın *Belirsizlik ya da Açıklık Problemi (Aperture problem)* denir. Belirsizlik problemi OA'nın en önemli problemidir.

Şekil 2.6 a)'da hareketli bir çubuğun önünde ortası delikli bir karton olduğunda, kartonun arkasındaki çubuğun hareketi aşağı yukarı, sağa sola gibi birçok yöne doğru hareket edebilir. Eğer kartonun arkasında kalan çubukların uçları tamamen kartonun arkasında kalsa idi kartonun ortasındaki delik kısımdan görünen çubuğun farklı yönlerde olan fiziksel hareketleri tam olarak ayırt edilemezdi[36]. Şekil 2.5 b)'de hareketin doğruya dik bileşeni kısıt çizgisi ile bulunabilir.



Şekil 2.6. a) hareketli çubuğun ortası delik kartonda hareketi, b) parlaklık değişiminin kısıt çizgisi

Genelde sayısal görüntülerden elde edilen OA algoritmasından elde edilen hareket alanı sonucu ile gerçekteki hareket alanı aynıdır. Ancak bunun aynı olmadığı durumlara da rastlanır. Bu durumlar istisna olarak kabul edilerek dikkate alınmaz. Örneğin Şekil 2.6 a)'da kendi eksenini etrafında dönen ışıklı berber işaret direğini gösterir. Şekil 2.6 b) berber işaret direğinin dönüş eksenini ve yönünü gösterir. Şekil 2.6 c) gerçekte hareketin soldan sağa doğru olduğunu gösterir. Şekil 2.6 d) OA algoritmasının tespit ettiği optik akışı gösterir. Şekil 2.6 a)'da hareket eden berber işaret direğinin yaptığı hareket ile görülen farklıdır. Genel olarak, bu gibi durumlar olağandışıdır ve bu olağandışılıklar ihmal edilir ya da OA uygulamalarında hareket alanına karşılık geldiğini varsayılmaktadır.



Şekil 2.7. a) dönen ışıklı direk berber işareti, b) berber işareti hareket yönü, c) hareket alanı d) optik akış alanı

Ancak bunun aynı olmadığı durumlara belirsizlik problemi ile başa çıkabilmek için bazı varsayımlarda bulunulmuştur. Nesnelerin katı ve şekil değişiminin olmadığı varsayımı yapılmıştır. Örneğin Resim 2.2'de koşan bir köpek katı olmayan doku hareketine iyi bir örnektir. OA hesaplamasında esnek hareketli nesne hareketi tam olarak hesaplanamaz. Bu nedenle bazı varsayımlarda bulunulur. Böylelikle büyüme, küçülme, kırılmalar vb. bozucu etkiler dikkate alınmaz. Yani nesnelerin hareketleri süresince enerji ve şekillerinin değişmediği varsayımı yapılır.



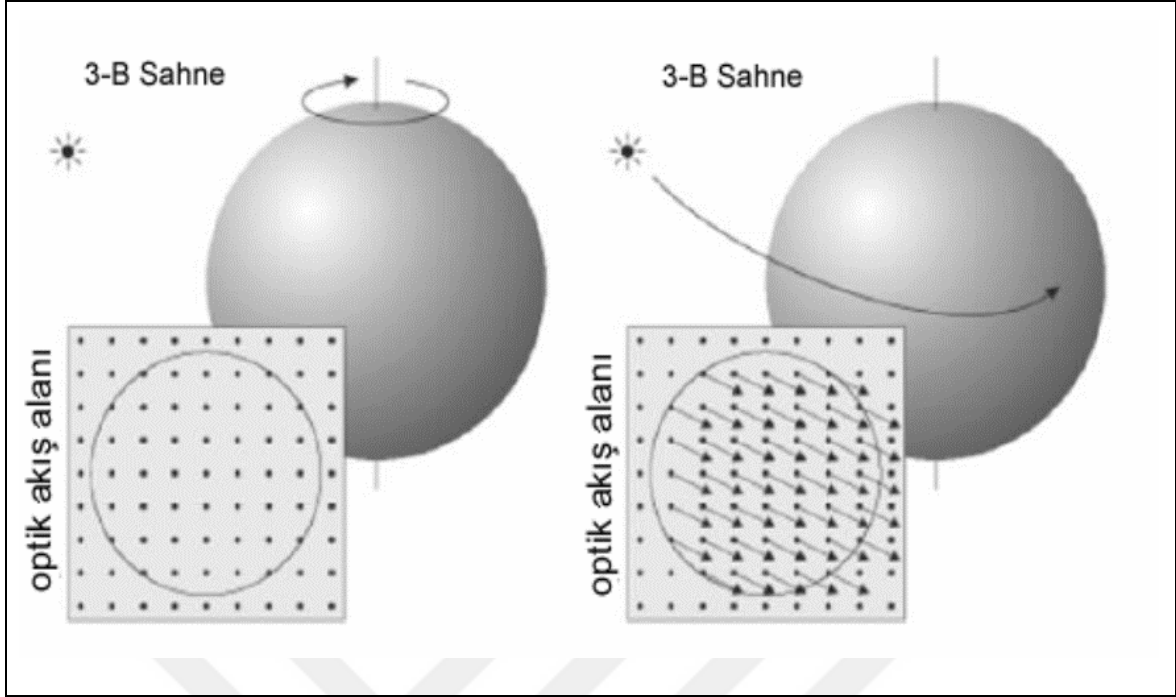
Resim 2.2. Katı olmayan doku hareketi

Optik akış hesabında karşılaşılan başka bir problem ise görüntü parlaklık sabitesinin bozulmasıdır, görüntüdeki gölgeler, lokal parlamalar, yerel şiddet düzgünlüğünü bozan yüzey saydamlığı yer almaktadır. Bunların yanı sıra, hareket eden veya sabit nesnelerin örtülmesi doğru tahmini engelleyecektir. Ayrıca bir ekranın veya yansıyan yüzeyin üzerinde hareket ve hız alanında farklılıklar görülebilecektir. Bu nedenle OA hesaplamalarında görüntü düzlemindeki OA ile gerçek nesnenin hızı arasında ilişki her zaman aynı olmayabilir. Örneğin; aşağıdaki Resim 2.3’de sabit bir ekran ve ekranda değişen görüntüler vardır. Ekran sabit olmasına rağmen ekrandaki hareketler OA algoritmasının çalışmasını olumsuz etkiler. Uygulamalarda bu olumsuzluklar yok sayılır.



Resim 2.3. Ekran sabir olmasına rağmen hareketi gösterir

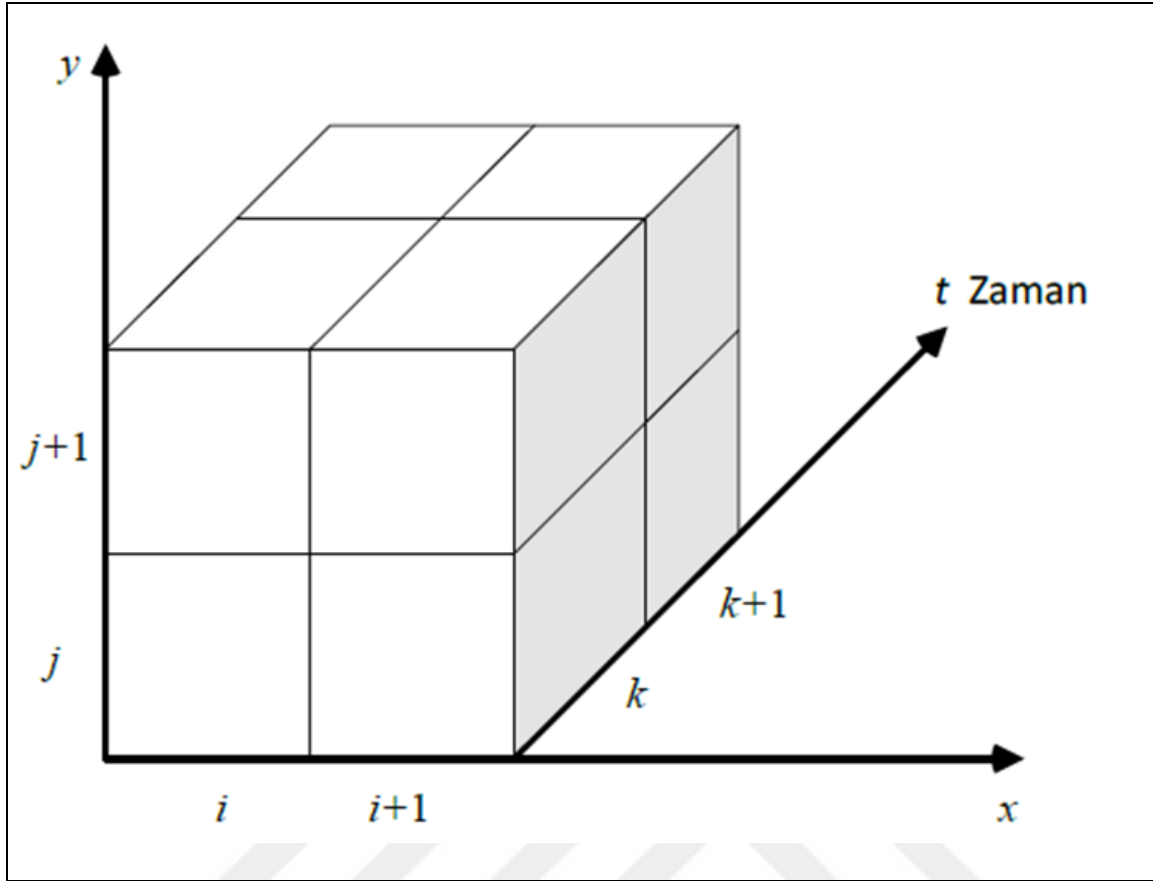
Sabit renk ve karakteristikte bir küre döndüğünde görünümünden kürenin dönüp dönmediği anlaşılamaz çünkü küre sabit renk ve karakteristiktir üzerinden yansıyan ışık ışınlarının parlaklığı 3B'li ortamda değişmeyecektir (Şekil2.7). Nesne 2B'li düzleme yansıtılınca ya da sayısal ortama alınınca da aynı gözükecektir. Bu durumda kürenin dönme hareketi OA'ile hesaplanamaz. Bu sebeple OA hesaplanırken parlaklığın sabit olduğu varsayımı yapılır. Aşağıdaki Şekil 2.7'de dönen Lambertian küresinin sabit ışık kaynağı altındaki hareketi ve hareketli ışık kaynağı altındaki hareketsiz durumda OA da meydana gelen değişimleri (a) sabit ışık kaynağı altında, kendi eksenini etrafında dönüyor. Burada OA alanı sıfır olurken gerçekte hareket alanı sıfır değildir. (b) hareketli ışık kaynağı altında sabit duran bir kürenin üzerindeki gölgelerde ve parlak kısımlarda değişimler oluyor, burada hareket alanı sıfır iken OA alanı sıfır değildir[37].



Şekil 2.8. a) Sabit ışık kaynağı altında, kendi eksenini etrafında hareket eden küre b) hareketli ışık kaynağı altında, sabit duran küre [33, 37]

2.4. Fark Teknikleri

OA'nın hesaplanmasında ardışık sayısal görüntülerin her birinin türevlerinin hesaplanması gerekmektedir. I_x, I_y ve I_t şiddet türevleri hesaplanır. Görüntü kareleri arasındaki piksellerdeki hareketlere karşılık gelen ışıklılık değerlerindeki değişimlerin yaklaşık farkının bulunması için sekiz ardışık ölçüme sahip bir küpün dört paralel kenarında bulunan değerlerin farklarının ortalamaları yaklaşık olarak tahmin edilir. Şekil 2.9'da x, y ve t yönlerindeki şiddet türevlerinin hesaplanmasında kullanılan sekiz ardışık değerlere sahip bir küpün dört paralel kenarında bulunan sekizli ardışık küp görülmektedir. Burada ki küpte i indeksi x yönünde sütun, j indeksi y yönünde satır ve k indeksi t yönünde zaman doğrusuna karşılık gelmektedir[29].



Şekil 2.9. x, y ve t yönlerindeki şiddet türevlerinin hesaplanmasında kullanılan küp

$$I_x \approx \frac{1}{4}(I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}) \quad (2.13)$$

$$I_y \approx \frac{1}{4}(I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}) \quad (2.14)$$

$$I_t \approx \frac{1}{4}(I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}) \quad (2.15)$$

Görüntünün x yönündeki türevinin maskesi denklem (2.16)'da y yönlerindeki şiddet türevinin maskesi denklem (2.17)'de ve t anındaki türevinin maskesi denklem (2.18)'de gösterilmiştir.

$$M_x = \frac{1}{4} \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \quad (2.16)$$

$$M_y = \frac{1}{4} \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix} \quad (2.17)$$

$$M_t = \frac{1}{4} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad (2.18)$$

Burada uzunluk birimi, ardışık sayısal görüntülerdeki piksellerin büyüklüğüdür; zaman birimi ise ardışık sayısal görüntüler arasındaki süredir.

2.4.1. Akış lablasyenleri hesabı

OA'da hız hesaplanırken u ve v hız bileşenlerinin Laplasyenlerini de tahmin edebilir. Denklem (2.19) ve (2.20)'de Laplasyenlerin hesaplanmasında kullanılan güvenilir bir yaklaşım yer almaktadır:

$$\nabla^2 u \approx \kappa(\bar{u}_{i,j,k} - u_{i,j,k}) \quad (2.19)$$

$$\nabla^2 v \approx \kappa(\bar{v}_{i,j,k} - v_{i,j,k}) \quad (2.20)$$

Denklem (2.19) ve (2.20)'de yer alan lokal ortalamalar olan, $\bar{u}$ ve $\bar{v}$ değerlerinin hesaplamaları aşağıda denklem (2.21) ve (2.22)'de belirtilmiştir.

$$\bar{u}_{i,j,k} = \frac{1}{6} \{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} + \frac{1}{12} \{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\} \quad (2.21)$$

$$\bar{v}_{i,j,k} = \frac{1}{6} \{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} + \frac{1}{12} \{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\} \quad (2.22)$$

Hız bileşenlerinin Laplasiyen hesabı, komşu piksellerin ağırlıklı ortalamalarından bir noktadaki değerin çıkarılması ile elde edilir. Çizelge 2.3.'de komşu piksellerin bölünebileceği uygun ağırlıkları gösteren maske verilmiştir.

Çizelge 2.2. Lablasyenlerin hesaplanmasında kullanılan maske

$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$
$\frac{1}{6}$	-1	$\frac{1}{6}$
$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$

2.4.2. Hataları minimize etme

Ardışık sayısal görüntüler üzerindeki bir noktanın $r = (x, y)$ parlaklığı diğer görüntülerde aynı olmasına rağmen gürültüler nedeni ile farklı olabilir. Bu nedenle Hareket Kısıtlama denklemi sıfıra eşit olmayabilir. Ayrıca, hız akışı tahmininde elde edilen hızlar gerçek hız değerinden farklı olabilir.

$$E^2 = \int_D (\nabla I \cdot \vec{v} + I_t)^2 + \alpha^2 \left[\left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 \right] dx dy \quad (2.23)$$

Denlem (2.23)'de D etki alanı olarak tanımlanmıştır. α : yumuşatma teriminin genliği olarak kullanılmıştır. OA'nın çözüm yöntemlerinden olan HS algoritmasında $\alpha = 0,5$ yerine $\alpha = 100$ önerilmektedir. Çünkü yapılan denemelerde $\alpha = 100$ en iyi sonucu vermiştir[38].

Hatalar iki şekilde ele alınabilir:

1. Görüntü parlaklığının değişim miktarı için denklemdaki hataların toplamını minimize etme:

$$E_b = I_x v_x + I_y v_y + I_t \quad (2.24)$$

2. Hız akışındaki düzgünlük için sapma tahminini minimize etme:

$$E_c^2 = (\bar{v}_x - v_x)^2 + (\bar{v}_y - v_y)^2 \quad (2.25)$$

Bu faktörleri birbirine bağlayan ağırlık ne olmalıdır? Pratikte imaj parlaklık hesabı, kuantalama hatası ve gürültü tarafından bozulacaktır ve bu yüzden E_b 'nin ideal olarak sıfır olması beklenmez. Bu miktar, gürültüye orantılı hata genliğine sahip olmaya eğilimlidir. Bu gerçek α^2 ile gösterilen uygun ağırlık faktörünü belirlemeye yardımcı olur.

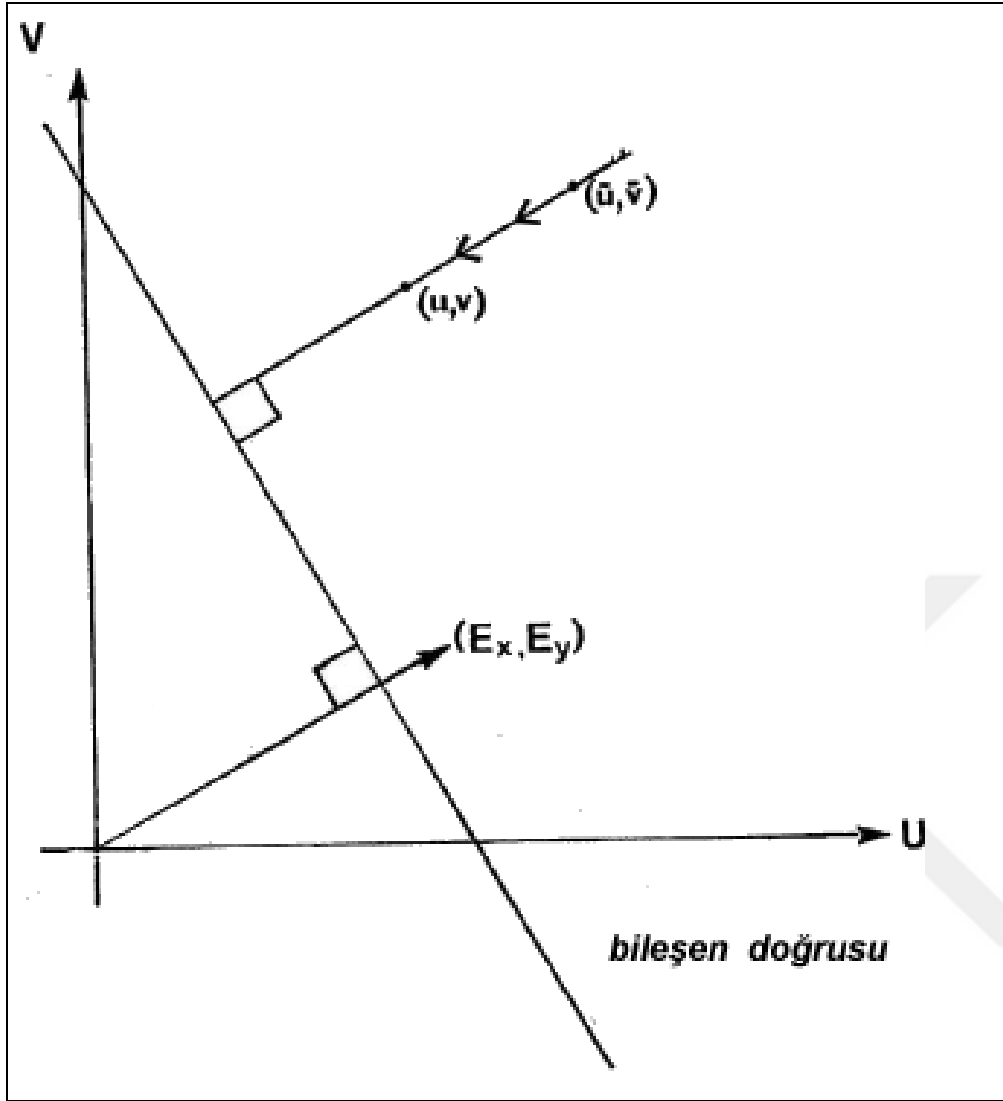
Minimize edilen toplam hata:

$$E^2 = \alpha^2 E_c^2 + E_b^2 \quad (2.26)$$

Hatanın u ve v 'ye göre türevini alıp denklemleri sıfırlarsak aşağıdaki sonuçları elde edilir.

$$\begin{aligned} (\alpha^2 + I_x^2 + I_y^2)(v_x - \bar{v}_x) &= -I_x[I_x \bar{v}_x + I_y \bar{v}_y + I_t] \\ (\alpha^2 + I_x^2 + I_y^2)(v_y - \bar{v}_y) &= -I_y[I_x \bar{v}_x + I_y \bar{v}_y + I_t] \end{aligned} \quad (2.27)$$

Bu, hatayı minimize eden akış hızı (u , v) değerinin, bileşen doğrusunu karşı sağ açılarda kesen doğru boyunca bileşen doğrusuna karşı gelen doğrultuda bulunduğunu gösterir. Bu ilişki genel olarak aşağıdaki şekilde gösterilmiştir. Sonuç olarak parlaklık gradyeninin küçük olduğu yerlerde, α^2 'nin önemli bir rolü olduğunu görebiliriz. Bu parametreler kabaca $I_x^2 + I_y^2$ 'nin tahmininde beklenen gürültü değerine eşittir.



Şekil 2.10. Hatayı minimize eden akış hızı değeri, bileşen doğrusuna dik olarak akış hızının lokal ortalamasından çizilen doğru

2.4.3. Yinelemeli çözüm (Iterative solution)

Ardışık iki görüntü karesi arasındaki OA hesaplamasında HS çözüm metodu uygulanırken Gauss-Jordan veya Gauss-Seidel yineleme yöntemlerinden biri kullanılabilir. Gauss-Jordan yineleme yöntemi görüntüdeki bir noktanın eşzamanlı olarak çözümü için kullanılması oldukça fazla zaman alabilir bunun yerine Gauss –Seidel tekrarlama yöntemi önerilir[29].

$$u^{(n+1)} = u^{(-n)} - I_x \frac{I_x [I_x u^{-n} + I_y v^{-n} + I_t]}{\alpha^2 + I_x^2 + I_y^2} \quad (2.28)$$

$$v^{(n+1)} = v^{(-n)} - I_y \frac{I_x [I_x u^{-n} + I_y v^{-n} + I_t]}{\alpha^2 + I_x^2 + I_y^2} \quad (2.29)$$

n : iterasyon sayısını belirtir. α düzgünlük faktörüdür ve pozitif bir sabittir. İki imaj veya video çerçevesi arasındaki ilişkili hareket büyükse, *düzgünlük faktörü* için büyük bir pozitif skalar değeri verilmelidir. Eğer ilişkili hareket küçükse, küçük bir pozitif skalar değeri verilmelidir. Uygulamaya en iyi uyan düzgünlük faktörünü bulmak için birden fazla değer denemesi gerekebilir.

$\bar{u}$ ve $\bar{v}$: u ve v 'nin ağırlıklandırılmış yerel ortalamalarını belirtir.

u^0 v^0 : başlangıç hız tahminlerini belirtir her ikisi de sıfıra eşitlenir.

u^{-n} v^{-n} u^n v^n komşuluk ortalamalarını belirtir.

Bu çalışma kapsamında yapılan tüm uygulamalarda işlemler 100 tekrarlamalı çözümlerle yapılmıştır.

Buradan elde edilen sonuçlardaki hız farkı belirli bir eşik değerinin altına indiğinde veya belirlenen bir tekrarlama sayısına ulaştığında tekrarlamanın durmasını sağlayabilir. Hem eşik değeri hem de tekrarlama sayısı kullanıldığı durumlarda ise bu koşullardan herhangi biri sağlandığında tekrarlama işlemi durdurulur. Eşik değeri, hız tahmininde kabul edilebilir sapma miktarının bir ölçüsüdür. Tekrarlama sayısı, hareket eden en büyük nesnenin kesit alanından daha büyük olmalıdır. Nesne kesitinin önceden bilinemediği durumlarda ise, tüm görüntünün kesitinin kullanılması uygun olacaktır. Birkaç farklı tekrarlama sayısı kullanılarak elde edilen sonuçlar karşılaştırıldığında, belli bir tekrarlama sayısından sonra hata değerinde küçük değişiklikler meydana geldiği görülmektedir. Bu karşılaştırmalar sayesinde daha doğru bir tekrarlama sayısı elde edilebilir.

3. DERİN ÖĞRENME

Bu tezde ardışık sayısal görüntülerdeki birden hareketli nesnelerin tanımlanması ve sınıflandırma işlemlerinde üstün özelliklerinden dolayı, yapay zekanın bir alt dalı olan derin öğrenme yöntemi kullanılmıştır. Derin öğrenme, insan algısına yakın bir doğruluk ile nesnelerin tanımlanmasını ve sınıflandırmasını yapabilmektedir. Görüntü işleme yapılırken bilgisayar grafiği, hesaplamalı fotoğrafçılık, görsel algılama, görüntü kodlama, sinyal işleme, istatistik bilgi teorisi, derin öğrenme, bilgisayarla görme, yapay zeka, makine öğrenmesi vb. gibi disiplinler arası çalışmalardan da faydalanılmaktadır.

Derin öğrenme konusunun anlaşılabilmesi için öncelikle öğrenmek, öğrenme, akıl ve zeka kavramlarının anlamları iyi anlaşılmalıdır. Özellikle zeka ve akıl kavramları birbirleri ile çok karıştırılmaktadır.

Öğrenmek, bilgi edinmek, bellemek, yetenek beceri kazanmak anlamlarına gelir[39]. Öğrenme, kişinin karşılaştığı bilindik durumlara ve sorunlara karşı geliştirmiş olduğu tepki ve davranış oluşturma, bunları farklı zaman ve benzer durumlara göre değiştirerek yenilerini geliştirebilme kabiliyetidir[40].

Öğrenme; tekrarlar ya da yaşantılar sonucunda davranışta gözlenen, uzun süreli kalıcı değişimlerdir. Bir bilgi veya becerinin öğrenilmiş bir davranış olabilmesi için öncelikle yaşantılar sonucunda meydana gelmelidir. Davranışta değişiklik meydana getirmelidir. Öğrenme, davranışta meydana gelen değişimler uzun süreli ve kalıcı olmalıdır. Yeni öğrenmeler sayesinde kişi gelişmektedir, daha önceden bilemediği, yapamadığı işlemleri bilebilir veya yapabilir hale gelir. Yani öğrenme sayesinde bir birey, içinde olduğu durumu veya ortamı algılar, edindiği izlenimler ve deneyimler sayesinde buna yeniden anlamlandırır, böylelikle içinde bulunduğu duruma ve ortama uygun sonuçlar üreterek buna kendisini alıştıırır.

Akıl, düşünme, anlama, kavrama, idrak etme, kara verme anlamlarına gelmektedir[41]. Akıl muhakeme ve bilgi elde etme gücü olarak ta tanımlanabilir. Genetik yolla gelen sevgi, korku, kıskançlık, doğal savunma güdülerinin yanı sıra yaşadığı çevre ile olan etkileşimlerden ve

toplumun şartlandırmalarının etkisi ile gelişen bir yetenektir[41]. Akıl, makineler tarafından, bilgisayar yazılımları ya da herhangi bir yolla taklit edilemez[41].

Zeka, insanın düşünme, akıl yürütme, objektif gerçekleri algılama, yargılama ve sonuç çıkarma yeteneklerinin tamamını içerir[42]. Zeka, belirli bir konuda çalışarak, öğretilerek eğitilerek elde edilen bilgi ve birikimlere dayalı yetenekler geliştirilebilir[41]. İlk kez karşılaşılan bir durum ya da ani olarak gerçekleşen bir olay karşısında anlama, öğrenme, analiz yeteneği vb. gibi özelliklerini kullanarak uyum sağlama olarak ta ifade edilebilir[41]. Yapay zeka, herhangi bir canlıdan faydalanılmadan, sadece yapay araçların kullanılması ile oluşturulan, görsel algı, konuşma tanıma, karar verme ve diller arası çeviri gibi normalde insan zekası gerektiren uyum sağlayabilme, anlama, öğrenme, analiz yeteneği, davranışlar, hareketler sergileyebilen, yazılımlar, yongalar veya makineler ile geliştirilmesinin genel adıdır. Satranç oynayabilen bilgisayarlar, kendi kendini sürebilen araçlar, borsalarda hisse senedi alım satımı yapabilen robotlar YZ temelli olarak geliştirilmiş teknolojilere örnekler olarak verilebilir. YZ nesnelerin evrimidir. Yapay sinir ağları (YSA), insan zekasının özelliği olan öğrenerek herhangi bir yardım almadan yeni bilgiler üretebilmek amacıyla oluşturulmuş sistemlerdir. YSA'lar, insan beynini biyolojik yapıları olan nöronların benzetimleri ile ortaya çıkmıştır. Bu benzetim modellerin yazılımlar ile bilgisayar sistemlerine uygulanmaya başlamasıyla birlikte pek çok alanda kullanılmaya başlanmıştır. Haykin, bir yapay sinir ağını, veri depolamasında kullanılması için doğal eğilimi olan basit birimlerden meydana gelen paralel olarak dağıtılmış bir işlemci olarak tarif eder[43]. Burada, YSA'ları veriyi alıp, depo eden ve kullanan fiziksel hücreli sistemler olarak ifade eder[44]. YZ, insan zekasını bilgisayarları kullanarak taklit eder, karmaşanın çok olduğu, bilgilerin az ya da ayrıntıların çok önemli olduğu, uzman eğitimi gerektiren uygulamalarda kullanılmaktadır. Bu nedenle endüstride her alanda kendisine yer bulmaktadır.

Makine Öğrenmesi (MÖ) 1949 yılında Donald Hebb tarafından yazılan “The Organization of Behavior” adlı kitapta makine öğrenmesi incelenmiş Pitts ve McCulloch tarafından 1943 yılında, önerilen bir sinir ağının ilk matematiksel modelinde öğrenmenin[45] nasıl olacağına çözüm getirmiştir[41, 46]. Hebb kuralı adı verilen bu öğrenme kuralını açıklamıştır. Bu kuralda, iki sinir arasındaki ağırlığın değeri, aynı zamanda sinirlerin etkinleştirilmesine bağlıdır. Hebb, sinir faaliyetlerini örnek alarak bunların hafızadaki basit bir yere yerleşebileceğini varsaymış ve bu kurama göre sinirlerin birbirlerini ortaklaşa uyardıklarını ve bu uyarım neticesinde aralarındaki sinaptik bağlantı ağırlıklarının kendi etkinlikleri

oranında artacağını ortaya koymuştur[41, 46]. YSA'ların temelini oluşturmuştur[46]. YSA'lar insan beyninin yapısını oluşturan nöronların çalışmasından esinlenilerek ortaya atılmış bir yöntemdir ve sinir hücrelerinin çalışması katmanlı ağ yapısının oluşturulmasında sinir hücrelerinin birbirleri ile çalışması örnek alınarak oluşturulmuştur.

MÖ, YZ'nin bir alt alanıdır. DÖ'de MÖ'nün bir alt alanıdır. YZ, MÖ ve DÖ arasındaki ilişkiler Şekil 3.1'de gösterilmiştir.



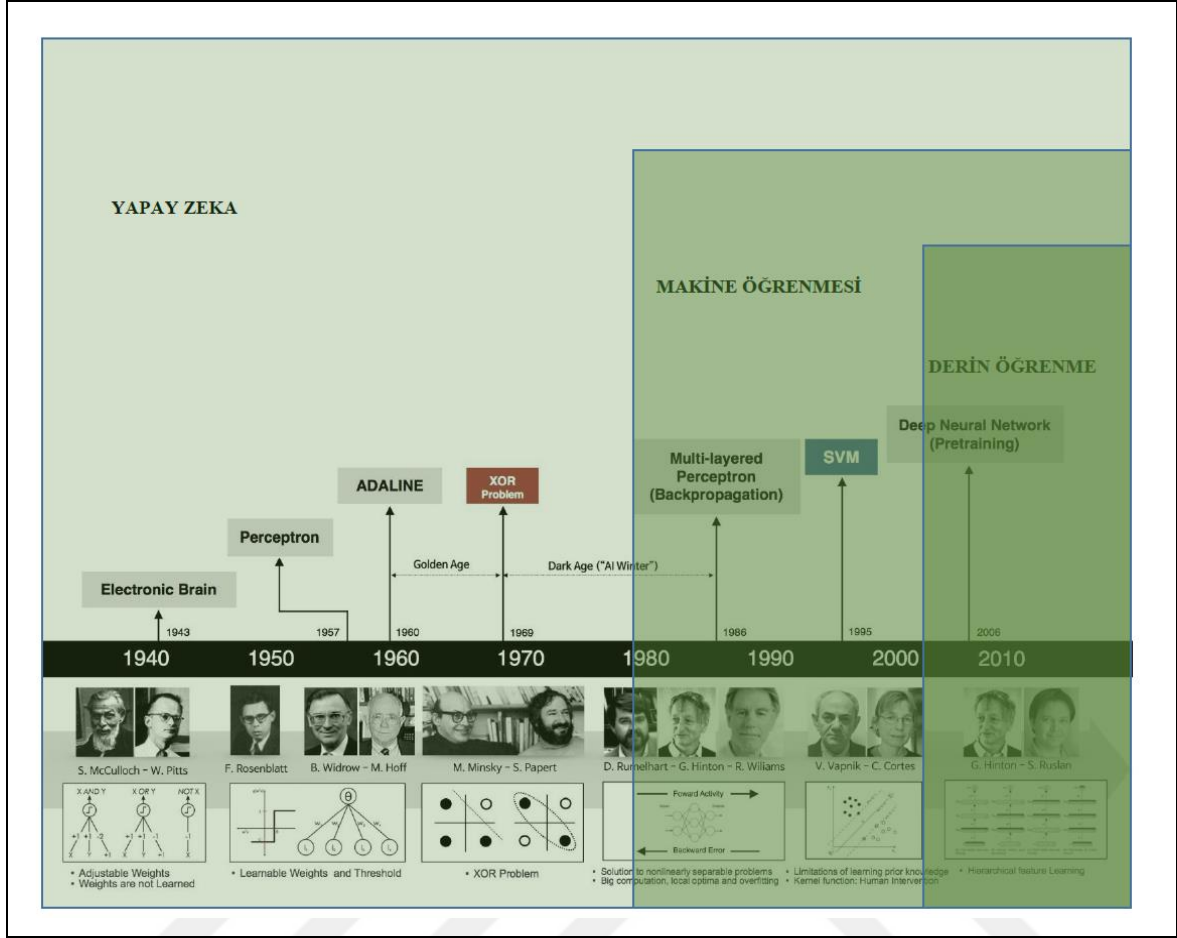
Şekil 3.1. Yapay zeka, makine öğrenmesi ve derin öğrenme ilişkileri

DÖ'de bir model, ile gelecekteki kararları ve tahminleri yapmak için tarihsel ve mevcut verileri kullanarak “eğitilir”. Yani model örneğin kediye ait 1000 adet görüntü kullanılarak kedinin özelliklerinin öğrenmesini sağlayarak eğitilir. Model eğitildikten sonra, eğitilen modelin yeni ve daha önceden görülmemiş kedi görüntülerine uygulandığında, görüntüde bir kedinin olup olmadığının tahminin sonucunu istatistiksel olarak gösterebilir. Derin Sinir Ağları (DSA) bilgisayarla görme, konuşma tanıma ve robotik gibi birçok yapay zeka uygulaması için yaygın olarak kullanılmaktadır[47].

Walter Pitts ve Warren McCulloch 1943 yılında, insan beyninin çalışma mantığından yola çıkarak bir sinir ağının ilk matematiksel modeli olan McCulloch-Pitts-Neuron yapay sinir

hücresini tasarladı[45, 48]. Alan Turing 1947 yılında “İstediğimiz şeyin deneyimlerden öğrenebilecekleri bir makine” olduğunu söylemiştir[49]. Alan Turing, 1950’de, genetik algoritmalara işaret eden “Turing Test” olarak adlandırılan bir makine önermiştir[50]. Arthur Samuel IBM’de Poughkeepsie Laboratuvarında, 1952 yılında IBM701 olarak tanıtılan bilgisayarda dama oyunu için program geliştirdi. Geliştirilen program her dama oynanmasında, hatalarını düzeltti, verileri daha iyi kazanma yollarını buldu. Bu bilgisayar programı YZ’nin ilk örneklerindendir[51]. Bir psikolog olan Rosenblatt, 1957’de “Biyolojik beynin algısal süreçlerine çok benzer optik, elektriksel veya tonal bilgi kalıpları arasındaki benzerlikleri veya kimlikleri tanımayı öğrenecek bir elektronik veya elektromekanik sistemi inşa edeceğini” açıkladı[52]. Rosenblatt, 1957’de farklı harfleri okuyup tanıyan bir YSA geliştirdi[52]. Bu derin sinir ağlarının (DSA) temeli olarak tanınmaktadır. Rosenblatt 1958 yılında ilk olarak perseptron kavramını kullanmıştır. Bir perseptron giriş hesaplama ve çıkış katmanlarından oluşur[53]. 1959’da David H. Hubel ve Torsten Wiesel, birincil görsel kortekste, basit hücreler ve kompleks hücreleri keşfettiler. Pek çok yapay sinir ağı (YSA), bu biyolojik gözlemlerden etkilenmiştir[54]. 1960 yılında Henry J. Sinir ağlarının eğitiminde kullanılan sürekli bir geri yayılım modelinin (hataların geriye doğru yayılması) temellerini açıklamıştır[55]. Ivakhnenko ve Lapa 1965 yılında, ilk çalışan DÖ ağlarını oluşturdular. Öğrenme algoritmaları, en iyi özellikleri bulmak ve sistemde iletirmek için her katmandaki istatistiksel yöntemleri kullanarak ileri beslemeli çok katmanlı algılayıcıları kullandı. Ivakhnenko 1971’de 8 katmanlı derin bir ağ oluşturabildi ve öğrenim sürecini Alpha adında bir bilgisayar tanımlama sisteminde başarıyla gösterdi[56]. Hubel ve Wiesel, hayvanlarda bulunan görsel korteksin davranışlarını taklit etmeye dayanan ilk evrimsel sinir ağlarının geliştirilmesine öncülük etmiştir[57]. Minimum miktarda ön işleme kullanmak için tasarlanmış çok katmanlı algılayıcıların varyasyonlarıdır[57]. Fukushima, 1980 yılında çok katmanlı yapay bir sinir ağı olan Neocognitron’u tanıtmıştır[58]. 1982’de Fukushima ve Miyake, görüntü tanıma için ESA’nın öncülü olarak düşünülebilecek bir sinir ağı modeli geliştirdi[59, 60]. Fukushima ve Miyake, Neocognitron, birçok hücre katmanının kademeli bağlantısından oluşan çok katmanlı bir ağ olduğundan ve hücreler arasındaki sinaptik bağlantıların verimlilikleri değiştirilebilir olduğundan bahsetmiştir[60]. Tekrarlayan sinir ağlarındaki olumsuzlukları gidermek için Jhon Hopfield 1982’de, Hopfield ağını tanıttı. Hopfield ağı, hücreler arası iletim düzenine sahip içerik adresleyebilen bir bellek sistemi olarak çalışabilen tekrarlayan sinir ağıdır[61]. Derin öğrenme kavramı ilk olarak 1986 yılında Rina Dechter tarafından yayınlanan “Learning While Searching In Constraint-Satisfaction-Problems” adlı makalede kullanılmıştır[62]. 1986 yılında Rumelhart, Hinton ve Williams,

geri yayılma sürecini daha ayrıntılı olarak anlattılar. Şekil tanıma, kelime tahmini gibi birçok işlem için mevcut sinir ağlarını nasıl geliştirebileceğini gösterdiler[63]. Sejnowski ve Rosenberg, 1987'te NETtalk programı, İngilizce kelimeleri bir çocuğun yaptığı gibi telaffuz etmeyi öğrendi ve metni konuşmaya dönüştürdü[64]. 1989 Watkins, doktora tezinde, makinelerde pekiştirici öğrenme tekniğinin uygulanabilirliğini gerçekleştiren Q-Öğrenme'yi tanıttı[65]. 1990 yılında Yann LeCun ve diğerleri DSA'ların ilk pratik uygulamasını yaptılar[66], el yazısını okumak için son yayılım teorileriyle birlikte evrimsel sinir ağlarını birleştirdi[66]. Makineler el yazısıyla yazılmış rakamları okudu[66]. Schmidhuber, 1993'te tekrarlayan sinir ağında 1.000'den fazla katmana sahip sinir ağını geliştirdi. Bu sinir ağlarının karmaşıklığı ve yeteneği ileriye doğru büyük bir sıçrama oldu[67]. 1997 yılında Schuster ve Paliwal, Çift Yönlü Tekrarlayan Sinir Ağını tanıttı[68]. 1998 yılında LeCun ve ark. Lenet-5 ağını tanıttı[69]. Krizhevsky ve ark. nesne tanımlama alanında, Büyük Ölçekli Görsel Tanıma Yarışmasını (ImageNet) 2012 yılında Evrimsel Sinir Ağı ile o zaman kadar ki %26,1 olan hata oranını %15,3 e kadar düşürdü[70]. 2014 yılında, Goodfellow ve ark. tarafından, Çekişmeli Üretici Ağlar (Generative adversarial networks - GAN) tanıtıldı. GAN'lar iki rakip ağı kullanır. İlki veriyi alır ve ayırt etmeden gerçeğe benzer örnekler oluşturmaya çalışır, ikincisi hem verileri hem de oluşturulan örnekleri alır ve her ilk ağda oluşturulan verilerin gerçek mi yoksa üretilip üretilmediğini belirlemeye çalışır. İlk ağ gerçek veriye erişemez. Eşzamanlı olarak ağlar birbirlerine karşı rekabet eder ve birbirlerini daha zeki, daha hızlı olmaları için çalışırlar[71].



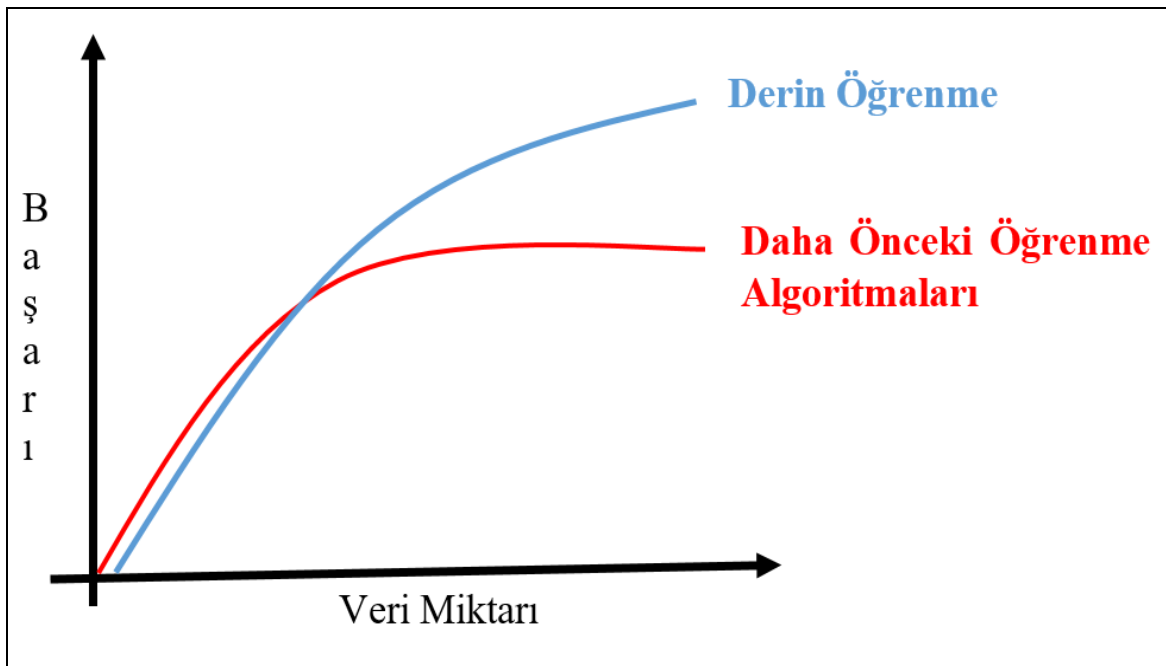
Şekil 3.2. Yapay zekanın gelişim sürecindeki kilometre taşları

Derin öğrenme, insan zekasının bilgisayarlara aktarılarak bilgisayarların veri setlerinden kendi kendine öğrenip daha sonra karşılaştığı veriler ile öğrendiklerini mukayese ederek çıkarım yapmasıdır. Bu konudaki çalışmalar artarak devam etmektedir. Bu amaçla çeşitli derin sinir ağı mimarileri geliştirilmiştir.

1. Evrişimli Sinir Ağları
2. Tekrarlayan Sinir Ağları
3. Uzun Kısa Süreli Hafıza Ağları
4. Sınırlı Boltzmann Makineleri
5. Geçitli Tekrarlayan Ağlar
6. Hopfield Ağları
7. Derin İnanç Ağları
8. Derin Oto-Kodlayıcılar

Bu tez çalışmasının amacı kapsamında yapılan çalışmalarda Evrişimli Sinir Ağları(ESA) kullanılmıştır. Bu nedenle ESA'lar ayrıntılı bir şekilde açıklanacaktır diğer DÖ mimarileri anlatılmayacaktır.

DÖ insan algısına yakın bir doğrulukta sonuçlar verebilmektedir. DÖ'nün bu başarısı kullanmış olduğu önceden eğitilmiş veri setlerine de bağlıdır. Bu nedenle DÖA'lar uygun şekilde eğitilmek üzere çok fazla veriye ihtiyaç duyar. Şekil 4.3'de DÖA'ların başarısı ve veri miktarı arasındaki ilişki gösterilmiştir.

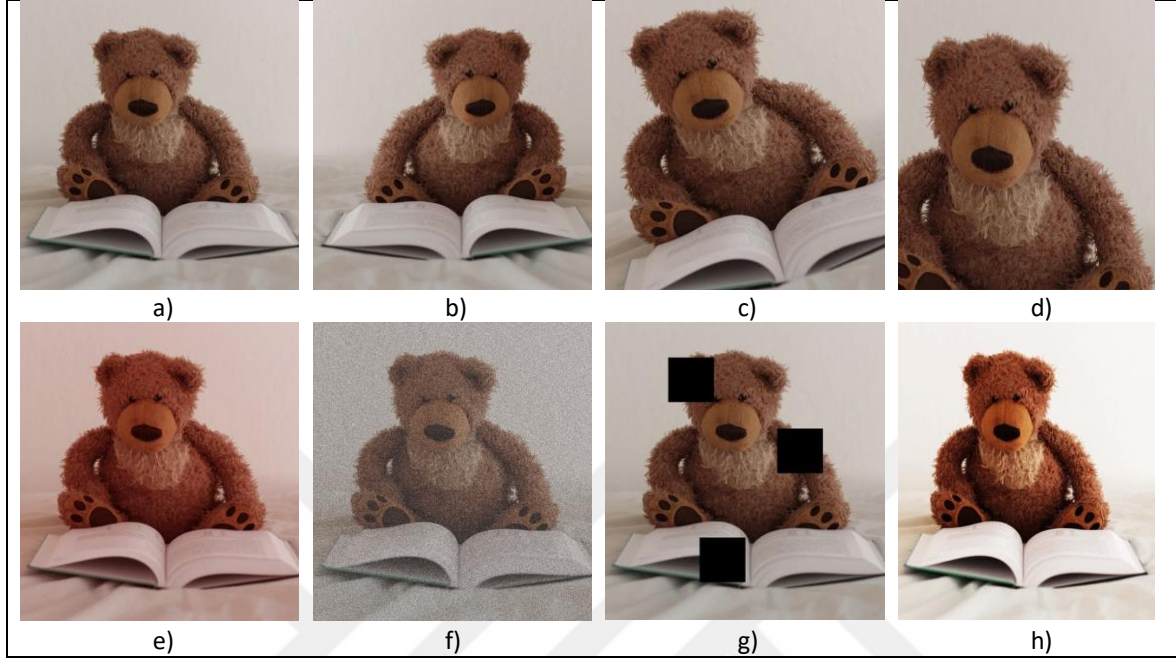


Şekil 3.3. Derin öğrenmenin başarısı ve veri miktarı arasındaki ilişki

Şekil 3.3'den de anlaşılacağı üzere veri miktarı arttıkça DÖA'ları amacı kapsamında eğitilen veri setlerinde eğitilen görüntüler yani eğitim verileri ne kadar çok ise veri setinin doğruluğu da o kadar çok olacaktır. Ancak eğitilecek çok sayıda veri bulmak zor olabilir bazı durumlarda da mümkün olmayabilir. Bu nedenle veri setinde kullanılan veriler eğitilirken veri artırma teknikleri kullanılarak mevcut verilerden yeni veriler elde edilerek bu veriler eğitilir. Böylelikle eğitim için kullanılan veriler artırılmış olur.

Bu nedenle mevcut verilerden daha fazla veri üretmek için çeşitli görüntü işleme teknikleri kullanılır. Kullanılan bu veri arttırma teknikleri aşağıdaki Şekil 3.4'de gösterilmiştir. Bu

teknikler ile görüntü verilerindeki temel bilgiler korunarak mevcut veri çeşitlendirmeye çalışır.



Resim 3.1. a) Normal, b) Aynalama, c) Döndürme, d) Kırpma, e) Renk değiştirme, f) Gürültü ekleme, g) Bilgi kaybı, h) Renk zıtlığı

Yukarıdaki Şekil 3.4 a)'da herhangi bir değişiklik yapılmamış normal görüntü, Şekil 3.4 b)'de asıl görüntünün korunduğu aynalanmış görüntü, bu görüntüde görüntü verisi korunarak görüntü, bir eksene göre çevrilir. Şekil 3.4 c)'de normal görüntünün belli bir açı ile döndürülmesidir, yanlış açılı görüntülerde ya da farklı açılarda bulunan nesnelerin tanımlanmasında hatayı azaltır. Şekil 3.4 d)'de kırpma işlemi ile görüntünün bir bölümüne rastgele odaklanma sağlanır böylelikle tanımlanması istenen görüntülerde nesnenin bir kısmının olmadığı durumlarda nesnelerin tanınmasını yapabilir, bazı uygulamalarda birkaç kez ardışık olarak rasgele kesme yapılabilir. Şekil 3.4 e)'de Renkli görüntülerde renklerin makul bir seviyede değiştirilmesi, farklı aydınlatma koşullarındaki olumsuzlukları giderir. Şekil 3.4 f)'de Gürültü ekleme, çevresel etkilerden kaynaklanan olumsuzlukları giderir. Şekil 3.4 g)'de görüntülerin belli alanları yok sayılır tanımlanması istenen görüntülerde olmayan kısımları ya da çeşitli sebeplerden dolayı görünmeyen hataları yok eder. Şekil 3.4 h)'de görüntünün ışıklılık değerleri değişimi kontrolü gibi teknikler ile normal görüntülerde meydana gelen ışık kaynağı değişimleri de yok eder.

Burada yapılan çeşitli görüntü işleme yöntemleri ile mevcut görüntülerin çoğaltılarak çeşitlendirmesi sağlanır bu sayede sınırlı sayıdaki görüntü verisi çoğaltılarak eğitim işlemi güçlendirilerek DÖA'ların tanımlama ve sınıflandırma başarısı arttırılmış olur.

3.1. Evrişimli Sinir Ağları (Convolutional Neural Network)

Bilgisayarlara, sayısal görüntülerdeki nesneleri nasıl tanıyacağını öğretmek için YSA'ların çeşitli yapıları kullanılır. Bu tez çalışmasının amacı kapsamında yapılan çalışmalarda da ardışık sayısal görüntülerden aynı anda birden çok hareketli nesnelerin gerçek zamanlı tespiti, tanımlanması ve sınıflandırılmasında görsel bilginin analizinde de günümüzde yaygın bir şekilde kullanılan YZ'nin bir alt dalı olan DÖ mimarilerinden olan Evrişimli Sinir Ağları(ESA) kullanılmıştır. ESA'lar sayısal bir görüntüdeki nesnelerin tanımlamasını ve sınıflandırılmasını yapabilen yapay zekanın bir alt dalı olan derin öğrenme algoritmasıdır. ESA'lar, görüntü tanıma ve bilgisayarla görme işlemlerinde uzmanlaşmış bir sinir ağıdır. ESA'lar çok katmanlı bir yapıya sahiptir. Geleneksel çok katmanlı algılayıcı mimarilerinin aksine, bir görüntüyü temel özelliklerine indirgemek için 'evrişim' ve 'havuzlama' adı verilen işlemler görüntüyü anlamak ve sınıflandırmak için kullanır[72]. ESA'lar adını en önemli işlemler basamaklarından biri olan ve ilk işlem basamağı olan evrişim katmanından alır.

ESA'lar, nöronlar arasındaki bağlantı modelinin hayvan görsel korteksinin yapısından ilham alınarak geliştirilen bir tür ileri beslemeli YSA'dır[73-76]. 1962 yılında Hubel ve Wiesel, kedilerin birincil görsel korteksinde iki sinir düğümü keşfettiler, Hubel ve Wiesel, bu sinir hücrelerinin de birkaç basit hücreden veri aldığı sonucuna vardı[77]. 1980'de Fukushima, Hubel ve Wiesel'in bulgularını görsel sistemin işleyen bir modeline dönüştürdü. Neocognitron adı verilen, iki ana hücre tipi içeren bu model, modern ESA'ların öncüsüdür[58, 75]. Ancak LeCun ve arkadaşlarının 1989 yılında yayınladıkları "Geri Yayılımlı El Yazısından Posta Kodu Tanıma" adlı makalesi ESA'ların temellerini oluşturur[78]. LeCun ve ark. bu çalışmalarında geri yayılım algoritması kullanarak denetimle eğitilmiş küçük ESA ile el yazısıyla yazılmış rakamların sınıflandırmasını yaptılar[78]. 1998 yılında LeCun ve ark. Lenet-5 ağını tanıttı[69]. Lenet-5 ağı 7 katmanlı ileri beslemeli bir ESA'dır. Lenet-5 ağının üç evrişim katmanı, iki havuzlama katmanı, bir tam bağlı katmanı ve bir tanede çıkış katmanı vardır[69]. LeCun ve ark. el yazısı karakter tanıma ile ilgili çeşitli yöntemleri gözden geçirdiler ve el yazısıyla yazılmış çeklerin

üzerindeki yazıları okuyabilen YSA'ların pratik uygulamasını ESA'larla yaptılar[69]. Paralel işlemci teknolojisinin gelişimi ile birlikte ESA'lar 2012 yılında büyük bir sıçrama yapmıştır. 2012 yılında düzenlenen IMAGE-NET yarışmasında Krizhevsky, ve diğerlerinin ImageNet Classification with Deep Convolutional Neural Networks adlı çalışmalarında elde ettikleri başarıdan sonra yaygın bir şekilde kullanılmaya başlamıştır[79]. ESA'lar büyük ölçekli dağıtılmış yüksek performanslı bilgi işleme kapasitelerine sahip GPU'ların geliştirilmesi ve ImageNet gibi görsel nesne tanıma projeleri sayesinde görüntü ve video tanımda son zamanlarda büyük bir başarı elde etti[80].

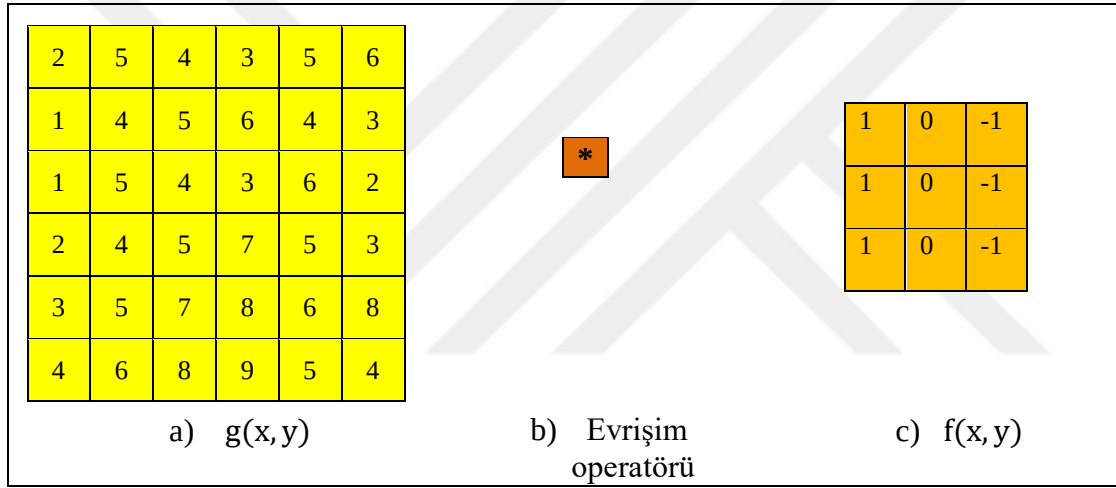
ESA'lar, bir görüntünün daha küçük ayrıntılardan veya özelliklerden oluştuğu gerçeğini kullanır. Sayısal görüntünün tüm özelliklerini tek başına analiz etmek her bir pikseli kullanan ağ yapısı ile görüntü hakkında bir bütün olarak bir karar verir. Dolayısıyla 8K (7680x4320x3) çözünürlüğe sahip renkli bir görüntünün hesaplama açısından sahip olduğu veri miktarının büyüklüğüyle işlem miktarını da o kadar arttıracaktır. ESA'nın rolü, iyi bir tahmin elde etmek için kritik olan bilgi kaybına neden olmadan sayısal görüntüleri evrişim yoluyla daha kolay bir forma getirerek işlemektir[81]. Böylelikle sayısal görüntülerdeki özellikleri öğrenmekle kalmayıp, aynı anda büyük veri kümelerinin ölçeklenebilir mimarisini tasarlarlarken önemli rol oynar[81]. ESA'lar ismini ilk katmanı olan evrişim işleminden almışlardır.

3.1.1. Evrişim katmanı (Convolution layer)

Bu katmanda evrişim işlemi ile giriş görüntüsünün belirli özellikleri vurgulanır, diğer özellikleri kaldırılır, değiştirilir veya geliştirilerek görüntüye ait tüm öz nitelikleri çıkartılarak görüntüden değerli bilgiler elde edilir. Evrişim işlemiyle sayısal görüntülerin, piksellerinin komşuluk değerlerinde yumuşatma, keskinleştirme, kenar iyileştirme, vb. gibi bazı özelliklerin öne çıkarılarak belirlendiği komşuluk işlemidir. Evrişim işlemi ilk olarak LeCun ve diğerlerinin posta kutularındaki el yazısı tanıma çalışmasında 5x5 boyutlarında maske kullanılarak yapılmıştır[78]. Evrişim katmanında giriş görüntüsü, filtre/maske/çekirdek(kernel)'ler ile çarpılır. Filtreler aynı zamanda birer sinyaldir. Filtreler iki boyutlu bir kare matris ile ifade edilir. Filtreler genellikle 1x1, 3x3, 5x5, 7x7, 9x9, 11x11 boyutlarında tek sayılardan oluşan kare matrisleridir, çünkü sonuç değeri maskenin ortasındaki pikselin değeri ile ifade edilir[82]. Evrişim işlemi matematiksel olarak denklem (3.1)'deki gibi ifade edilir.

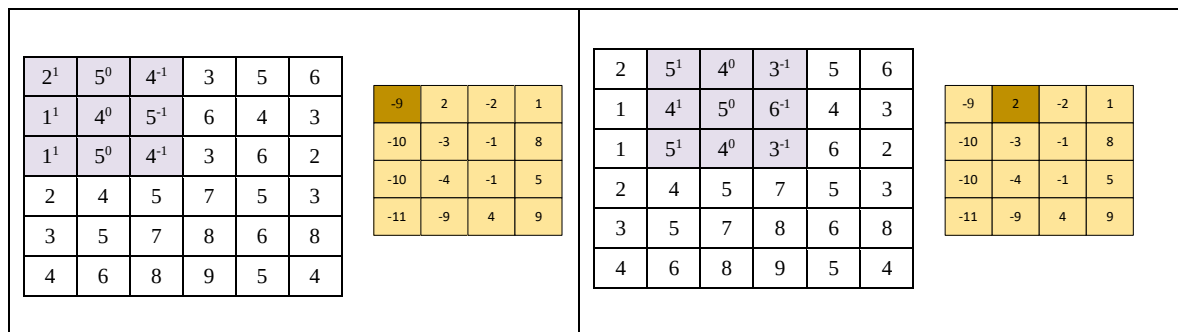
$$h(x, y) = g(x, y) * f(x, y) = \sum_m \sum_n g(x, y) f(x-m, y-n) \quad (3.1)$$

Denklem (3.1)'de $h(x, y)$ evrişim işleminin sonucu, $g(x, y)$ giriş görüntüsü, $f(x, y)$ maske ya da diğer adı ile filtre, evrişim operatörü ise $(*)$ dir. ESA'larda evrişim katmanında 2B'li giriş görüntüsü ile bir filtrenin işleme sokulması, giriş görüntüsü ile fitrenin ilk pikselden başlanarak eleman elemana çarpılarak çarpımların toplamı olarak ifade edilir. Evrişim işlemi ile giriş görüntüsünü bulanıklaştırma, keskinleştirme, kenar algılama, gürültü azaltma vb. gibi çeşitli görüntüde değişiklikler yapılır. Aşağıdaki Şekil 3.4'de evrişim işleminin nasıl yapıldığı giriş görüntüsü, evrişim ve filtre operatörü ile nasıl yapıldığı gösterilmiştir.



Şekil 3.4. a) giriş görüntüsü matrisi, b) evrişim operatörü ve c) filtre

Aşağıdaki Şekil 3.5'de 6x6 boyutlarında bir giriş görüntü matrisi ile 3x3 boyutlarında ki bir filtre ile evrişim işleminin 1 adım aralığında kaydırılarak giriş görüntüsüne evrişim işleminin nasıl yapıldığı adım adım gösterilmiştir.



Şekil 3.5. Sayısal görüntüye filtre uygulanması

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

Şekil 3.5. (devam) Sayısal görüntüye filtre uygulanması

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

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

-9	2	-2	1
-10	-3	-1	8
-10	-4	-1	5
-11	-9	4	9

Şekil 3.5. (devam) Sayısal görüntüye filtre uygulanması

6x6 boyutlarındaki giriş görüntüsü üzerinde, 3x3 boyutlarındaki filtre uygulanırken 3x3 boyutlarındaki filtre satır boyunca bir piksel aralıklarla kaydırılır. Kaydırma aralığına adım(stride) denir. Uygulamalarda genellikle bir adım kaydırma işlem yapılır. Satır bitiminde bir alt satırın başından işleme devam edilir. Bu şekilde devam edilerek işlem tamamlanır. Elde edilen sonuç evrişmiş matristir.

Çizelge 3.1. 6x6 boyutlarındaki matris ile 4x4 boyutlarındaki matrisin evrişim sonucu

$$\begin{aligned}
h(1,1) &= (2 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = -9 \\
h(1,2) &= (5 * 1) + (4 * 0) + (3 * -1) + (4 * 1) + (5 * 0) + (6 * -1) + (5 * 1) + (4 * 0) + (3 * -1) = 2 \\
h(1,3) &= (4 * 1) + (3 * 0) + (5 * -1) + (5 * 1) + (6 * 0) + (4 * -1) + (4 * 1) + (3 * 0) + (6 * -1) = -2 \\
h(1,4) &= (3 * 1) + (5 * 0) + (6 * -1) + (6 * 1) + (4 * 0) + (3 * -1) + (3 * 1) + (6 * 0) + (2 * -1) = 1 \\
\\
h(2,1) &= (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) + (2 * 1) + (4 * 0) + (5 * -1) = -10 \\
h(2,2) &= (4 * 1) + (5 * 0) + (6 * -1) + (5 * 1) + (4 * 0) + (3 * -1) + (4 * 1) + (5 * 0) + (7 * -1) = -3 \\
h(2,3) &= (5 * 1) + (6 * 0) + (4 * -1) + (4 * 1) + (3 * 0) + (6 * -1) + (5 * 1) + (7 * 0) + (5 * -1) = -1 \\
h(2,4) &= (6 * 1) + (4 * 0) + (3 * -1) + (3 * 1) + (6 * 0) + (2 * -1) + (7 * 1) + (5 * 0) + (3 * -1) = 8 \\
\\
h(3,1) &= (1 * 1) + (5 * 0) + (4 * -1) + (2 * 1) + (4 * 0) + (5 * -1) + (3 * 1) + (5 * 0) + (7 * -1) = -10 \\
h(3,2) &= (5 * 1) + (4 * 0) + (3 * -1) + (4 * 1) + (5 * 0) + (7 * -1) + (5 * 1) + (7 * 0) + (8 * -1) = -4 \\
h(3,3) &= (4 * 1) + (3 * 0) + (6 * -1) + (5 * 1) + (7 * 0) + (5 * -1) + (7 * 1) + (8 * 0) + (6 * -1) = -1 \\
h(3,4) &= (1 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = 5 \\
\\
h(4,1) &= (1 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = -11 \\
h(4,2) &= (1 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = -9 \\
h(4,3) &= (1 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = 4 \\
h(4,4) &= (1 * 1) + (5 * 0) + (4 * -1) + (1 * 1) + (4 * 0) + (5 * -1) + (1 * 1) + (5 * 0) + (4 * -1) = 9
\end{aligned}$$

Çizelge 3.1’de 6x6 boyutlarındaki giriş görüntüsüne bir adım aralıkla 3x3 boyutlarındaki bir filtre ile bir adım kaydırılarak evrişim işleminin yapıldığı matematiksel işlem gösterilmiştir. Evrişim işleminin sonucunda 4x4 boyutlarında sonuç görüntüsünü oluşturan evrişmiş matris elde edilir. Elde edilen 4x4 boyutlarındaki sonuç matrisi görüntüye ait özelliklerin çıkartıldığı öznitelik haritasıdır. Çeşitli filtreler kullanılarak elde edilen öznitelik haritaları görüntülere ait farklı detayları ön plana çıkartır. Bu nedenle ESA’larda görüntüye ait farklı detayları yakalamak için her evrişim katmanlarında giriş görüntüsü çok sayıda filtreleme işlemi uygulanır. Böylelikle her evrişim katmanında görüntünün farklı farklı özellikleri ön plana çıkartılarak nesnelere ait detay bilgileri elde edilir.

Evrişim işlemi sonucunda oluşan öznitelik haritalarının boyutları ile giriş görüntüsünün boyutları arasında farklılıklar olur. Giriş görüntüsünün satır ve sütun sayılarını korumak için kenarlara piksel ekleme yani dolgu(padding) işlemi uygulanır. Örneğin; giriş görüntüsü satır sayısı m sütun sayısı n ile ifade edilirse 6x6 boyutlarındaki giriş görüntüsü için $m = 6$ ve $n = 6$ ’dır. 3x3 boyutlarında bir filtre ile evrişim işlemi yapıldığında sonuç görüntüsü 4x4 boyutlarında olur.

Giriş görüntüsü ile evrişim işlemi sonucunda elde edilen özellik haritası çıkartılmış görüntünün satır ve sütun sayılarındaki farklılıklar olur. Giriş görüntüsü işe evrişim işlemi sonucunda elde edilen görüntünün boyutlarını eşitlemek için özellik haritası çıkartılmış sonuç görüntünün kenarlarına ekleme yani dolgu(padding) işlemi yapılır. Sıfır dolgu işlemi Şekil 3.7’de gösterilmiştir. Ekleme işlemi p ile gösterilir. Kaç piksel ekleme yapılması gerektiği denklem (3.2)’deki formül ile hesaplanır.

$$p = (f - 1) / 2 \quad (3.2)$$

Evrişim işleminden sonra $p = (f - 1) / 2$ formülünden elde edilen sonuç giriş görüntüsünün çevresine eklenmesi gereken piksel sayısını verir. Denklem (3.3)’de dolgu işleminden sonra sonuç görüntüsünün boyutları gösterilmiştir.

$$h = (m + 2p - f + 1) \times (n + 2p - f + 1) \quad (3.3)$$

Kaydırma adımı(stride) s ile ifade edilir. Evrişim işlemi sonucunda oluşan özellik haritasını içeren sayısal görüntünün boyutu denklem (3.4) ile hesaplanır.

$$\left(\left(\frac{(m+2p-f)}{s} \right) + 1 \right) \times \left(\left(\frac{(n+2p-f)}{s} \right) + 1 \right) \quad (3.4)$$

Aşağıdaki Çizelge 3.2’de evrişim işleminden sonra giriş görüntüsünün çevresine sıfır dolgu işleminin yapıldığı görülmektedir.

Çizelge 3.2. Sıfır dolgu işlemi

0	0	0	0	0	0
0	-9	2	-2	1	0
0	-10	-3	-1	8	0
0	-10	-4	-1	5	0
0	-11	-9	4	9	0
0	0	0	0	0	0

3.1.2. Etkinleştirme katmanı (Activation layer)

ESA’larda evrişim işleminden sonra Etkinleştirme Katmanı gelir. Evrişim işleminde elde edilen sonuç etkinleştirme işlemi ile doğrusal hale getirilir. Gerçek dünyadan alınan görüntü, ses gibi veriler doğrusal değildir. Etkinleştirme işlemi yapılmadan YSA’larda yapılan öğrenme sınırlı bir öğrenme olacaktır. Bu nedenle evrişim işleminden sonra etkinleştirme işlemi yapılır[83]. Nesne tanıma ve sınıflandırmadan konuşma tanıma, bölümleme, sahne anlama ve tanımlamaya, makine çevirisi, testten konuşma sistemlerine, kanser tespit sistemlerine, parmak izi tespitine, hava tahminine, kendi kendine giden arabalara ve birkaçından bahsedilecek diğer alanlara kadar farklı alanlarda kullanılan YSA’ların ağırlıklı toplamalarının hesaplanmasında ve sapmasında kullanılır[84]. Etkinleştirme işlemi ile YSA’larda işlenen gerçek dünyaya ait sayısal görüntülerden daha başarılı öğrenme yapılır. Etkinleştirme katmanı için ihtiyaçlara göre Etkinleştirme Fonksiyonları(EF) geliştirilmiştir.

Basamak (Step) etkinleştirme fonksiyonu

ESA’larda etkinleştirme katmanında bir eşik değeri olarak iki değer üreten bir sınıflandırma çıktısı (0 ya da 1) verir. Denklem (3.5) ile ifade edilir. Genellikle YSA’larda iki çıkış üreten katmanlarında tercih edilirler.

$$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases} \quad (3.5)$$

Sigmoid etkinleştirme fonksiyonu

Sigmoid EF doğrusal olmayan sonuçlar üretir. Bu nedenle yaygın olarak kullanılan etkinleştirme fonksiyonlarından. Sigmoid fonksiyonu, (0,1) aralığında değerler üreten EF’dir denklem (3.6)’de gösterilmiştir.

$$f(x) = \left(\frac{1}{1 + \exp^{-x}} \right) \quad (3.6)$$

Tanjant hiperbolik (Tanh) etkinleştirme fonksiyonu

Tanh fonksiyonu sigmoid EF ile benzerdir. Tanh EF’si doğrusal olmayan bir EF’dir denklem (3.7)’de gösterilmiştir. Tanh EF’si (-1,1) aralığında değerler üretir[84]. Tanh EF’si çok katmanlı sinir ağları için daha iyi eğitim performansı sağlaması açısından sigmoid işlevine kıyasla tercih edilen EF’dir. Fonksiyonun sağladığı ana avantaj, sıfır merkezli çıktı üretmesi ve böylece geri yayılma sürecine yardımcı olmasıdır[84].

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right) \quad (3.7)$$

ReLU (Rectified Linear Unit) etkinleştirme fonksiyonu

Doğrultulmuş lineer birim (ReLU) fonksiyonu, doğrusal olmayan bir EF’dir. ReLU fonksiyonu negatif girdiler için 0 değerini üretir. Pozitif girdiler için (0,∞) aralığında

değerler ürettiği denklem (3.8)'de görülmektedir. Ağdaki negatif değerleri kullanamadığı için işlem yükünü azaltır, Bu nedenle çok katmanlı YSA'larda tercih edilen etkinleştirme fonksiyonudur[83, 85].

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \quad (3.8)$$

Softmax etkinleştirme fonksiyonu

Çoklu sınıflandırma problemleri için geliştirilmiş bir fonksiyondur. Softmax fonksiyonu verilen her giriş değerinin bir sınıfa ait olup olmama olasılığını gösteren, sigmoid fonksiyonu ile benzer (0,1) aralığında değerler üreten etkinleştirme fonksiyonudur.

$$f(x_i) = \left(\frac{\exp(x_i)}{\sum_j \exp(x_j)} \right) \quad (3.9)$$

Softplus etkinleştirme fonksiyonu

Sigmoid ve Tanh gibi geleneksel etkinleştirme fonksiyonlarına alternatif olarak sunulan bu fonksiyon ReLU EF'nin yumuşatılmış bir versiyonudur. Softplus EF (0, +∞) aralığında türevlenebilir bir çıktı üretmektedir.

$$f(x) = \log(1 + \exp^x) \quad (3.10)$$

ELU etkinleştirme fonksiyonu

Üstel lineer birim (Exponential Linear Unit-ELU), negatif girdiler hariç ReLU ile benzerdir. Negatif girdilerde ise genellikle 1.0 alınan alfa parametresi almaktadır[86].

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha \exp(x) - 1, & \text{if } x \leq 0 \end{cases} \quad (3.11)$$

$$f' = \begin{cases} 1, & \text{if } x > 0 \\ f(x) + \alpha, & \text{if } x \leq 0 \end{cases} \quad (3.12)$$

PReLU (Parametric Rectified Linear Unit) etkinleştirme fonksiyonu

Parametrik ReLU etkinleştirme fonksiyonu olarak adlandırılır. PReLU, neredeyse sıfır ekstra hesaplama maliyeti ve çok az aşırı uyum riski ile model uyumunu iyileştirir. İkinci olarak, özellikle doğrultucunun doğrusal olmayanlarını dikkate alan sağlam bir başlatma yöntemi türetir. PRelu etkinleştirme fonksiyonu, son derece derin düzeltilmiş modelleri doğrudan sıfırdan eğitmeyi ve daha derin veya daha geniş ağ mimarilerini araştırmayı sağlar[87]. PReLU EF denklem (3.13) ve denklem (3.14)'de negatif girdi değerleri için extra alfa sabiti ile verilen girdi değerlerinin çarpımlarının sonuçlarını çıktı olarak verildiği gösterilmiştir.

$$f(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ \alpha_i x_i, & \text{if } x_i \leq 0 \end{cases} \quad (3.13)$$

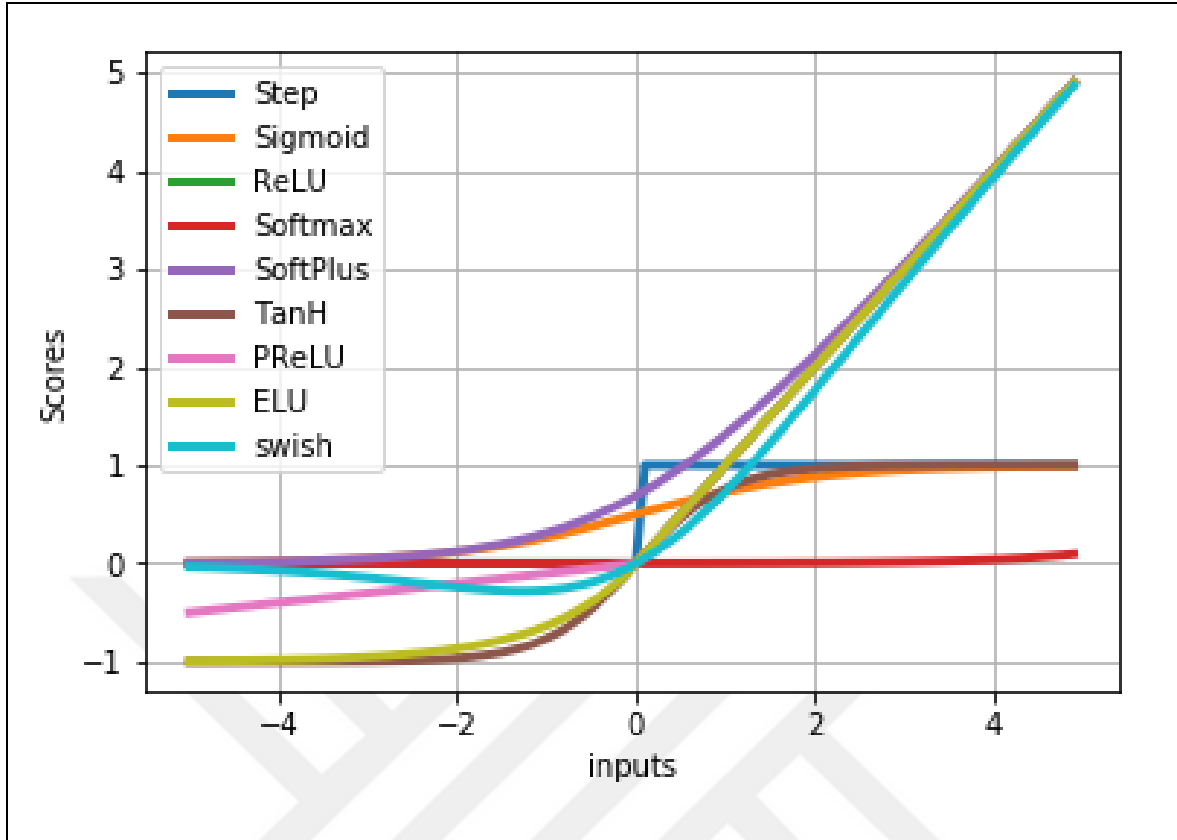
$$f(x_i) = \max(0, x_i) + \alpha_i \min(0, x_i) \quad (3.14)$$

Swish etkinleştirme fonksiyonu (swishEF)

Google araştırmacıları tarafından geliştirilmiştir[83]. SwishEF giriş ile sigmoid fonksiyonu çarpılır, $f(x) = x \cdot \text{sigmoid}(x)$ çarpımının sonucu SwishEF'nin sonucunu verdiği denklem (3.15)'de görülmektedir[83, 88].

$$f(x) = x \cdot \text{sigmoid}(x) = \frac{x}{1 + e^{-x}} \quad (3.15)$$

YSA'larda ihtiyaçları karşılamak için Etkinleştirme Katmanlarında kullanmak üzere çeşitli EF'ler geliştirilmiştir. Etkinleştirme katmanlarında sıklıkla kullanılan bazı EF'ler aşağıdaki Şekil 3.6'da grafik üzerinde çizdirilerek gösterilmiştir.

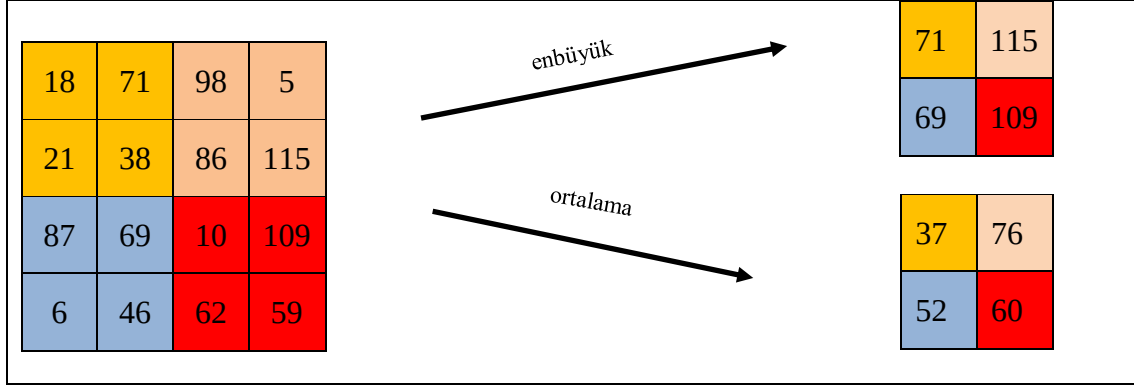


Şekil 3.6. Yapay sinir ağlarında sıklıkla kullanılan etkinleştirme fonksiyonları

3.1.3. Havuzlama katmanı (Pooling layer)

Havuzlama işleminde giriş bilgisinin kanal sayısı değiştirilmez, yükseklik ve genişlik bilgileri her bir özellik haritası için en önemli bilgileri koruyarak küçültülür. Enbüyük(maksimum), enküçük(minimum), ortalama(average) ve toplam(sum) gibi havuzlama türleri mevcuttur. Yapılan uygulamalarda enbüyüğü havuzlama işlemi daha iyi sonuçlar verdiği için daha çok tercih edilmektedir. Havuzlama işleminde giriş görüntüsünün boyutu önemli ölçüde azaltılmaktadır. En büyüğü havuzlama işleminde havuzlama yapılırken 2x2 boyutlarındaki bir matrisin içindeki enbüyük elaman alınır, ortalama havuzlama işleminde ise 2x2 matrisin içindeki değerlerin toplamının ortalaması alınır. Şekil 3.7’de 2x2 matris boyutlarında Enbüyüğü havuzlama ve ortalama havuzlama işlemleri gösterilmiştir. “Enbüyüğü havuzlama Gürültü Bastırıcı olarak da çalışır. Gürültülü etkinleştirmeleri tamamen ortadan kaldırır ve boyut azaltma ile birlikte gürültü gidermeyi de gerçekleştirir”[81]. “Öte yandan, Ortalama Havuzlama basitçe bir gürültü bastırma mekanizması olarak boyut azaltma gerçekleştirir. Dolayısıyla, Enbüyüğü Havuzlamanın Ortalama Havuzlamadan çok daha iyi performans gösterdiği söylenebilir”[81]. Havuzlama

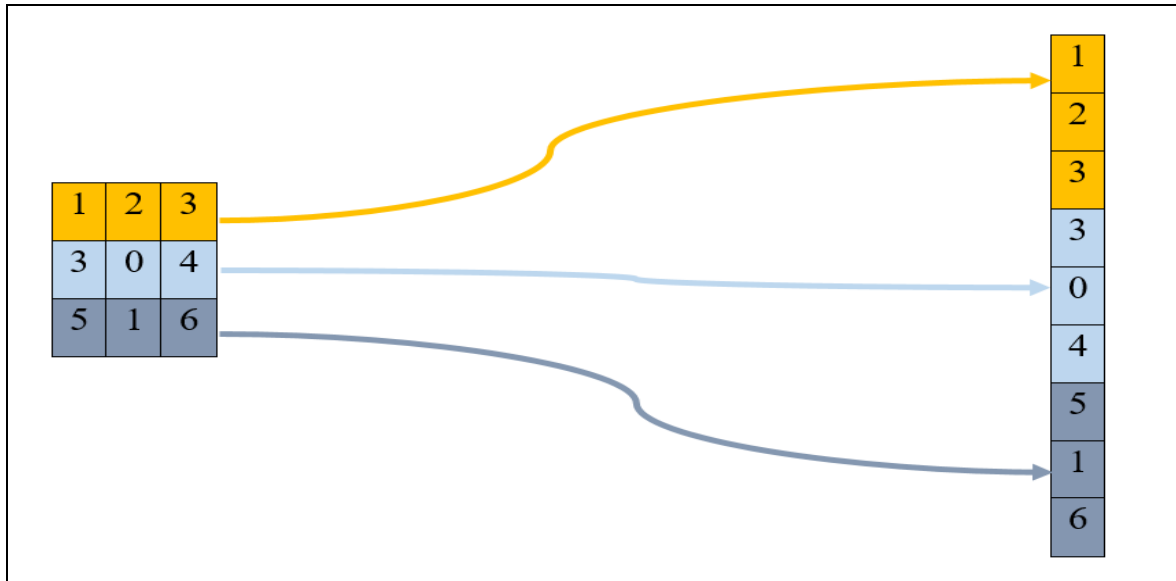
katmanı, ağın görüntüsünün her bir özelliğindeki en önemli bilgilere odaklanarak, girdi boyutunu küçülterek çok daha hızlı eğitim yapılmasını sağlar[72].



Şekil 3.7. Enbüyüğü havuzlama ve ortalama havuzlama işlemleri

3.1.4. Düzleştirme katmanı (yapay sinir ağları için giriş katmanı)

Evrişim ya da havuzlama işleminden sonra düzleştirme katmanı gelir. Düzleştirme katmanında, evrişim ya da havuzlama katmanlarından elde edilen sonuçları tek boyutlu sütun matrisi haline getirir. Tam bağımlı katman için düzleştirme işlemi uygulanır. Şekil 3.8’de tam bağımlı katman için girdi olacak evrişim ya da havuzlama işleminden elde edilen özellik haritasının tek boyutlu sütun matrisi haline getirilmiş hali görülmektedir.



Şekil 3.8. Tam bağımlı katmana giriş için düzleştirilen verilerin elde edilmesi

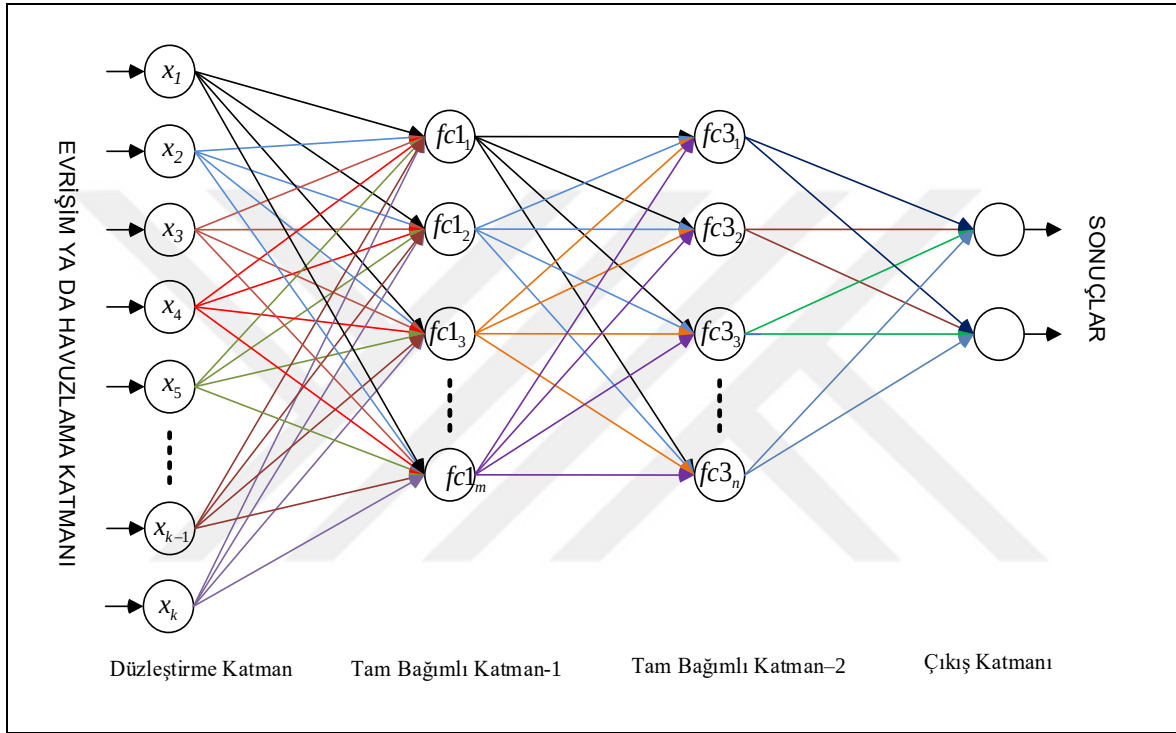
3.1.5. Sinir ağı katmanı (SAK)

YSA'da mutlaka giriş katmanı ve bir çıkış katmanı vardır. Giriş ve çıkış katmanlarının dışında genellikle ara katmanlarda bulunur. Giriş ve çıkış katmanları dış dünya ile bağlantılıdır. Giriş katmanı gerçek dünyadan verilerin alındığı düğüm noktalarıdır. Çıkış katmanı ise YSA'nın sonucunun gerçek dünyaya aktarıldığı düğüm noktalarıdır. Giriş ve çıkış katmanları arasında kalan tüm katmanlar genellikle gizli katmanlar olarak anılırlar. Gizli katmanların gerçek dünya ile bağlantıları olmaz. Gizli katmanlar, kendinden önceki katmandan aldığı verileri ağırlığı ile çarparak sonucunu kendisinden sonra gelen katmana giriş olarak iletir. Tüm ESA'larda düzleştirme katmanından elde edilen tek sütun matrisinin değerlerini SAK'ın giriş katmanında bulunan her bir sinire bağlanır ve tuttuğu değerleri bu sinire aktarır. Yani giriş katmanı dış dünyadan verilerin düzleştirme katmanından sonra ilk girildiği sinir katmanıdır. Giriş katmanı sonra gelen katmanlar için çıkışlar üretilir. Bu çıkışlar gizli katmana aktarılır gizli katmanın çıkışı kendisinden sonra gelen gizli katmana aktarılır ardışık olarak bu işlem böyle devam eder. Sonuncu katman, sonuç katmanı olduğu için kendisinden önce gelen gizli katmandan aldığı verileri işleyerek sonucu dış dünyaya aktarır. SAK'nın sahip olduğu sinirler kendi aralarında farklı şekillerde bağlanırlar. Bu bağlantı türlerine göre adlandırılırlar. Katmanları oluşturan sinirlerin önceki ve sonraki katmanlarda bulunan sinir düğümlerine bağlanma şekillerine göre de adlandırılırlar. Bu bağlantı şekilleri *tam bağlı*, *kısmi bağlı*, *ileri yönlü* ve *çift yönlü* olmak üzere çeşitli bağlantı tipleri vardır.

Tam bağlı katmanlar

Tam bağlı katmanlarda katman sayısı ne kadar olursa olsun her katmandaki bir sinirin çıkışı sonraki katmandaki her bir sinire giriş olarak mutlaka bağlanır. SAK'larda giriş katmanında bulunan düğümlere, düzleştirme katmanında sütun matrisi haline getirilen gerçek dünyadan elde edilen verilerin ilk bağlandığı sinir düğümleridir. Bu düğüm noktaları, düzleştirme katmanının her elemanına bağlıdır. Giriş katmanında buluna düğümlerin çıkışları kendisinden sonra gelen katmandaki her bir düğüme mutlak bağlanır. Bu bağlantı işlemi sonraki katmanlarda da devam eder(bağlanır (tıpkı çok katmanlı yapay sinir ağlarında olduğu gibi). Yani katman sayısı ne kadar olursa olsun her katmanın çıkışı sonraki katmanın her bir sinirine giriş olur. ESA'larda her bir düğümün bir ağırlık değeri vardır. Sinire gelen giriş verisi bu ağırlık değeri ile çarpılır, ardından bir sapma vektörü eklenir. Tam bağlı

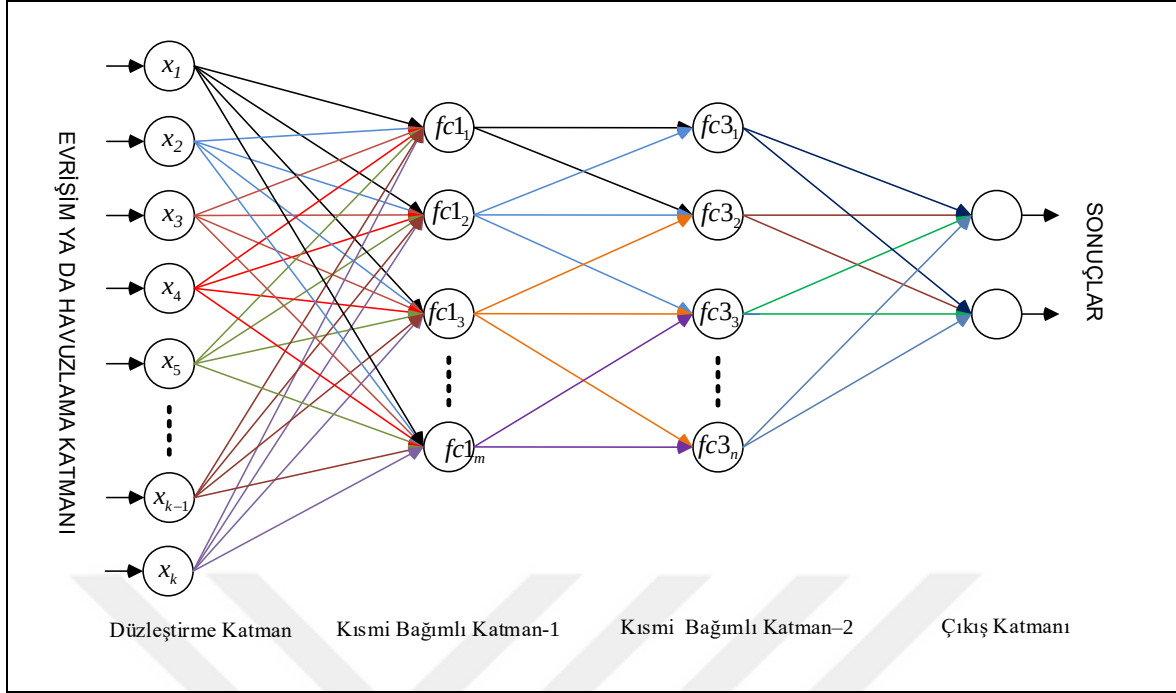
katman önceki katmanlardan elde edilen özellikleri değerlendirerek bu özelliklere göre sınıflandırma yapar. Yani tam bağlı katmanlarda YSA'lar ile öğrenme işlem yapılır. Tipik olarak, son tam bağlantılı katmanın bağlı olduğu sonuç katmanı, modelin tahmin etmeye çalıştığı sınıflandırma etiketlerinin her biri için değer üretir. Ürettiği değeri gerçek dünyaya bu katmandan aktarır. Şekil 3.9'da ESA'larda SAK'ta tam bağlı katmanın sinirlere bağlantı şeklinin nasıl yapıldığı gösterilmiştir.



Şekil 3.9. ESA'larda SAK'ta tam bağlı katmanın sinirlere bağlantısı

Kısmi bağlı katmanlar

Düzeleştirme katmanında sütun matrisi haline getirilen veriler, kısmi bağlı katmanın giriş katmanında ki her biri bir nöronlara mutlak bağlanır. Ancak kısmi bağlı birinci katmanın çıkışı, kendisinden(giriş katmanı) sonra gelen katmanlardaki nöronların ger birine giriş olarak bağlanmazlar. Yani giriş katmanındaki bir sinirin çıkışı sonraki katmanın her bir nöronuna bağlanmaz. Bir katmandaki her bir sinirin çıkışının sonraki katmandaki her bir sinirin girişine bağlı olmayıp en yakın komşu sinir(ler)e bağlı olduğu bağlantı şeklidir. Bu bağlantı gizli katmanlarda da aynıdır. Bu bağlantı tipine kısmi bağlı katman denir.



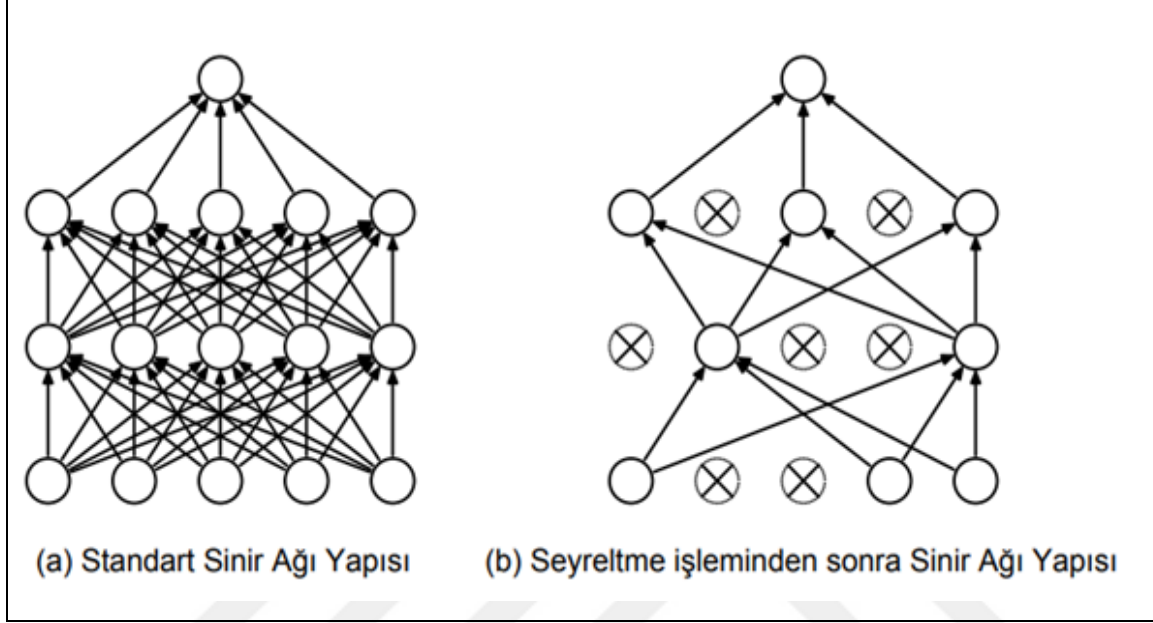
Şekil 3.10. ESA’larda SAK’ta kısmi bağı katmanın sinirlere bağlantısı

Şekil 3.10’da ise ESA’ların yapısında bulunan katmanlar ve bu katmanların birbirlerine bağlantıları gösterilmiştir. ESA’lar bu katmanları da yapılan işlemlerle sayısal görüntülerin tüm ayrıntılarını ve özelliklerini çıkartarak nenelerin tanımlamasını yapar.

3.1.6. Seyreltme katmanı (Dropout layer)

YSA’larda veriler eğitilirken benzer verilerin çok fazla kullanılması aşırı öğrenmeye neden olur ve sinir ağları ezberler, ezber yapan sinir ağı sonuç üretirken uydurur[41]. Aşırı uydurmayı önlemek için sinir seyreltme(Dropout) işlemi uygulanır[41, 89]. “Çok sayıda parametresi olan derin sinir ağları, çok güçlü makine öğrenimi sistemleridir. Bu tür ağlarda aşırı uyum ciddi bir sorundur. Büyük ağların kullanımı da yavaştır, bu da test zamanında birçok farklı büyük sinir ağının tahminlerini birleştirerek aşırı uydurma ile başa çıkmayı zorlaştırır. Bu sorunu çözmek için seyreltme(dropout) yöntemi kullanılır”[89]. Seyreltme işlemi, görüntü tanıma, konuşma tanıma, belge sınıflandırması ve hesaplamalı biyolojide denetimli öğrenme görevlerinde sinir ağlarının performansını artırdığını ve birçok kıyaslama veri setinde başarılı sonuçlar vermektedir[89]. Seyreltme işlemi ile bir sinir düğümü yok sayılarak yok sayılan düğümün tüm bağlantıları kesilir yani o düğümün önceki ve sonraki katmanlarla olan bilgi akışı engellenir[41]. Bu teknik nesne sınıflandırma, basamak tanıma, konuşma tanıma, belge sınıflandırması ve hesaplamalı biyoloji verilerinin analizi dahil

olmak üzere çok çeşitli uygulama alanlarında YSA'ların performansını iyileştirdiği tespit edilmiştir[89]. Aşağıdaki Şekil 3.11'de Seyreltilme işleminin yapılışı görülmektedir. Şekil 3.11. a) iki gizli katmanlı standart sinir ağı yapısı Şekil 3.11. b) seyreltilmiş sinir ağı yapısı bazı sinirlerin bağlantıları ihmal edilmiş[89] hali görülmektedir.

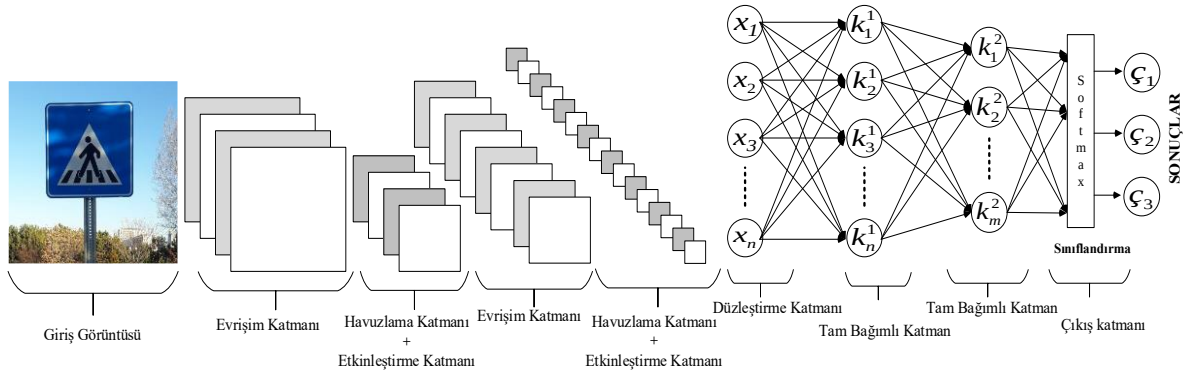


Şekil 3.11. Seyreltilmiş sinir ağı modeli a) iki gizli katmanlı standart sinir ağı yapısı b) seyreltilmiş sinir ağı yapısı bazı sinirlerin bağlantıları ihmal edilmiş[89]

3.1.7. Sınıflandırma katmanı

Bu katmanda ESA'lerden eğitilmiş veri setine göre sınıflandırılması istenen kaç tane sınıf var ise bu sınıfların her biri için değer üretir. Bu değer (0 – 1) aralığındadır. (0) benzerlik olmadığını (1) ise yüzde yüz benzer olduğunu ifade eder. Bu tez çalışmasında 20 tane nesne için eğitilmiş veri seti hazırlanmıştır. Tanımlaması ve sınıflandırması yapılacak 20 nesne gurubu için değer üretir. Üretilen değerlerin en yüksek olanı tahmin değeri olur.

Şekil 3.12'de ESA'ların yapısında bulunan tüm katmanlar gösterilmiştir. ESA'lar bu katmanları da yapılan işlemlerle sayısal görüntülerin tüm ayrıntılarını ve özelliklerini çıkartarak nenelerin tanımlamasını yapar. ESA'ların çeşitlerine katmanların sayılarında ve yerlerinde farklılıklar olabilir.



Şekil 3.12. ESA'ların yapısında bulunan katmanların gösterimi

3.1.8. Benzerlik katsayısı

Benzerlik katsayısı iki kümenin benzerliklerini ölçmek için kullanılır. Bu iki küme arasındaki benzerlikler ölçülürken kümeleri oluşturan elemanların özelliklerine bakılır ve bu özelliklere göre derecelendirilir. Uygulamalarda elde edilen sonuçların elde edilmesi gereken sonuçlarla aralarındaki benzerliklerin ve farklılıkların ölçülmesi gerekir. Veri setleri arasındaki benzerlikleri anlamak ve birbirlerine ne kadar benzeyip benzemediğinin ölçülmesi ve ölçüm sonuçlarının değerlendirilmesi gerekir. Veri setleri arasındaki benzerlikleri ölçmek istatistiksel bir işlemdir. Veri setleri arasındaki benzerlikler, benzerlik katsayıları kullanılarak ölçülür.

Örneğin Resim 3.2'de bir trafik levhası görülmektedir. Resim 3.16'da görülen trafik levhası yeşil çerçeve için alınmıştır. Yeşil çerçeve içindeki alan trafik levhasının gerçekte kapladığı alandır. Kırmızı çerçeve içinde ki alan ise nesne tespit algoritmasından elde edilen sonuçtur. Yani kırmızı çerçeve içindeki alan, nesne tespit algoritmasının sonuç olarak ürettiği trafik levhası alanıdır. Resim 3.2'de de açıkça görüldüğü gibi alanlar arasında farklar bulunmaktadır. Nene tespit algoritmasının bulduğu alanın ne kadarı gerçekte ki trafik levhası alanı ile örtüştüğünün hesaplanması yani yapılan işlemin sağlamasını yapmak gerekir.



Resim 3.2. Trafik işaretinin yeşil çerçeve ile gerçek sınırları, kırmızı çerçeve ile nesne tespit algoritmasından elde edilen sınırları gösterilmiştir

Yapılan işlemin gerçekte olması gereken alan ile olan ilişkisinin ölçülmesi yani bulunan alanın doğruluğunun sağlamasını yapmak için iki veri seti arasındaki benzerlikleri ve farklılıkları yani kesişen alanları ölçmek, karşılaştırmak ve değerlendirmek gerekir. Bu işlemler için çeşitli yöntemler geliştirilmiştir. Geliştirilen bu yöntemlerin en yaygın olarak kullanılanları *Dice* ve *Jacard* benzerlik katsayılarıdır. Görüntü işleme uygulamalarında görüntülerin bölütlenmesi ve ayrıştırılmasında yapılan işlemin doğruluğunun ölçülmesinde bu benzerlik katsayılarından yararlanılır.

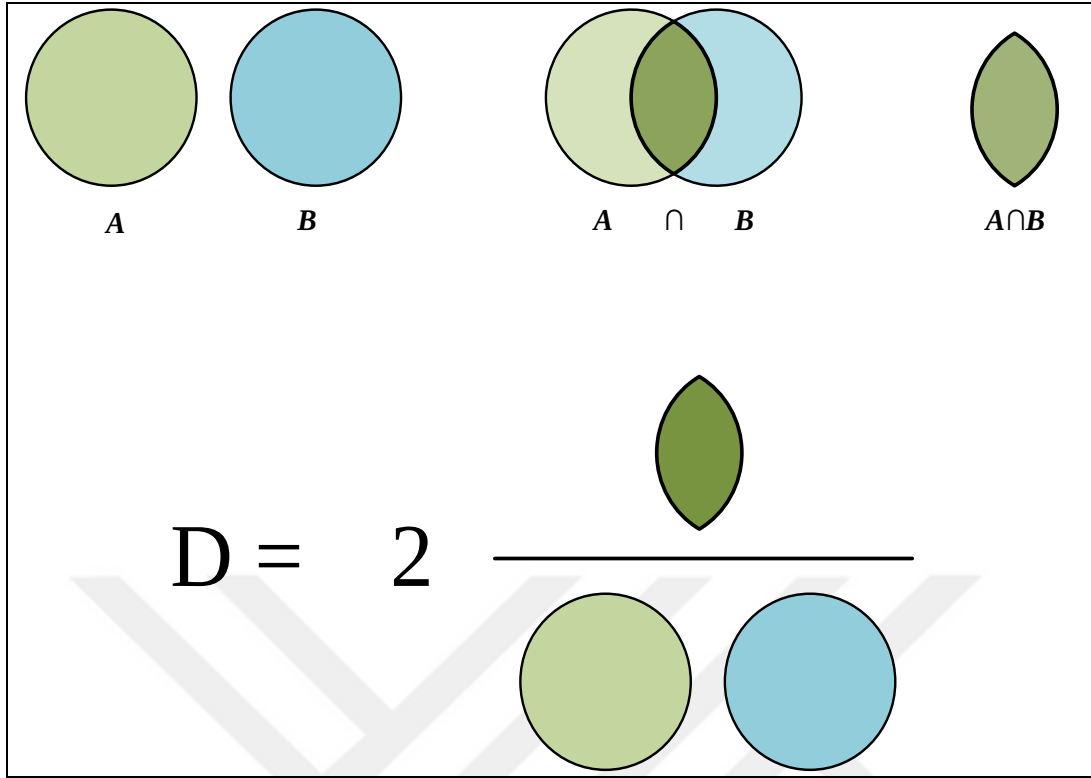
Dice benzerlik katsayısı

Veri setlerinin sonuçlarının doğrulanması için kullanılan istatistiksel bir yöntemdir. İki verinin birbiri ile ne kadar benzeştiğinin ya da örtüştüğünün ölçmek için kullanılan bir yöntemdir. Lee R. Dice tarafından 1945 yılında önerilen bu yöntem, iki örneğin benzerliğini ölçmek için istatistik olarak Rastlantı İndeksi(Coincidence Index) olarak adlandırılan katsayıyı kullanarak türler arasındaki ilişkiyi ifade etmektedir[90]. Thorvald Sørensen türlerin

varlıklarından ziyade türlerin yokluğuna odaklanan bir formül kullandı. Bu nedenle Sørensen-Dice indeksi, Dice katsayısı, Dice örtüşmesi, Sørensen indeksi gibi isimlerle verilmiştir[90]. Dice katsayısının formülü denklem (3.16)'da görülmektedir. Denklem (3.16)'daki D zar katsayısını ifade eder. Formülden elde edilen D katsayısı $(0 - 1)$ aralığındadır. Elde edilen değer 1'e ne kadar yakınsa iki küme arasındaki benzerlikte o kadar çok örtüşen oranların fazla olduğu anlaşılır. Değerin 1 olması yüzde yüz kesişmeye 0 olması kesişmenin hiç olmadığı anlamına gelir. A , A kümesindeki öğeleri temsil eder, B , B kümesindeki öğeleri temsil eder. $A \cap B$, A ve B kümelerindeki kesişen yani ortak, benzer, kesişen olan öğeleri temsil eder.

$$D = 2 \frac{|A \cap B|}{|A| + |B|} \quad (3.16)$$

Aşağıdaki Şekil 3.13'de Dice benzerlik katsayısının şekiller ile ifadesi görülmektedir. Şekil 3.13'de gösterilen şekil, Denklem (3.16)'da Dice katsayısının matematiksel ifadesinin şekillendirilmiş halidir. A ve B kümelerinin kesişimlerinin mutlak değerinin 2 katı, A ve B kümelerinin mutlak değerlerinin toplamına bölünmesini ifade eder.



Şekil 3.13. Dice benzerlik katsayısının şekillerle matematiksel olarak ifadesi

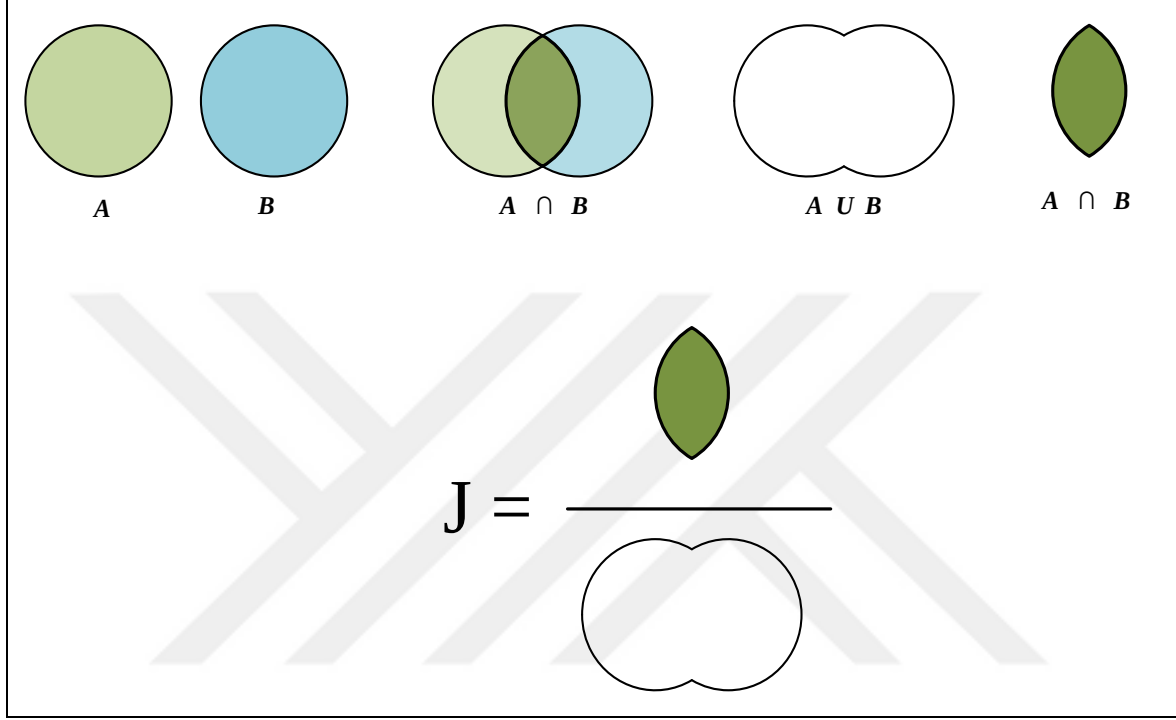
Jaccard benzerlik katsayısı

Veri setlerinin benzerliğini ve çeşitliliğini ölçmek için kullanılan istatistiksel bir yöntemdir. Jaccard benzerlik katsayısı 15 Aralık 1907 yılında Paul Jaccard tarafından ortaya atılmıştır[91]. Jaccard indeksi, bilgi çıkarımı, veri madenciliği, makine öğrenimi ve daha pek çok alanda birçok pratik uygulamaya sahip kümelerdeki klasik bir benzerlik ölçüsüdür[92]. İki sonlu veri setleri olan A ve B veri setlerinin benzerliklerinin örtüşmesinin göreceli boyutunu[92] Jaccard katsayısı denklem (3.17)'de gösterilmiştir. Denklem (3.17)'de J Jaccard katsayısını, A , A kümesindeki öğeleri temsil eder, B , B kümesindeki öğeleri temsil eder. $A \cap B$, A ve B kümelerindeki kesişen yani ortak olan öğeleri temsil eder iken $A \cup B$, A ve B kümelerinin bileşimini yani ortak olan öğelerini temsil eder.

$$J = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (3.17)$$

Şekil 3.18'de Jaccard benzerlik katsayısının şekiller ile ifadesi görülmektedir. Şekil 3.14'de A kümesi ile B kümesinin toplamından A ve B kümelerinin kesişimi çıkartılarak A ve B

kümelerinin birleşimi elde edilir. A ve B kümelerinin kesişimi A ve B kümelerinin birleşimine bölünür. Yani iki kümenin benzer elemanlarının sayısı ile iki kümenin farklı elemanlarının toplamına bölünerek birlik üzerinde kesişim oranı yani Jaccard katsayısı elde edilir. Jaccard katsayısı $(0 - 1)$ aralığında bir değerdir.



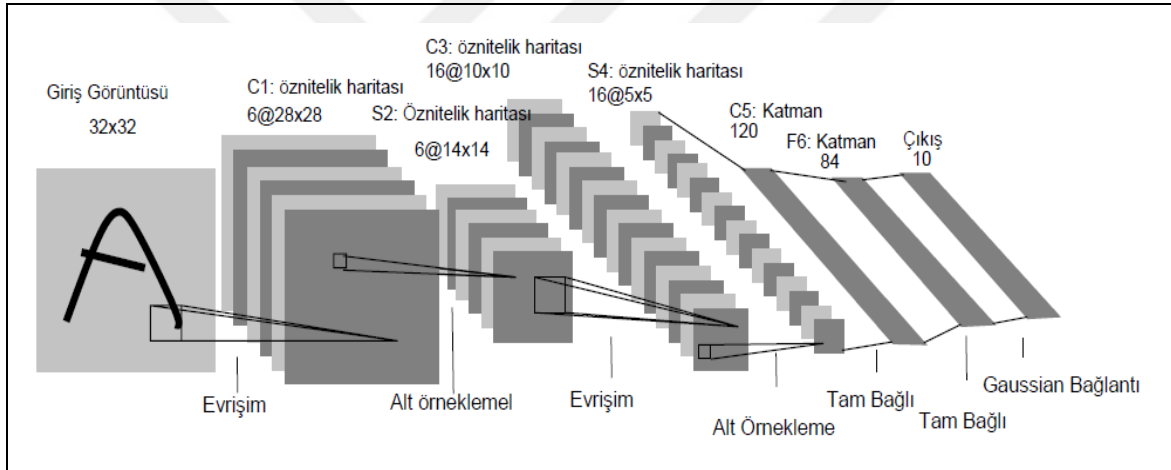
Şekil 3.14. Jaccard benzerlik katsayısının şekillerle matematiksel olarak ifadesi

3.2. ESA Mimarileri

DÖ'nün görüntü işleme uygulamalarında nesneleri tanımlamak için geliştirilmiş ESA yapılarıdır. Large Scale Visual Recognition Challenge-Büyük Ölçekli Görsel Tanıma Yarışması (ILSVRC) ImageNet yarışmasıyla ESA'lar öncülüğünde görüntü tanımlama ile ilgili ihtiyaçları karşılamak için çeşitli ESA yapıları geliştirilmiştir. ImageNet yarışması öncülüğünde ortaya çıkan ESA mimarilerdir. "ImageNet yarışmasının amacı görüntülerde bulunan ana nesneleri belirlemek, tanımlamasını ve sınıflandırmasını yapmaktır. ImageNet'e dünyanın dört bir yanından araştırmacıların kolayca erişilebildiği ve katkı sağlayabildiği bir işlenmiş görüntü veri tabanı sağlamak için devam eden bir araştırma çabasıdır"[93]. Günümüzde araştırmacılar tarafında yaygın olarak kullanılmaktadır.

3.2.1. LeNet mimarisi

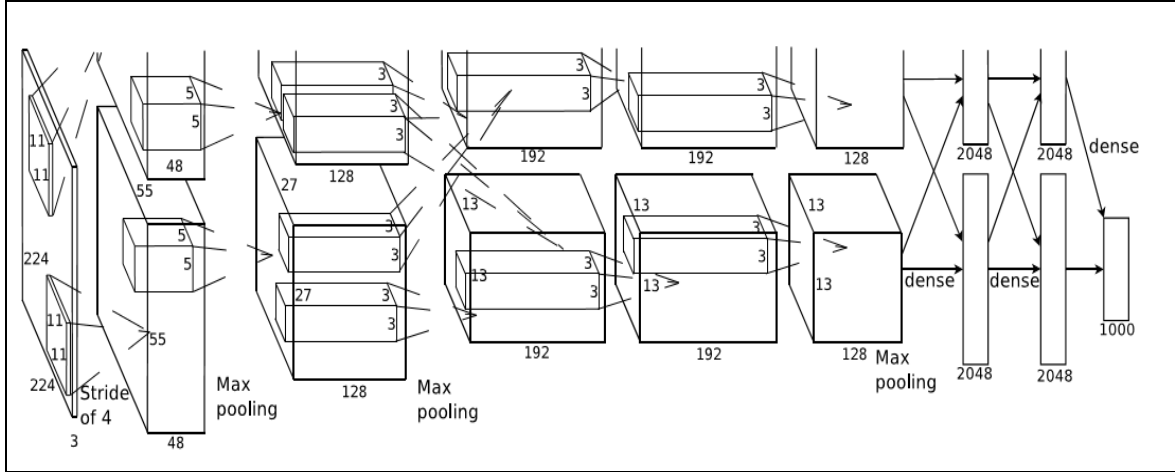
İlk ESA mimarisi olan LeNet, LeeCun ve ark. tarafından 1998 yılında geliştirilmiştir[69]. LeeCun ve ark. tarafından geliştirilen LeNet mimarisi, ESA'ların öncüsü olmuştur. Gradyan tabanlı öğrenme algoritmalarını kullanarak geliştirilen bu ağ el yazısı karakterlerini tanımlar. Ticari işlemlerde yaygın bir şekilde kullanılan banka çekleri üzerindeki yazıları başarılı bir şekilde okumuştur[69]. Giriş görüntüsü alındıktan sonra evrişim işlemi yapılır. Evrişim işleminden sonra etkinleştirme işlemi yapılır. Etkinleştirme fonksiyonu ile veriler doğrusallaştırılır. Sonra havuzlama işlemi yapılır. Havuzlama işleminde ortalamayı havuzlama yöntemi kullanmıştır. Şekil 3.15'de LeNet-5 mimarisini oluşturan katmanlar ve bağlantıları görülmektedir.



Şekil 3.15. LeNet mimarisini oluşturan katmanlar[69]

3.2.2. AlexNet

AlexNet mimarisi, Krizhevsky ve ark. tarafından 2012 yılındaki ImageNet görsel tanıma yarışmasında sınıflandırma doğruluk oranı tüm zamanların rekoru olan %83,6'ya çıkararak birinci olmuştur. Bu başarıdan sonra ImageNet yarışmasıyla popüler olmuştur. AlexNet mimarisi 5 tane ardışık evrişim, havuzlama katmanı ile seyreltme işlemi uygulanan 3 tam bağlı katmandan oluşur[79]. AlexNet'te etkinleştirme fonksiyonu olarak ReLu kullanılmıştır[79]. Şekil 3.16'da AlexNet mimarisini oluşturan katmanlar görülmektedir.

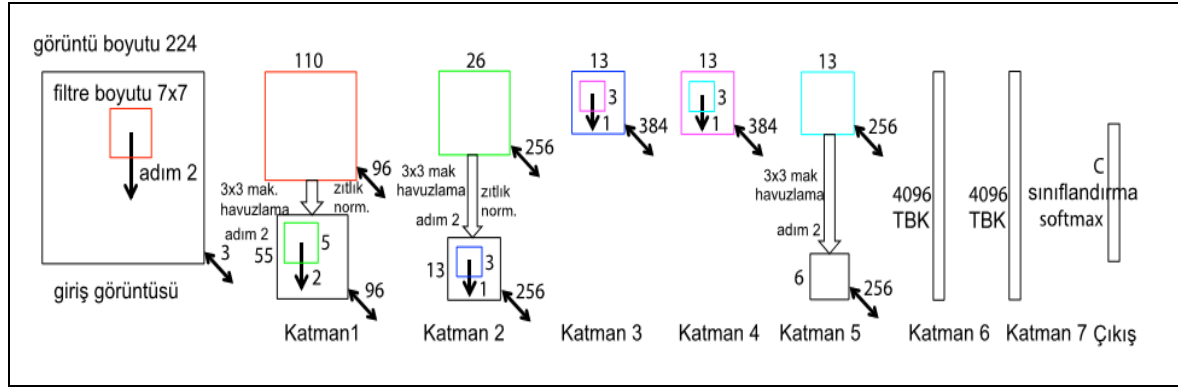


Şekil 3.16. AlexNet mimarisini oluşturan katmanlar[79]

3.2.3. ZFNet mimarisi

ImageNet yarışması, 2012 yılındaki AlexNet'tin doğruluk oranı tüm zamanların rekoru olan %83,6'ya çıkarmasından sonra DÖ tüm ilgiyi üzerine çekerek büyük bir çıkış yapmıştır. 2013 yılındaki ImageNet yarışmasının kazananı %88,8'lık başarı oranı ile ZFNet olmuştur[94]. ZFNet adını tasarımcıları Matthew Zeiler ve Rob Fergus'un soyadlarının baş harfinden almıştır.

Şekil 3.17'de ZFNet mimarisinin yapısı görülmektedir. Tüm filtreler ve özellik haritaları kare şeklinde olan 8 katmanlı mimariye sahiptir. Bir görüntünün 3 boyutlu renkli görüntüleri (224x224x3) kullanır. Filtreler x ve y düzleminde 2 adım kaydırılır. İlk katmanda her biri 7x7 boyutunda 96 farklı evrişim filtresi kullanılır. Sonra katmanda öznelik haritaları düzeltilmiş bir doğrusal fonksiyonu ReLu'dan geçirilir (gösterilmemiştir). Sonraki katman 2-3-4-5'te benzer işlemler tekrarlanır. Son iki katman, vektör formundadır. Girdi olarak üst evrişimli katmandan özellikleri alarak tamam bağlıdır. Son olan çıkış katmanında softmax işlemi uygulanır[94].



Şekil 3.17. ZFNet mimarisinin yapısı

3.2.4. VGGNet mimarisi

2014 yılında ImageNet yarışmasına katılan Oxford Üniversitesi, Görsel Geometri Grubu (VGG- Visual Geometry Group) tarafından oluşturulan ve % 7,3'lik hata oranı ile nesne sınıflama konusunda başarı elde eden ESA mimarisidir. VGGNet mimarisine 3x3 boyutunda evrişim filtreleri kullanarak mimarinin derinliği artırmıştır. Bu ağ derinliğin evrişim ve tam bağlı katmanların sayısını arttırarak önemli bir iyileştirmenin elde edilebileceğini göstermiştir. VGG16-VGG19 katmanlı yapıları vardır. Aralarındaki fark katman sayılarının fazla olmasıdır. Maksimum havuzlama yöntemi kullanılmıştır.

VGGNet-ConvNet yapısı (sütunlarda) gösterilir. Konfigürasyonların derinliği, daha fazla katman eklendikçe soldan (A) sağa (E) doğru artar (eklenen katmanlar koyu gösterilmiştir). Evrişimli katman parametreleri, “evrişimli alan boyutu-kanal sayısı” olarak belirtilir. ReLU aktivasyon işlevi kısalık için gösterilmemiştir.

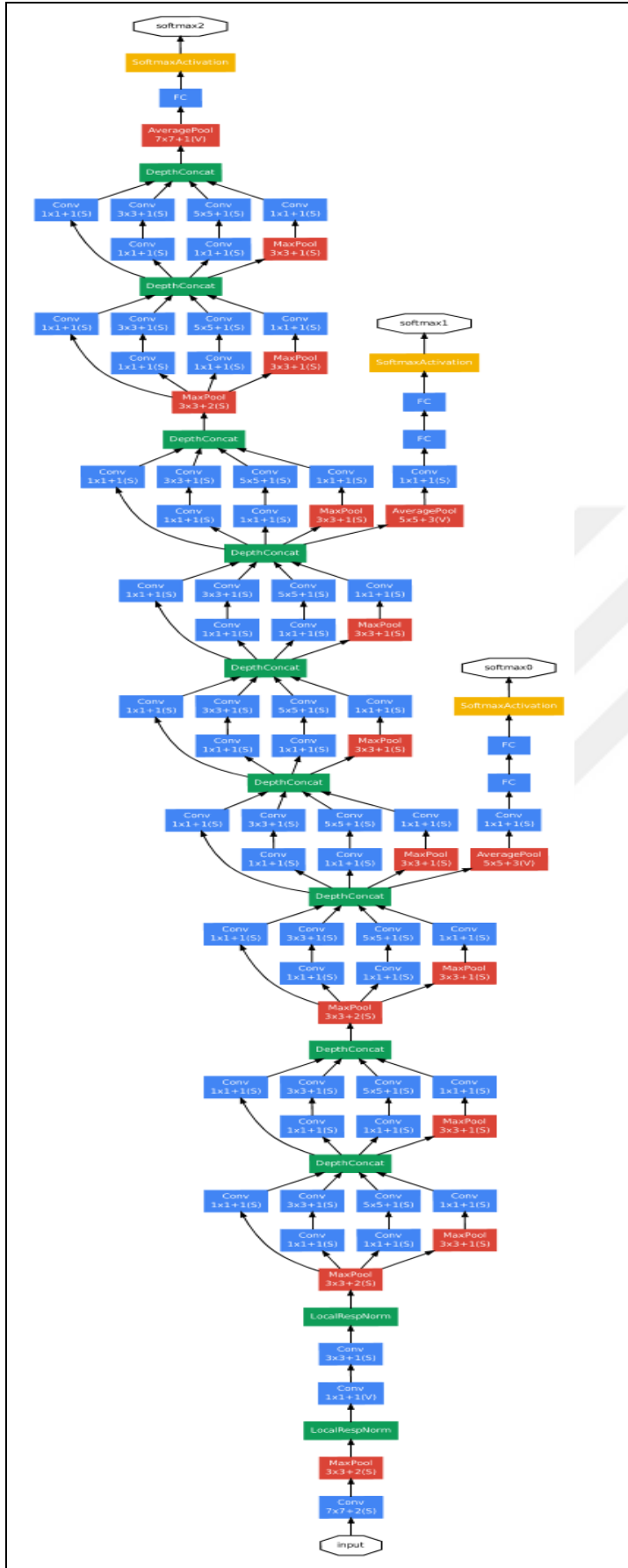
Aşağıdaki Çizelge 3.2’de VGGNet-ConvNet mimarisini oluşturan tüm katmanlar ve yapılan işlemler gösterilmiştir.

Çizelge 3.3. VGGNet-ConvNet mimarisini oluşturan katmanlar

VGGNET-ConvNet Yapısı					
A	A-LRN	B	C	D	E
11 ağırlık katmanı	11 ağırlık katmanı	13 ağırlık katmanı	16 ağırlık katmanı	16 ağırlık katmanı	19 ağırlık katmanı
Giriş görüntüsü (224x224x3) RGB					
Evrişim3-64	Evrişim3-64 LNR	Evrişim3-64 Evrişim3-64	Evrişim3-64 Evrişim3-64	Evrişim3-64 Evrişim3-64	Evrişim3-64 Evrişim3-64
Maksimum Havuzlama					
Evrişim3-128	Evrişim3-128	Evrişim3-128 Evrişim3-128	Evrişim3-128 Evrişim3-128	Evrişim3-128 Evrişim3-128	Evrişim3-128 Evrişim3-128
Maksimum Havuzlama					
Evrişim3-256 Evrişim3-256	Evrişim3-256 Evrişim3-256	Evrişim3-256 Evrişim3-256	Evrişim3-256 Evrişim3-256 Evrişim1-256	Evrişim3-256 Evrişim3-256	Evrişim3-256 Evrişim3-256 Evrişim3-256 Evrişim3-256
Maksimum Havuzlama					
Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512 Evrişim3-512 Evrişim3-512
Maksimum Havuzlama					
Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512	Evrişim3-512 Evrişim3-512 Evrişim3-512 Evrişim3-512
Maksimum Havuzlama					
Tam Bağlı Katman-4096					
Tam Bağlı Katman-4096					
Tam Bağlı Katman-1000					
Softmax					

3.2.5. GoogleLeNet mimarisi

2014 yılındaki ImageNet yarışmasının kazananı %5,7 hata oranı ile Szegedy ve arkadaşlarının sunduğu GoogLeNet olmuştur[95]. LeNet mimarisine göre GoogLeNet mimarisi 22 katmandan oluşan daha derin ve daha geniş bir yapıya sahiptir havuzlama katmanları ile birlikte katman sayısı 29 olmuştur[95]. “GoogLeNET ağına çıkışına doğru Tam Bağımlı Katmanlar yerine ‘Ortalamayı Havuzlama’ kullanarak önemli olmayan çok sayıda parametrenin ortadan kaldırılmasını sağlamıştır”[41]. Aşağıdaki Şekil 3.18’de tüm katmanları ile birlikte gösterilen GoogLeNet mimarisini görülmektedir.

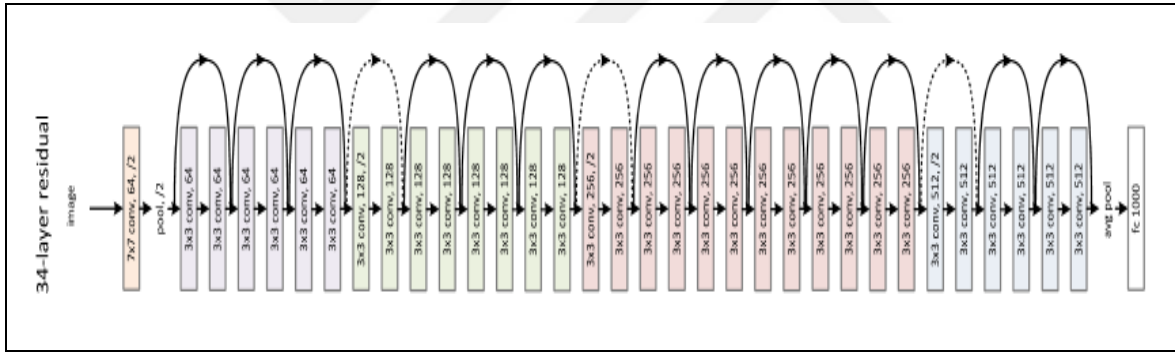


Şekil 3.18. Tüm katmanları ile birlikte GoogLeNet mimarisi[95]

3.2.6. ResNet mimarisi

Microsoft ResNet, Residual Neural Network ve ResNet olarak adlandırılır. Residual Neural Network, Kalıntı Sinir Ağıdır. Kalıntı Sinir Ağıdır. Yaygın olarak kısaca ResNet olarak kullanılır. ResNet, Microsoft Araştırma Birimi YZ Grubu tarafından He ve arkadaşlarının geliştirdiği ResNet mimarisi, ImageNet yarışmasının 2015 yılındaki %96,43'lük başarı oranı ile şampiyonu olmuştur. ResNet mimarisi atlama bağlantıları veya kalıntı blokları kullanarak, belirli katmanları atladı böylelikle hesaplama karmaşıklığına sebep olan parametre sayısını azalttı hem ağın derinliğini arttırırken hem de modelin başarısını geliştirdi[96].

Aşağıdaki Şekil 3.19'da 34 katmanlı ResNet mimarisini oluşturan katmanlar ve bağlantılar görülmektedir.



Şekil 3.19. ResNet mimarisini oluşturan 34 katmanlı yapı[96]

Aşağıdaki Çizelge 3.4'de ResNet mimarisini oluşturan evrişim katmanları ve evrişim katmanlarında kullanılan filtreler, enbüyüğü havuzlama, kaydırma adımı vb. gibi özellikler gösterilmiştir. Çizelge 3.3'de gösterilen ResNet mimarisi 34 katmanlı bir yapıdan oluşmaktadır[96].

Çizelge 3.4. 34 katmanlı ResNet mimarisi[96]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

3.3. Derin Ağlarda Kullanılan Programlama Dilleri

YZ uygulamaları için kullanılacak olan programlama dilinin seçim, yapılacak olan uygulamanın yapısına bağlıdır. Her dilin kendine has üstün özellikleri vardır ve çalışma performans farkları olabilmektedir. Bazı programlama dilleri, bazı alanlarda daha özelleşmiş fonksiyonlara sahiptir. YZ uygulamalarında, kullanılan DÖ Kütüphaneleri de çok önemlidir. Bu nedenle kullanılacak olan DÖ Kütüphaneleri ile etkin çalışan programlama dilinin seçimi önemlidir. DÖ uygulamalarında Python, Java, C ++ ve R dili yaygın olarak kullanılan programlama dillerindendir.

3.3.1. Python programlama dili

Python prgramlama dili, Guido van Rossum tarafından 1991 yılında geliştirilmiş ve kullanıma sunulan Python programlama dili[97], genel amaçlı programlamalar için etkin bir şekilde kullanılan yüksek seviyeli bir dildir. “Python, yorumlanmış, etkileşimli, nesneye yönelik bir programlama dilidir. Modülleri, istisnaları, dinamik yazımı, çok yüksek düzeyde dinamik veri türlerini ve sınıfları içerir. Prosedürel ve fonksiyonel programlama gibi nesneye yönelik programlamanın ötesinde çoklu programlama paradigmalarını destekler. Python olağanüstü gücü çok net bir sözdizimi ile birleştirir”[98]. Python dili en esnek dillerden biridir ve çok çeşitli amaçlarla kullanılabilir. Python, basitliği sayesinde Yapay Zeka alanında en yaygın kullanılan programlama dillerinden biridir. Veri yapıları ve sıkça

kullanılan diğer Derin Öğrenme algoritmaları ile sorunsuz bir şekilde kullanılabilir. Python, Scipy ve Numpy gibi makine öğrenimi için özel kütüphaneler içermektedir.

3.3.2. C++ programlama dili

C++ programlama dili, nesne yönelimli ve yüksek seviyeli, genel amaçlar için geliştirilmiş bir programlama dilidir[99, 100]. C++ programlama dili, Bell Laboratuvarlarında 1979 yılında Bjarne Stroustrup tarafından geliştirilmeye başlanan ilk sürümünün adı “C with classes” dır ve C programlama dilinin bir eklentisi olarak geliştirilmiştir[99, 100]. 1983 yılında C++ adını almıştır. C++ dilini Bjarne Stroustrup doktora öğrenciyken geliştirmiştir[100]. Büyük küçük harf duyarlıdır. Nesne Yönelimli (Object-Oriented) bir programlama dilidir. Hem işlevli hem de performansı bakımından önemli yazılım dillerindendir[99]. Nesne Yönelimli (Object-Oriented) bir programlama dili olması sebebi ile birçok bilgisayar oyununda, işletim sistemlerinde ve uygulamaların geliştirilmesinde temel dil olarak kullanılır, bu gibi nedenlerden dolayı günümüzde yaygın bir şekilde kullanımı devam etmektedir. Amazon ve Google Chrome'un temellerini oluşturan C++, aynı zamanda robotik kodlamada kullanılmaktadır. C++ dilinde çok sayıda kaynak ve kütüphane bulmak mümkündür.

3.3.3. Java programlama dili

“Java, 1991 yılında James Gosling, Patrick Naughton, Chris Warth, Ed Frank ve Mike Sheridan tarafından tasarlandı. İlk çalışan versiyonunun geliştirilmesi 18 ay sürmüştür. Java dili başlangıçta “Oak” olarak adlandırıldı. 1992 sonbaharında Oak'ın ilk uygulaması ile 1995 ilkbaharında Java'nın kamuya duyurulması arasında, Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin ve Tim Lindholm dilin tasarım ve evrimine önemli katkıda bulundu. Java'nın karakterinin çoğu C ve C++ dillerinden miras alınmıştır.”[101]. Java dili 1995 yılına kadar Oak isimi ile anılmıştır. 1995 yılında “Java” olarak yeniden isimlendirilmiştir. Java, açık kaynak kodlu, nesneye yönelimli, ortamdan bağımsız, çok işlevsel, yüksek seviyeli, adım adım işletilen yaygın olarak kullanılan bir programlama dilidir. Java'nın nesne odaklı özelliklerinin çoğunu alırken C ++ dilinden etkilenmiştir[101]. Java uygulamaları, herhangi bir Java Virtual Machine'de bilgisayar mimarilerinden bağımsız olarak çalıştırılabilen bir sınıf dosyasıdır. “Java hemen hemen her çeşitteki ağ uygulamalarının temelinde bulunan gömülü programlamalarda, mobil uygulamalarda, oyunların yazılımında, web temelli içerik

oluşturmada, kurumsal şirketlerin yazılım geliştirmesinde kullanılan genel bir standarttır. 9 milyondan fazla geliştiricisiyle uygulamalarını ve hizmetlerini verimli bir şekilde geliştirmeye ve dağıtmaya olanak tanır”[102].

3.3.4. Scala programlama dili

Scala programlama dilinin ilk sürümü Martin Odersky ve araştırma ekibi tarafından 2003 yılında geliştirildi. Scala, Java'ya benzer bir programlama dilidir. Scala, hem nesne yönelimli hem de fonksiyonel programlama yapmaya imkan tanır. Scala, bu iki temel programlama modelinin başarılı bir şekilde kullanılabildiği hibrit bir dildir Scala programlama dili, derleyici tabanlı ve yoğun, hızlı ve verimli çok paradigmalı bir programlama dilidir.

3.3.5. R programlama dili

R dili açık kaynak kodlu R vakfi tarafından desteklenen, istatistiksel hesaplamalar ve grafikler için geliştirilmiş GNU tasarısının bir parçasıdır. Bu dil, Bell laboratuvarlarında geliştirilen modern S dilinin bir sürümü olarak geliştirilmiştir[103]. R programlama dili S ve S-Plus programlama dillerinin başka bir türevi olarak ilk defa Auckland Üniversitesi İstatistik Bölümünde, Ross Ihaka ve Robert Gentleman tarafından 1991 yılında geliştirilmiş ve 1993 yılında kullanıma sunulmuştur[103, 104]. R dili veri analizi için geliştirilmiş bir programlama dilidir, istatistiksel ve grafiksel yöntemlerin hesaplaması kapsamlı bir şekilde kolaylıkla yapabilen yapıya sahiptir[104]. R dili, istatistiksel modellerin üretilmesinde esneklik sağlamaktadır. R Dili oldukça yeni bir dil olmasına rağmen akademik araştırmalarda ve sektörde gittikçe artan bir oranda kullanımı yaygınlaşmaktadır.

3.3.6. Lua programlama dili

Lua programlama dili, ağırlıklı olarak gömülü sistemler için geliştirilmiş hafif paralel gömülebilir betik bir dildir[105, 106]. Lua dili, ANSI C'de yazıldığından dolayı çapraz platform desteğine sahip olan sade bir C API'sine sahiptir[105]. 1993 yılında Pontifical Katolik Üniversitesi'nde Bilgisayar Grafik Teknoloji Grubu üyesi Roberto Ierusalimsky, Luiz Henrique de Figueiredo ve Waldemar Celes tarafından Brezilya'da geliştirilmiştir[107]. Yapısal programlama, nesne yönelimli programlama, işlevsel programlama, veri tabanı programlama ve veri açıklama yapılarını destekler. C dilinde

gerçekleştirilen çalışmaları kolaylaştırmak amacıyla geliştirilen bu dil, gelişerek bir programlama dili haline gelmiştir. Lua programlama dili, hızlı ve az yer kapladığı için oyun programlama tercih edilir.

C ve Fortran tabanlı sayısal işlemler için tasarlanmış yüksek başarımli, dinamik bir programlama dilidir. Nitelikli bir derleyici, dağıtık koşturucu, sayısal hesaplamalarda yüksek doğruluk oranı ve geniş bir matematiksel işlev kütüphanesine sahiptir[108]. Jeff Bezanson, Stefan Karpinski, Viral B. Shah ve Alan Edelman tarafından 2009 yılında MIT laboratuvarlarında geliştirilmeye başlandı[109].

3.4. Derin Ağlarda Kullanılan Derin Öğrenme Kütüphaneleri

DÖ uygulamaları için C++, python, java, R, Julia vb. gibi farklı programlama dilleri kullanılarak geliştirilmiş DÖ kütüphaneleri vardır. Bu kütüphaneler, geliştirilme amacına uygun farklı özelliklere sahip olabilir. Burada kullanım amacına göre geliştirilen ve yaygın olarak kullanılan nesne tanımlama ve sınıflandırma kullanılabilecek üstün özellikleri olan DÖ kütüphanelerinden bazıları tanıtılacaktır.

3.4.1. Caffe kütüphanesi

Caffe, Yangqing Jia tarafından doktora çalışması döneminde Berkeley Görme ve Öğrenme Merkezi (Berkeley Vision and Learning Center) tarafından geliştirilmiş bir python derin öğrenme kütüphanesidir[110, 111]. Caffe, BSD 2-Clause lisansı altında yayınlanmıştır. Caffe, GitHub'daki aktif bir katılımcı topluluğunun yardımı ile Berkeley Görme ve Öğrenme Merkezi (BVLC) tarafından geliştirilmiştir ve bu merkez tarafından korunur ve geliştirilmeye devam etmektedir[110]. Evrişimli sinir ağlarında oldukça iddialıdır[111]. Son derece hızlıdır çekirdek kısmı C++ dilinde yazılmıştır. Yeni uygulamalarda, akademik çalışmalarda ve endüstriyel uygulamalarda kullanılmaktadır. Bu nedenle Caffe oldukça hızlıdır. Caffe, bilgisayarla görme uygulamaları için tasarlanmıştır. Caffe Model Zoo, önceden eğitilmiş farklı görevler birçok model içerir. Caffe, C ++ API'si için desteğe sahiptir.

3.4.2. Keras kütüphanesi

Keras kütüphanesi, Python programlama dilinde yazılmış ve TensorFlow, R, CNTK veya Theano programlama dili ile birlikte çalışabilen üst düzey programlama için geliştirilmiş bir sinir ağları API'sidir[112]. Hızlı deneyler sağlamaya odaklanarak Google mühendisi olan Francois Chollet tarafından geliştirilmiştir[112, 113]. ESA'ları, Yinelenen Sinir Ağları ve ikisinin kombinasyonlarını destekler. CPU ve GPU'da sorunsuz çalışabilir. Yeni başlayanlar için kolay ve hızlı modelleme sağlayan kullanıcı dostu bir kütüphanedir. İzin verilen MIT lisansı altında kullanıma sunulmuştur.

3.4.3. Tensorflow kütüphanesi

TensorFlow kütüphanesi, veri akışı grafiklerini etkin olarak kullanan sayısal hesaplamalar için geliştirilmiş açık kaynaklı bir DÖ kütüphanesidir. TensorFlow, Google Beyin Ekibi tarafından Google'ın Makine İstihbaratı araştırma organizasyonu içerisinde makine öğrenmesi ve derin sinir ağları araştırması için geliştirildi. Şubat 2017'de 1.0 versiyonu kullanıma sunuldu. 21.000'in üzerinde çalışanı ve çoğu dışarıdan gelen katkılarla hızlı bir gelişme kaydetti. TensorFlow çapraz platformdur. Neredeyse her şeyde çalışır: GPU'lar ve CPU'lar (mobil ve gömülü platformlar dahil) ve hatta tensör matematiği için özel donanım olan tensör işleme üniteleri (TPU'lar). Tensorflow, C ++ dilinde geliştirilmiş yüksek performanslı çekirdeğe sahiptir. Kullanımı kolay dostu bir kütüphane olması, Python programlama dili kullanıcıları için öğrenmesi kolay olması, DÖ modellerini çalışırken görmek için TensorBoard aracının olması, Google ve diğer şirketlerden uzman mühendislerin oluşturduğu topluluk desteğinin olması, mobil cihazlarda TensorFlow modellerini çalıştırmak için TensorFlow Lite'ı kullanılabilme olanağının olması ve Tensorflow.js , javascript kullanarak tarayıcıda gerçek zamanlı derin öğrenme modelleri çalıştırmaya olanak tanınması gibi avantajları vardır. TensorFlow, MxNet ve CNTK gibi yapılara kıyaslandığında biraz yavaş olması, OpenCL desteğinin olmaması ve hata ayıklamanın zor olması gibi olumsuz tarafları da vardır.

3.4.4. Theano kütüphanesi

Theano, MILA grubu tarafından BSD lisansı altında yayınlanan açık kaynak kodlu geliştirilmiştir. Theano kütüphanesi Montreal Üniversitesi'nde Yoshua Bengio

başkanlığında yürütülen DÖA'ların incelenmesi, araştırılması ve geliştirilmesi amacıyla LISA grubu (şimdi MILA) tarafından geliştirilmiştir[111]. Theano, 2007 den beri büyük ölçekli hesaplamalarda kullanılmaktadır. Theano, CPU veya GPU'da çalıştırılabilen hızlı sayısal hesaplamalar için geliştirilmiş bir Python kütüphanesidir[114, 115]. Theano kütüphanesi, matematiksel ifadeleri başarılı bir şekilde yapar. Özellikle çok boyutlu matrisleri ve zor matematiksel ifadeleri kolay bir biçimde tanımlar, üst düzey işlemleri içeren ifadeleri tanımlar, optimize eder değerlendiren bir Python kütüphanesidir[115].

3.4.5. Torch kütüphanesi

Torch, GPU'ları ilk sıraya koyan, açık kaynak kodlu makine öğrenme algoritmalarına geniş desteği olan bilimsel bir bilgi işlem kütüphanesidir[116]. Kolay ve hızlı bir betik dili olan LuaJIT ve bunun altında yatan bir C / CUDA uygulaması sayesinde kullanımı kolay ve verimlidir[116, 117]. Torch'un amacı, süreci basit hale getirirken bilimsel algoritmaları oluştururken maksimum esnekliğe ve hıza sahip olmaktır. Torch, makine öğrenmesi, bilgisayarla görme, sinyal işleme, paralel işleme, görüntü, video, ses ve diğerleri arasında ağ oluşturma ve Lua topluluğunun yaptıkları üzerine inşa edilmiş paket ekosistemiyle birlikte gelir[116]. Torch'un merkezinde, karmaşık sinir ağı topolojilerinin uygulanmasında maksimum esnekliğe sahip, kullanımı kolay olan popüler sinir ağı ve optimizasyon kütüphaneleri yer almaktadır[116, 118]. Sinir ağlarını oluşturabilir ve bunları CPU ve GPU'lar üzerinden verimli bir şekilde çalıştırabilir. Bu kütüphane BSD lisansı altında serbestçe kullanılabilir ve web üzerinden herkes tarafından alınabilir[118].

3.4.6. Digits kütüphanesi

NVIDIA, DÖ GPU Eğitim Sistemi (NVIDIA Deep Learning GPU Training System (DIGITS)), DÖ'nün gücünü mühendislerin ve veri bilimcilerinin hizmetine sunar. Digits kütüphanesi, görüntü sınıflandırma, bölümlendirme ve nesne algılama görevleri için son derece hassas derin sinir ağlarını (DSA) hızlı bir şekilde eğitmek için kullanılabilir[119]. DIGITS, verileri yönetme, çoklu GPU sistemlerinde sinir ağları tasarlama, eğitme, gelişmiş görselleştirmelerle gerçek zamanlı performansı izleme ve dağıtım için sonuç tarayıcısından en iyi performans gösteren modeli seçme gibi genel derin öğrenme görevlerini basitleştirir[119]. DIGITS tamamen kullanıcı ile etkileşimlidir. Böylece kullanıcılar programlama ve hata ayıklama yerine ağları tasarlamaya ve eğitmeye odaklanabilir[119].

3.4.7. Deeplearning4j kütüphanesi

Deeplearning4j (DL4J) kütüphanesi Scala, Java ve Java Sanal Makinası için yazılmış ilk ticari nitelikli, açık kaynak kodlu ve dağıtılmış bir derin öğrenme kütüphanesidir[120]. DL4J, Adam Gibson tarafından yönetilen San Francisco ve Tokyo merkezli bir makine öğrenme grubu tarafından geliştirilmiştir. Apache 2.0 Lisansı altında yayınlanan açık kaynaklı bir derin öğrenme kütüphanesidir. DL4J, kısıtlı Boltzmann makinesi, derin inanç ağları, derin otomatik kodlayıcılar, yığılmış dengeleyici otomatik kod çözücünün ve özyinelemeli sinir tensör ağının, word2vec, doc2vec ve GloVe'nin uygulamalarını içerir. Bu algoritmaların tümü, Hadoop ve Apache Spark ile entegre olan DL4J, derin öğrenme algoritmaların çözümünde dağıtılmış GPU'lar ve CPU'ları çalıştırmak için kullanılır[120, 121]. Ölçeklenebilir ve büyük veriler kolayca işlenebilir, Apache Spark ile entegrasyon kolaylığı sunar, topluluk desteği vardır ve iyi bir dokümantasyonu vardır. Java programlama diliyle sınırlıdır.

DSA'larının programlarının geliştirilmesinde ve yazılımında, programcılara büyük kolaylıklar sağlayan DÖ kütüphaneleri geliştirilmiştir. Farklı ihtiyaçları karşılamak için yeni kütüphaneler de geliştirilmektedir. Çizelge 3.5'de yaygın olarak kullanılmakta olan DÖ kütüphaneleri verilmiştir. Bu DÖ kütüphanelerinin üreticisi, başlangıç sürümü tarihi, yazılım lisansı yazılım dili, açık kaynak kodlu olup olmadığı, hangi platformlarda çalıştığı, hangi dil ile yazıldığı, arayüzü OpenMP desyeğinin olup olmadığı, OpenCL desteğinin olup olmadığı, CUDA desteğinin olup olmadığı gibi özellikler ve bu özelliklerinin birbirleri ile karşılaştırmaları gösterilmiştir.

Çizelge 3.5. DÖ kütüphanelerinin özelliklerinin birbirleri ile karşılaştırılması

 <https://www.aisoma.de> **Comparison of common Deep Learning Frameworks** [Updated: Sep/2019]

Framework	Creator	Initial Release	Software license	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation	Pretrained models	Recurrent nets	Conv. nets	RBM / DBNs	Parallel exec.	Active Dvpt.
Caffe	Berkeley Vision and Learning Center; engineering team; DeepLearning4J community; originally Adam Gibson	2013	BSD	Yes	Linux, macOS, Windows	C++	Python, MATLAB, C++	Yes	Under dvpmt.	Yes	Yes	Yes	Yes	Yes	No	?	
DeepLearning4J		2014	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	No	Yes	Computational Graph	Yes	Yes	Yes	Yes	Yes	
Keras	François Chollet	2015	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano	Can use Theano, Tensorflow or PlaidML	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
MATLAB + Deep Learning Toolbox	MathWorks		Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	CUDA code with GPU Coder	No	Yes	Yes	Yes	No	With Parallel Computing Toolbox	Yes
Microsoft Cognitive Toolkit (CNTK)	Microsoft Research	2016	MIT license	Yes	Windows, Linux (macOS via Docker on roadmap)	C++	Keras, C++, Cmd, B rainScript (.NET on roadmap)	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Apache MXNet	Apache Software Foundation; Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan (Facebook)	2015	Apache 2.0	Yes	Linux, macOS, Windows, AWS, Android, iOS, JavaScript	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PyTorch		2016	BSD	Yes	Linux, macOS, Windows	Python, C, C++	Python, C++	Yes	separately maintained package	Yes	Yes	Yes	Yes	Yes		Yes	Yes
Apache SINGA	Apache Incubator	2015	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python (Keras), C/ C++, Java, Go, JavaScript, R, Julia, Swi	No	Supported in V1.0	Yes	?	Yes	Yes	Yes	Yes	Yes	
TensorFlow	Google Brain	2015	Apache 2.0	Yes	Linux, macOS, Windows, Android	C++, Python, CUDA	C++, Java, Go, JavaScript, R, Julia, Swi	No	On roadmap but already	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Theano	Université de Montréal	2007	BSD	Yes	Cross-platform	Python	Python (Keras)	Yes	Under dvpmt.	Yes	Yes	Lasagne's model zoo	Yes	Yes	Yes	Yes	No
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	2002	BSD	Yes	Linux, macOS, Windows, Android, iOS	C, Lua	Lua, LuaJIT, C, utility library for C++/OpenCL	Yes	Third party implementations	Yes	Twitter's Autograd	Yes	Yes	Yes	Yes	Yes	No

Source: Wikipedia



4. ARDIŞIK SAYISAL GÖRÜNTÜLERDE BULUNAN ÇOK SAYIDA HAREKETLİ NESNENİN TESPİTİ, İZLENMESİ, TANIMLANMASI VE SINIFLANDIRILMASI

Tez çalışmasının amacı kapsamında yapılan çalışmalar sonucunda özgün bir Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Gerçek Zamanlı Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması (NTİTS) yöntemi geliştirilmiştir. Geliştirilen NTİTS yöntemi ile analizi edilecek olan sayısal görüntüler daha önceden kayıt edilmiş sayılar görüntüler veya sayısal kameralardan gerçek zamanlı olarak alınan ardışık sayısal görüntüler olabilir. NTİTS yöntemi, önceden kayıt edilmiş sayısal görüntülerden ya da sayısal kameralardan gerçek zamanlı olarak alınan ardışık sayısal görüntülerde hareket edebilen canlı, cansız, endüktif, kapasitif özellikteki her türlü hareketli nesneyi, hareketsiz nesnelerden ve arka plandan ayırıştırarak hareketli nesneleri tespit edip, tespit edilen bu hareketli nesnelerin izlenmesini, tanımlanmasını ve sınıflandırılmasını yapmaktadır. Tez çalışması boyunca yapılan çalışmalar sonucunda geliştirilen özgün NİTİTS yöntemi iki aşamalı bir yöntemdir. İki aşamalı NİTİTS yönteminde öncelikle OA algoritmasını kullanarak ardışık sayısal görüntülerde piksellerin ışıklılık değerlerindeki değişimleri hesaplanarak, ışıklılık değerlerinde değişim olan pikselleri bulunur. OA algoritmasından elde edilen hareketli nesnelere ait bu pikselleri, hareketli nesnelere ait hız vektörleri olarak ifade eder. OA'dan elde edilen hız vektörleri NİTİTS yöntemi ile anlamlandırılır ve gruplandırır. Işıklılık değerlerinde değişim olan bu pikseller grupları, giriş görüntüsü olan sayısal görüntülerde hareketin olduğu, hareketli nesnelere ait görüntülerine karşılık gelir. Hareketli nesnelere ait bu görüntüler giriş görüntüsünden her zaman daha küçüktür. Giriş görüntüsünde birden fazla hareketli nesne olabilir. Geliştirilen NİTİTS yöntemi ile giriş görüntüsüne sığabilen maksimum sayıdaki hareketli nesne tespitinde OA ile tespit edilebilmektedir. Tespit edilen bu hareketli nesnelere ait görüntüler DSA ile işlenerek tanımlanması ve sınıflandırılması yapılır. DSA'lar sayısal görüntüleri birçok katmanda kullanılan hesapları tek bir seferde yapan, tanımlanması gereken parametreleri bile kendisi keşfedebilen, belki de daha iyi parametreler ile değerlendirmeler yaparak sayısal görüntülerdeki nesnelerin tanımlanmasını ve sınıflandırmasını görüntünün bir kısmı olmasa bile insan algısına yakın doğrulukta yapar. NİTİTS yöntemi ile tanımlama ve sınıflandırmada DSA olarak Liu ve ark. tarafından gerçek zamanlı nesne algılama tekniği olarak Kasım 2016 yılında önerilen [122] Tek Atış Çoklu Kutu Algılama (Single Shot

Multibox Detector) (TAÇKA) kullanılmıştır. TAÇKA tek bir seferde çoklu kutu algılama ile VGG-16 ağını temel alarak sayısal görüntülerdeki nesnelerin tanımlamasını ve sınıflandırmasını yapar.

Görüntü işleme uygulamalarının en büyük problemlerinden birisi de işlenecek olan görüntünün çözünürlüğüdür. Çünkü sayısal görüntülerde çözünürlük arttıkça sayısal görüntünün kalitesi ile birlikte işlenecek olan sayısal verinin miktarının artmasıdır. Dolayısıyla yüksek çözünürlüklü sayısal görüntülerin analizi için yüksek özellikli donanımlar gerektirmesidir. Örneğin 4K çözünürlüğüne sahip renkli sayısal görüntünün, her bir görüntü karesinde $3840 \times 2160 \times 3 = 24\,883\,200$ adet piksel bulunmaktadır. Ayrıca saniyede $60fps$ görüntü alabilen bir sayısal kamera ile çalışılıyorsa $1sn$ 'lik görüntüde $24\,883\,200 \times 60 = 1\,492\,992\,000$ adet piksel bulunur. Her bir piksel 1Byte'lık bir veri içerdiği için bir saniyelik görüntü $1\,492\,992\,000$ Byte lık sayısal veriye karşılık gelmektedir. Bu kadar büyük verinin özellikle gerçek zamanlı analizi için yüksek özellikli pahalı sistemler gerektir. Büyük veri yığınlarının işlenmesi sistem gereksinimleri ve işlem süresi bakımından her zaman mümkün olmayabilir[123].

Sayısal görüntülerin çözünürlüğü ne kadar yüksek ise görüntünün alındığı ortam ile ilgili o kadar çok bilgi içerir. Ortam ile ilgili ne kadar çok bilgi varsa görüntünün işlenmesinden elde edilecek veride o kadar sağlıklı olur. Bu nedenle geliştirilecek olan görüntü işleme yöntemleri, giriş görüntüsünde küçültme işlemine tabi tutmadan (veri kaybına sebep olmayacak şekilde analiz edilip faydalı bilgiyi çıkarma) yapması çok önemlidir. Ancak görüntü işleme problemlerine getirilen çözümler incelendiğinde yüksek çözünürlüklü sayısal görüntülerin analizi yapılırken genellikle bu görüntülerin yeniden boyutlandırılarak küçültüldüğü görülmektedir. Hatta bazı uygulamalarda, işlem adımları sırasında görüntüler birkaç kez daha küçültülür. Görüntü küçültme işlemi sonucunda sayısal görüntüye ait pek çok faydalı bilgi azalır ya da tamamen yok olur. Oysaki sayısal görüntüler küçültülmeden görüntünün ilgili kısmı yani görüntülerdeki ilgi alanları tespit edilip sadece ilgi olan faydalı bilginin olduğu kısım ya da kısımlar tespit edilip bu kısım ya da kısımlar ana görüntüden ayrıştırılarak bu kısım ya da kısımlar analiz edilirse küçültme işlemine gerek kalmadan analiz edilebilir. Böylelikle görüntülerdeki önemli bilgileri içeren kısımlarda ilgili hiçbir veri kaybı olmaz. Bilgi kaybı olmadan analiz işlemi yapılacağı için analiz sonucundan elde edilen bilgiler çok daha sağlıklı olacaktır. Böylece sistemin güvenilirliği artacaktır. Bu nedenle geliştirilecek olan görüntü işleme yöntemleri, giriş görüntüsünü küçültme işlemi yapmadan

sistem gereksinimlerini en alt düzeyde tutarak işlemleri yapması geliştirilen yöntemin güvenilirliğini arttırırken sistemi gerekliliklerini ve maliyeti azalması bakımından da oldukça önemlidir.

Bu tez çalışmasının amacı sayısal görüntülerde hareketli nesnelerin, tespiti, izlenmesi, tanımlaması ve sınıflandırmasında giriş görüntülerini küçültmeden ilgi alanını bulup ilgi alanındaki nesneleri ayrıştırıp tanımlaması ve sınıflandırılmasının yapılması gerekliliği ortaya çıkmıştır. Bu nedenle geliştirilecek olan görüntü işleme sisteminin işlemini nasıl yapılsın ki bu işlemler yapılırken görüntüde veri kaybı olmadan yapılsın problemine çözüm aranmıştır. Yapılan araştırmalar ve çalışmalar sonucunda iki aşamalı bir yöntem ile giriş görüntüsünü küçültmeden hareketli nesnelerin tespiti, izlenmesi takibi ve sınıflandırılması probleminin çözülebileceği bir anlaşılmıştır.

Şöyle ki, giriş görüntüsü olarak kullanılan ardışık sayısal görüntüleri oluşturan tüm pikselleri herhangi bir küçültme işlemine tabi tutmadan giriş görüntüsündeki ilgi alanını, görüntüyü oluşturan tüm pikseller kullanılarak tespit etmelidir. İlgi alanı bulunduktan sonraki hesaplamalarda analize sayısal görüntüyü oluşturan piksellerin hepsi ile kullanılmayacaktır. Yani görüntülerin analizinde ilgi alanlarını tespit edip bu ilgi alanlarını oluşturan bölgeler giriş görüntüsünden tüm ayrıntıları ile alınarak analiz edilecektir. Böylelikle ilgi alanındaki görüntünün tanımlamasını ve sınıflandırmasında görüntü küçültülmeyeceği için bilgi kaybı olmadan yapılacaktır. Sadece ilgi alanına giren görüntüler analiz edileceği için nesnelerin tanımlama ve sınıflandırma işlemlerinin doğruluğu, güvenilirliği artarken sistem gereksinimleri, kurulum maliyetleri ve işlem süreleri azalacaktır.

Geliştirilen NTİTS yönteminde ilgi alanı harekettir. Bu nedenle öncelikle ardışık sayısal görüntülerden hareket tespiti yapılmalıdır. Hareket tespiti yapıldıktan sonra hareketli nesnelere ait piksellerde tanımlama ve sınıflandırma işlemi yapılmalıdır. Böylelikle geliştirilecek olan yöntemin problemi, etkin bir şekilde çözeceği anlaşılmıştır. İlgi alanı olan hareket tespit edildikten sonra sadece hareketli nesnelere ait görüntülerde herhangi bir küçültme işlemi yapılmayacağı için hem tanımlama ve sınıflandırma işleminin güvenilirliği daha çok daha yüksek olacaktır. Bu nedenle öncelikle ardışık sayısal görüntülerde hareket eden nesnelerin tespitinin nasıl yapılacağı araştırılırken sistemin çalışmasında insan gözünün çalışması prensibinden esinlenilmiştir. İnsan gözü baktığı noktada her yeri eşit çözünürlükte

görmez. İşte insan gözünün bu özelliğinden ilham alarak özgün NİTİTS yöntemi geliştirilmiştir.

4.1. Doğayı Taklit, Canlıyı Taklit (Biomimicry, Biomimetic)

Doğa, tarih boyunca sanatçılar, tasarımcılar, mühendisler, bilim adamları ve bir parçası olan barındırdığı tüm canlılar için her zaman problemlerin çözümünde en önemli bir esin kaynağı olmuştur. Doğada bulunan esin kaynağı olan canlı cansız tüm varlıklar biomimikri, biyomimetik biliminin konusuna girer ve bu başlık altında incelenir.

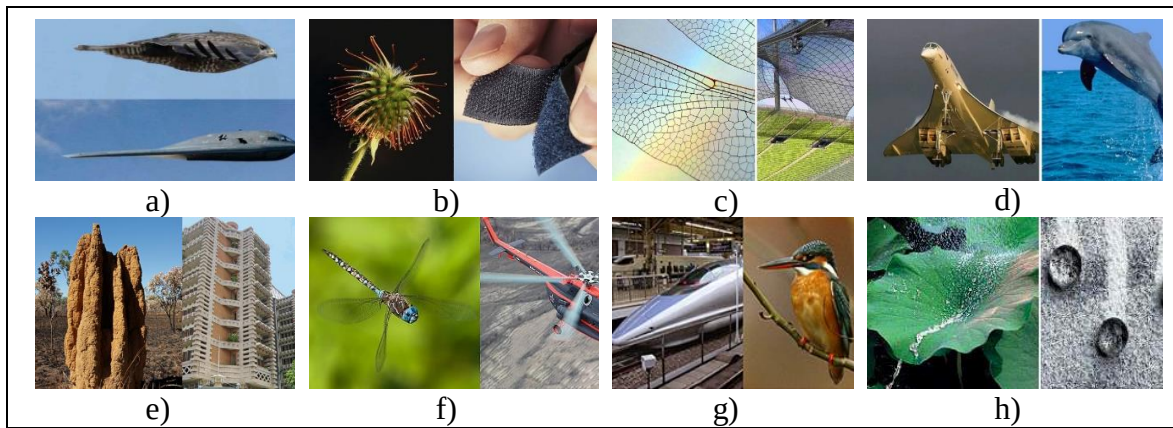
Biomimicry (Biyomimikri) terimi bio Yunanca hayat anlamına gelen ve mimicry Yunanca benzetim anlamına gelen kelimelerinin bir araya getirilmesiyle oluşmuştur. Bu iki kelimenin birleştirilmesiyle de hayatı taklit, yaşamı taklit, hayattan taklit anlamında Biyomimikri kelimesi üretilmiştir[124, 125]. Doğa, yüzyıllardır tasarımcılar ve bilim insanları tarafından, büyük bir ilham kaynağı, en iyi, en etkili yenilik kaynağı olarak kabul edilir[126, 127]. Biyomimikri, doğal tasarımların, süreçlerin ve sistemlerin incelenmesi yoluyla insan sorunlarına çözüm için ilham alınan uygulamalı bir bilimdir[128].

Biyomimikri; doğadaki modelleri inceleyerek doğadan öğrenen; doğada var olan formları, süreçleri, sistemleri model alan ve doğanın tasarımlarını taklit ederek insanlığın sorunlarına ekonomik ve sürdürülebilir çözümler geliştiren bir bilim dalıdır. Doğaya dostu, doğadan beslenen ve doğaya adaptasyonunu sağlamış çözümler nelerdir nasıl olabilir sorusunun cevabını Janine Benyus 1997 yılında yayınladığı “ Biomimicry: Inspired by Nature” adlı kitabında cevabın doğanın kendisinde olduğunu belirtmiştir[129]. Benyus kitabında dünyanın 3,8 milyar yaşında olduğundan ve 8,3 milyar yıldır nesilden nesile aktarılan, bir kısmı da deneme, yanılma yoluyla daha kullanışlı ve ekonomik ve hayatta kalma mücadelesine biyomimikri bilimine katkı yaptığını söylemektedir[129]. Yine Benyus kitabında dünyanın 3,8 milyar yıl boyunca oluşturduğu olağanüstü zeki bir gezegen olduğunu ve içinde barındırdığı canlıları yeniden incelemenin ve keşfetmenin insanlığın problemlerine yenilikçi çözümler üreteceğini iddia etmektedir.[129]. Benyus’a göre biyomimikri; yüz yıllardır doğadaki canlıların hayatta kalma mücadelesi boyunca deneme yanılma yolu da dahil tüm yetenekleri ile geliştirildikleri bilgi birikiminden esinlenerek daha yenilikçi daha çevre dostu ve daha ekonomik teknolojilerle ilgisi olan biyomimikri bilimi,

aslında doğadaki tüm canlıların kendi sorunlarına ürettiği çözümleri, insanlığın ihtiyaçlarına yönelik olarak üretilen malzemelerin geliştirilmesini tarif eder [129].

Biyomimetik terimini, biyomedikal mühendisliği gibi bir bilim dalının oluşturmaya öncülük eden ve bu yönüyle bilinen Amerikalı bilim insanı olan Otto H. Schmitt, doktora araştırmasında bir sinirin elektriksel etkisini açıkça taklit eden fiziksel bir cihaz üretme girişimini 1957 yılında ortaya koymuştur[130]. Otto Schmitt, 1969 yılında bir makalesinde biomimetik terimini ilk defa kullandı ve Webster's Dictionary'de 1974 yılında alındı[130]. Biyolojik olarak esinlenen tasarım veya doğadan adaptasyon veya türetme 'biyomimetik' olarak adlandırılır. Biyolojiyi veya doğayı taklit etmek anlamına gelir. Biyomimetik, Yunanca biomimesis kelimesinden türetilmiştir.

Biyomimikri ve biyomimetik kelimelerinin arasında özü itibari ile anlam farkı bulunmaktadır. Biyomimikri, tabiatın varoluşunda beri test ettiği canlıların modellerini ve stratejilerini örnek alarak, insanlığın sorunlarına karşı sürdürülebilir, ekonomik çözümler arayan yaklaşımdır. Biyomimetik, canlıların yapı ve işlevlerinin, teknolojik ihtiyaçlara uygun ürün oluşturma modelleri olarak kullanılmasıdır. Biyomimikriye olan ilgi, sorunları en pratik ve en verimli şekilde çözmesiyle her geçen gün artarak devam etmektedir.



Resim 4.1. Doğadan esinlenen tasarımlar görülmektedir.

Resim 4.1 a), b), c), d), e), f), g), h)'de Doğadan esinlenen bazı canlı türlerinden ilham alınan tasarımlar görülmektedir. Bu tasarımlar sayesinde canlıların çevresel şartlara uyum sağlamıştır ve hayatta kalma mücadelesinin sonucunda kendilerine üstün özellikler kattığı bazı örnekler görülmektedir.

Resim 4.1 a)'da Northrop Grumman firması tarafından üretilen B2 Spirit ağır bombardıman uçağının tasarımı ile alaca şahinin uçarken ki görüntüleri arasındaki benzerlik şaşırtıcı derecede birbirlerine benzemektedir.

Resim 4.1 b)'de İsveçli mühendis Georges de Mestral 1941'de Alplerde köpeğiyle bir av gezisinden geldikten sonra köpeğinin üzerine ve kendi giysilerine takılan bitkilerin kancalarından esinlenerek velcro denilen cırt cırt bantları icat etmiştir. O günün şartlarındaki imkanlar yüzünden uzun süren imalat süreci sonunda 1951 yılında patentini almayı başarmıştır.[131-133].

Resim 4.1 c)'de 1968-1972 yılları arasında yapılan Münih Olimpiyat stadının çatısının yapımında Yusufçuk böceğinin kanat yapısından ilham alınmıştır. Yusufçuk böceğinin kanatlarının kalınlığı milimetrenin 1/3 000'i kadardır. Kanat kalınlığı ince olmasına rağmen dayanıklılığını kanatlarının bine yakın bölmelerden çok sayıda bölmeden oluşmasıdır[134].

Resim 4.1 d)'de 1969 yılında İngiliz-Fransız ortak yapımı olan ses hızını aşmayı başaran Concord uçakların yapımında yunusların burunlarının yapısından esinlenilmiştir. Concord uçakların burun yapısının, yunusların burun yapılarına benzetimi sayesinde sürtünme kuvvetleri önemli oranda azaltılmıştır.

Resim 4.1 e)'de Termitler yaptıkları yapılarını duvarların bir dizi ısıtma ve soğutma kanalları açıp bu kanallar sayesinde yaşam alanlarının ısını bu havalandırma deliklerini açıp kapatarak ısı kontrolü yaparlar. İç sıcaklık çok ısındığında, ısınan havanın yükselmesiyle, sıcak hava havalandırma deliklerinden dışarıya atılır böylelikle pasif bir şekilde ısı kontrolü yaparlar[125]. Termitlerden ilham alan mimar Mick Pearce, kendi boyutundaki geleneksel binalara göre havalandırma için %90 daha az enerji kullanan Eastgate Binası'nı tasarladı[135]. Termitler mimari açıdan pek çok tasarımcıya esin kaynağı olmuştur.

Resim 4.1 f)'de Yusufçuk böceğinin dört kanadının her birini bağımsız olarak hareket ettirebilme yeteneği sayesinde yusufçuğun her yöne uymasını etkili süzülme, güçlü yükselen uçuş, çiftleşme sırasında tandem uçuş, düşük hızda manevra, havada süzülme gibi çok çeşitli ve en karmaşık uçuş manevralarını gerçekleştirmesini sağlar en önemlisi, yusufçuklar havada asılı kalabilir [136-138]. Yusufçuklar uçuş sırasında aniden yavaşlayabilir ve

dönebilir. Hızlı bir şekilde hızlanabilir ve hatta geriye doğru uçabilirler[136, 137]. Yusufçukların havada süzölmelerine izin veren daha geniş arka kanatları vardır[136, 138]. Bu üstün uçuş özellikleri sayesinde Yusufcuk böceklerini mühendislik tasarımlarında uçaklar ve helikopterler için çok ilginç ve benzersiz bir biyolojik esin kaynağı olmuştur.

Resim 4.1 g)'de Shinkansen demiryolu projesinin başkanı olan Eiji Nakatsu, aynı zamanda bir kuş gözlemcisidir. Yalıçapkını (kingfisher) kuşları ve penguenlerin gaga yapısının uçarken ve suya dalarken sürtünmeyi azaltıp hızı arttırdığını fark etmiş ve hızlı trenlerin tasarımlarında yalıçapkını kuşların gaga yapısından ilham alarak hızlı trenlerin uç kısmını yeniden tasarlamıştır. Bu tasarım sonucunda trenler tünele girerken veya tünelden çıkarken basıncın neden olduğu patlayıcı ses, trenlerin burun yapısının Yalıçapkını kuşunun burnuna benzetimiyle ses etkisi büyük ölçüde azalmıştır, %15 daha az elektrik kullanır, hatta %10 daha hızlı çalışabilir hale gelmiştir [139-141].

Resim 4.1 h)'de Ohio Eyalet Üniversitesi'nden Prof. Dr. Bharat Bhushan nilüfer çiçeğinin yaprağını inceleyerek yaprak üzerinde bulunan mikron düzeyindeki çıkıntılar sayesinde yaprağın kendini temizleyebildiğini keşfetmiştir. Nilüfer çiçeğinin yapraklarının üzerindeki çıkıntılar sayesinde toz ve kir tutmamasından ilham alarak kendini temizleyebilen camı ve dış cephe boyasını geliştirmiştir[142, 143]. Artık günümüzde kendini temizleyebilme özelliğinden esinlenerek, kendi kendisini temizleme özelliği olan pek çok dış cephe kaplama malzemesi geliştirilmesi gibi örnekler vermek mümkündür.

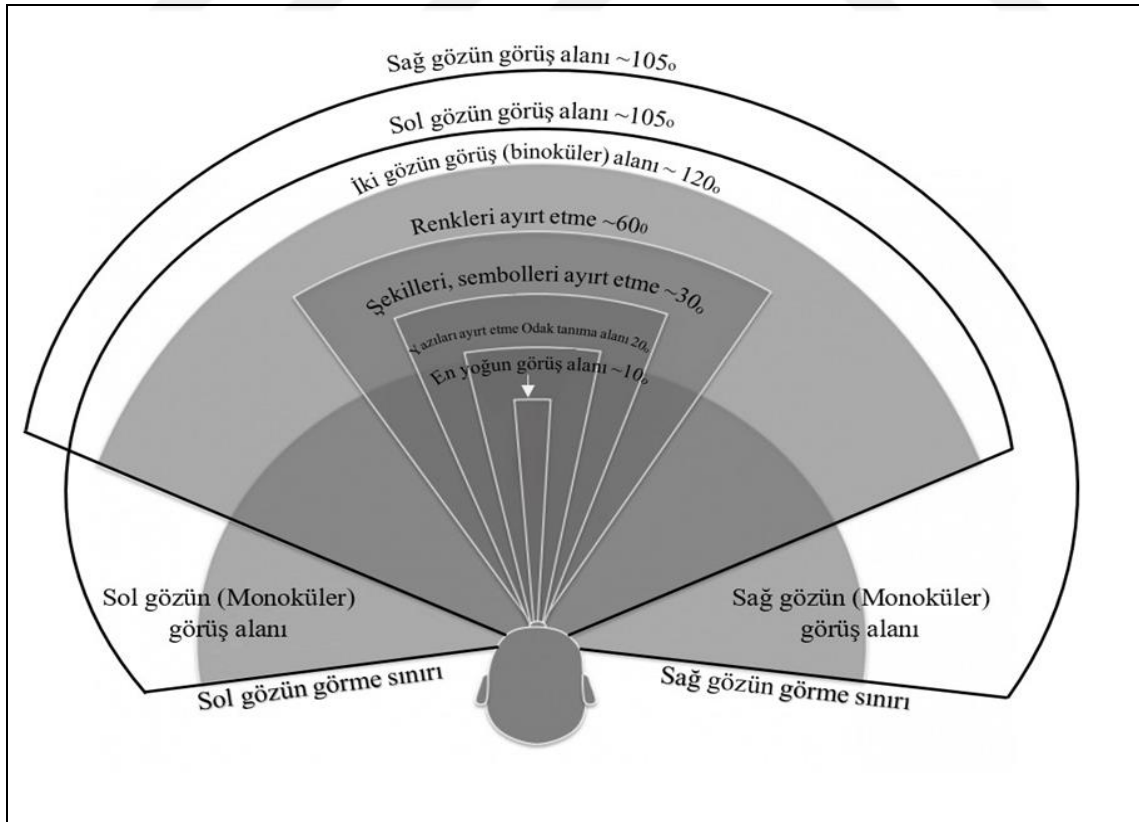
Dikkatli ve ilgili gözlemci ve tasarımcılar doğada karşılaştıkları herhangi bir olumsuz olaydan ya da ihtiyaçtan yola çıkarak insan sorunlarını çözümünde doğadan esinlenerek günümüze değin pek çok tasarım ve buluşlar yapmışlar ve yaptıkları buluşlar alanlarında oldukça başarılı olmuştur. Geliştirdikleri ürünler kaynak isafını önlemekle kalmayıp insanlığın yararına ve çevreye dostu olmuştur. Bu nedenlerden dolayı Biyomimikri ve Biomimetik'e olan ilgi her geçen gün artarak devam etmektedir.

Tez çalışmasının amacı kapsamında geliştirilen özgün NİTİTS yönteminin çalışmasında ardışık sayısal görüntülerdeki ilgi alanı olan hareketli nesnelerin tespit edilmesinde Biyomimetik biliminin içinde yer alan insan gözünün çalışması prensibinden esinlenilmiştir. Şöyle ki insan gözü görüş alanının içindeki her noktayı eşit çözünürlükte görmez. İnsan gözü, görüş alanının merkezinin en yakın çevresini en yüksek çözünürlüklü olarak görür. Görüş

alanının merkezinden dışarıya doğru uzaklaştıkça gözün gördüğü alanın çözünürlüğü azalmaya başlar. İnsan gözünün görüş alanının merkezinden uç kenarlara doğru gidildikçe çözünürlük azaldığından görüş alanının en uç noktalarında insan gözü en az çözünürlükte görür. Görüş alanının merkezinden en uzak bölgelerdeki nesneler siluet halinde görülür. Örneğin topluluk önünde konuşma yapan bir kişi, konuşur iken karşıya baktığında görüş alanının merkez bölgesindeki insanları çok yüksek çözünürlüklü olarak net bir şekilde görür. Görüş alanının merkez bölgesindeki kişilere ait pek çok detayı görebilir. Yani kişilerin gözlüklümü, gözlüksüz mü olduklarını saç şekillerini, saç renklerini, yüz tiplerini, elbiselerine ait özellikleri vb. gibi pek çok detayları görür. Görüş alanının merkezinden en uzak alanlarda yani uç kenarlarında olan kişilere ait özellikleri yani kişilerin gözlüklümü, gözlüksüz mü olduklarını saç şekillerini, saç renklerini, yüz tiplerini, elbiselerine ait özellikleri vb. gibi detayları göremez. Görüş alanından en uzak olan buradaki kişileri siluet halinde görür. Görüş alanını uç bölgelerinde, konuşmacının dikkatini çekecek bir hareketlilik olduğunda konuşmacı gözünü hareketliliğin olduğu dikkatini çeken noktaya kaydırırsa, konuşmacının görüş alanının merkezi değişecektir. Yeni görüş alanının merkezi yani konuşmacının gözünü kaydırıldığı hareketliliğin olduğu bölge olacaktır. Hareketliliğin olduğu bu bölgeyi konuşmacı hemen çok yüksek çözünürlüklü görmeye başlarken, biraz önceki görüş alanının merkezi olan bölgenin çözünürlüğü azalacak ve bu önceki bölgeyi daha düşük çözünürlüklü olarak görecektir. Yeni görüş alanı merkezindeki siluet halinde gözüken kişi ya da kişiler bir anda çok yüksek çözünürlüklü olarak görüldüğü için bu bölgedeki kişilere ait pek çok detayı konuşmacı görecektir. Örneğin hareketlenmenin olduğu alandaki kişi ya da kişiler konuşmacıyı protesto ediyorsa protesto edenleri ve bunların hareketlerini görecektir. Eğer protestocu, konuşmacıya bir yumurta fırlatırsa, konuşmacı kendisine fırlatılan yumurtayı görebilecektir. Kendisine fırlatılan yumurtaya karşı kendini korumaya çalışacaktır. İşte saniyeler içerisinde meydana gelen bu beklenmedik olayda gözün bu çalışma hızı ve beyinle koordineli olarak gördüklerini yorumlaması prensibinden esinlenilerek ardışık sayısal görüntülerdeki hareketli nesnelerin tespit edilip tanımlanmasında, görüntünün tümü ile ilgilenirken görüntüde ihtiyaca cevap verecek olan alanlar belirlenip sadece bu alanlarla çalışılmalıdır. Yani sayısal görüntülerdeki verilerde öncelikle ilgi alanı tespit edilmelidir. İlgi alanına göre ilgi alanı algoritmasından geliştirilmeli ve ilgi alanı algoritması ile ilgi alanları tespit edilip ayrıştırılmalıdır. Daha sonra ayrıştırılan bu ilgi alanları analiz edilmelidir. İlgi alanı algoritması bir elek gibi verileri eleyerek değerli verileri, değersizlerden ayrıştırmalıdır. Ayrıştırılan bu değerli veriler analiz için kullanılmalıdır. Böylelikle gereksiz veri yığınları ile uğraşılacak zorunda

kalınmayacaktır. Tıpkı insan gözün yaptığı gibi, insan gözü, görme alanındaki her yeri eşit çözünürlükte görmediği için göz insan beynini gereksiz veriler ile yormaz Tez çalışmasının amacı kapsamında geliştirilen özgün NTİTS yöntemi ile insan gözünün çalışma özelliği modellenmiştir.

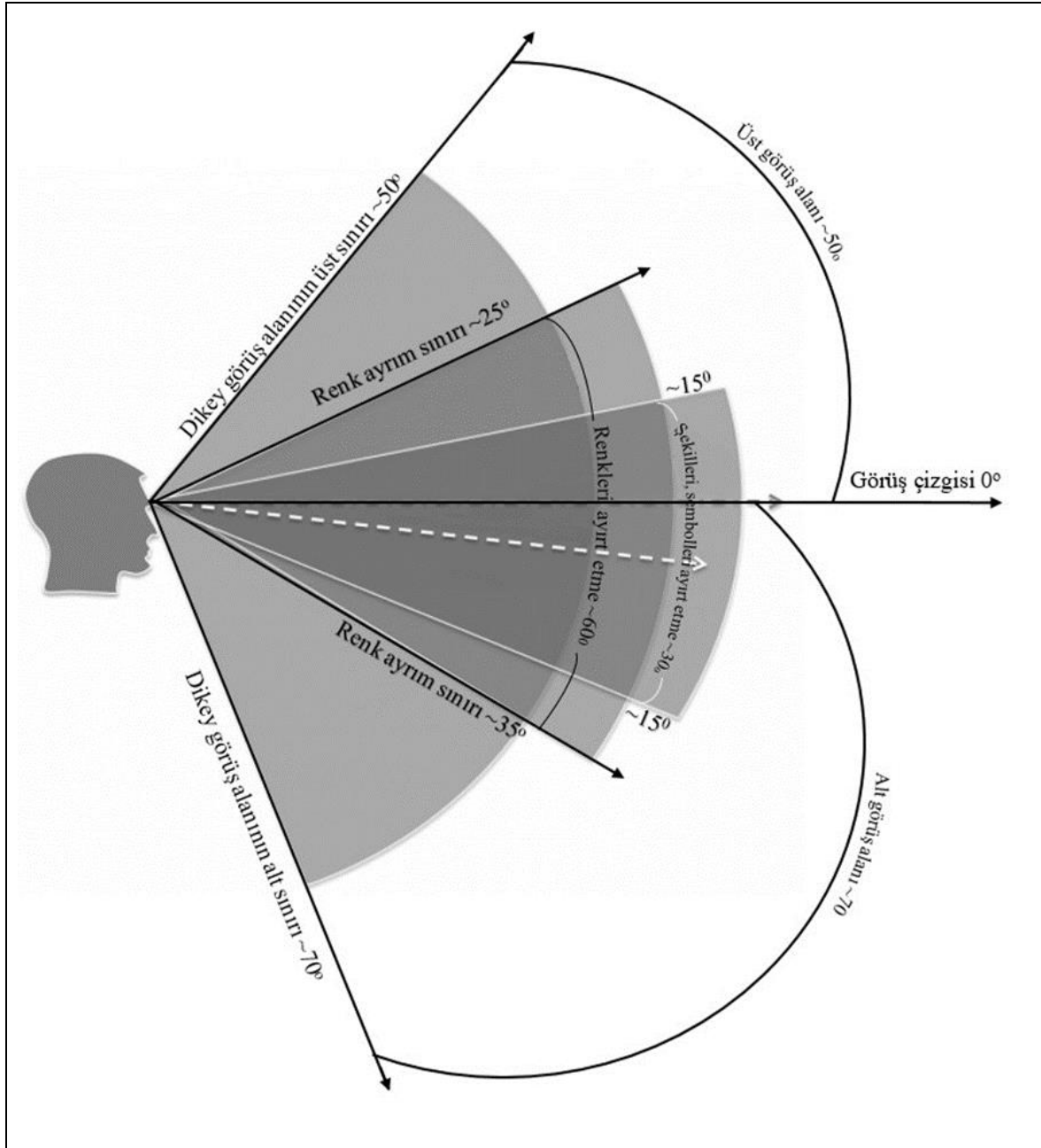
Şekil 4.1 ve Şekil 4.2’de insan gözünün yatay ekseninde ve dikey ekseninde görüş açıları verilmiştir, insan gözünün görme alanları ve görme açıları incelendiğinde tez çalışması kapsamında geliştirilen özgün NTİTS yönteminin daha iyi anlaşılacaktır. Şekil 4.1 ve Şekil 4.2’de insan gözünün görüş alanında yatay ve dikey ekseninde gördüğü alanlar göz ve kafa hareket ettirilmeden gözlerin her ikisinin de düz olarak görüş çizgisi boyunca karşıya baktığında gördüğü alanlar ve bu alanların gözün görme merkezi ile olan açıları ve görüş alanlarının özellikleri gösterilmiştir. Şekil 4.1 ve Şekil 4.2’de gösterilen şekillerdeki görme alanları ve görme açıların değerleri, gözün görme yeteneğine göz çevresinin yapısına ve ırklara bağlı olarak azda olsa farklılıklar gösterebilir. Tek gözün görme alanına “monoküler görme alanı” denir. İki gözün görme alanına ortak görme alanı ya da iki gözün kesiştiği görme alanı “binoküler görme alanı” denir[144].



Şekil 4.1. İnsan gözünün yatay eksenindeki görüş açılarına göre görüş netlikleri[144-148]

Şekil 4.1’de belirtildiği gibi, sol göz için yatay ekseninde sol gözün monoküler görüş alanı yaklaşık olarak 105° ’lik bir görüş alanına sahiptir. Şekil 4.1’de belirtildiği gibi, sağ göz için yatay ekseninde sol gözün monoküler görüş alanı yaklaşık olarak 105° ’lik bir görüş alanına sahiptir. İnsan gözleri sağ ve sol olmak üzere ikisi birlikte koordineli çalışır. Gözlerin baktığı bölgedeki bir nesneyi aynı anda gözlemlediği alana, merkezi görüş alanı denir. Bu merkezi görüş alanı, “dürbün alanı” olarak da adlandırılır. Şekil 4.1’de belirtildiği gibi, iki gözün ortak olarak gördüğü alanlar vardır. Her bir gözün yatay ekseninde ortak olarak gördüğü bu binoküler alan yaklaşık olarak 120° ’dir[144-148].

İnsan gözü iki gözün görme alanı olan binoküler alanı içindeki bölgenin her yerini eşit çözünürlükte görmez. Bu alandaki insan gözünün görme keskinliği merkezden 30° ’ye kadar her $2,5^{\circ}$ ’de yaklaşık olarak %50 azalırken, bunun ötesinde daha hızlı azalır[147]. Renk algısı 20° ’de güçlü ancak 40° ’de zayıftır, bu nedenle her bir göz için 30° tipik olarak renk algısında bir eşik olarak kabul edilir[145, 147]. İnsan gözünün bir öğeyi “gerçek” olarak ayrıntılı algılaması ve üzerine odaklanması 60° ’lik bir açı (ksenin merkezinden göz başına 30°) alanında meydana gelir. Bu 60° ’lik açının kapsadığı alanda renkler ayırt edilebilir. 60° ’lik bu alanda da İnsan gözünün görme yeteneğinde farklılıklar vardır. Açı küçüldükçe görme netliği artar. 30° ’lik alanda yazıları sembolleri ayırt eder. 30° ’lik alanın içinde de 20° ’lik alanda insan gözü odaklanarak en yoğun olarak gördüğü alandır. 20° ’lik alanın içinde gördüğü 10° ’lik alan insan gözünün gördüğü en yoğun alandır[144, 145, 147].



Şekil 4.2. İnsan gözünün dikey eksenindeki görüş açılarına göre görüş netlikleri[144-148]

Şekil 4.2’de insan gözünün dikey eksenindeki görüş açılarına göre görüş açıları gösterilmiştir. İnsan gözünün dikey görüş alanında ise insan gözünün görme yeteneğine göz çevresinin yapısına ve ırklara bağlı olarak azda olsa farklılıklar gösterebilir. İnsan gözünün her ikisinin de gördüğü üst sınır, normal görüş hattı çizgisinden yaklaşık 50° yukarıdadır. . İnsan gözünün her ikisinin de gördüğü alt sınır, normal görüş hattı çizgisinden dikey eksenin yaklaşık 70° aşağıdadır. İnsan gözü dikey eksenin, iki gözün ve başın hareket ettirilmeden ve her iki gözünde düz olarak karşıya bakarken yaklaşık 120°’lik bir açıda görür[144, 145, 147]. İnsan gözünün dikey eksenin görmüş olduğu 120°’lik bu açının kapsadığı alanın her

yerini eşit çözünürlükte görmez. Dikey ekseninde, görüş çizgisinden yaklaşık 25° yukarıda ve 35° aşağıda ki açının kapladığı yaklaşık 60° 'lik alanda renk ayrımını yapabilir. Dikey ekseninde, görüş çizgisinden yaklaşık 25° yukarıda ve 35° aşağıda ki açının kapladığı yaklaşık 30° 'lik alanda sembolleri ve şekilleri tanır.

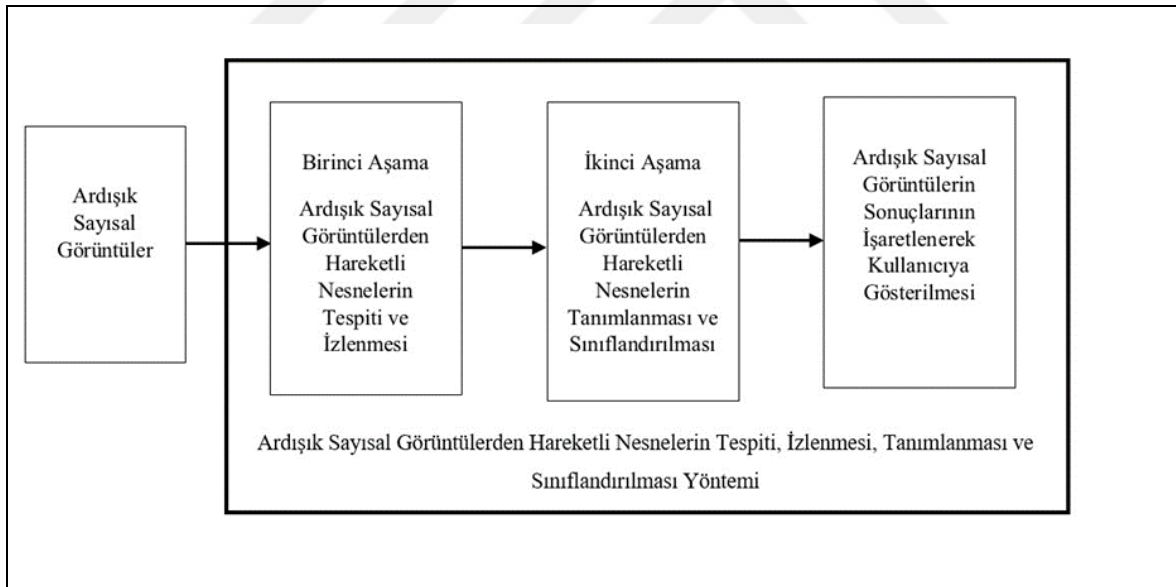
İnsan gözü, sadece odaklandığı merkez noktanın en yakın çevresini en yüksek çözünürlükte görürken odaklandığı merkez noktadan kenarlara doğru uzaklaştıkça gözün gördüğü görüntünün çözünürlüğü azalmaya başlar. Gözün gördüğü alanların çözünürlüklerindeki değişimlerde yumuşak bir geçiş vardır. İnsanlar normal hayatta istemli veya istemsiz olarak sürekli hareket halindedir. Kafanın hareketi, göz kapaklarının hareketi ve gözünün hareketi ile görüş alanı dinamiktir sürekli değişir. İnsan oturuyorken bile istemli veya istemsiz olarak gözlerini, göz kapaklarını ve kafasını hareket ettirerek görüş alanında gördüğü görüntüyü sürekli olarak yeniler.

Şekil 4.1. ve Şekil 4.2’de gösterilen şekillerden de anlaşılacağı üzere insan gözlerinin baktığı alanda gördüğü bölgeler, insan gözünün beyni gereksiz bilgilerle yormaması sadece görüş alanı içinde ilgilendiği alana yönelerek bu alana odaklanır ve bu alandaki tüm detayları alarak bu gölgedeki görüntüyü analiz etmesi için beyne iletir. Beyin gözden gelen görüntü verilerini işleyerek ne gördüğünü anlamaya çalışır. İlgi alanı ile ilgili çok fazla bilgi olduğu için en çok bu alan ile ilgili bilgi edinir. İnsan beyni gördüğü sahneden edindiği bu bilgilere göre kendisine bir sonuç çıkarır ve bu sonuca göre davranış ya da algı üretir. İşte insan gözünün bu çalışma prensibi bu tez çalışmasına esin kaynağı olmuştur. Yani insan gözü gereksiz bilgilerle beyni yormaz. İnsan gözünün çalışmasının verdiği ilhamla, bilgisayarla görü konusu içine giren özgün NİTİTS yöntemi geliştirilmiştir. Geliştirilen NİTİTS yöntemi, sayısal kameradan aldığı sayısal görüntülerde ilgi alanı olan bölgeleri tespit ederek ana görüntüden ayırır. Ayırttığı ilgi alanı olan bu görüntüleri analiz eder. İlgi alanı olmayan görüntünün diğer kısımları ile analiz aşamasında yani nesnelerin tanımlanması ve sınıflandırılması sırasında ilgilenmez. Analiz aşamasında ilgi alanı olmayan kısımlar hesaba katmadığı için işlenecek veri miktarı azalır. Böylece gereksiz verilerle işlemci çalışmaz bu nedenle de sistem gereksinimleri, kurulum maliyetleri ve işlem süreleri azalır.

Ardışık Sayısal Görüntülerde Aynı Anda Birden Çok Hareketli Nesnenin Gerçek Zamanlı Tespiti, tanımlanması ve Sınıflandırılması (NTİTS) Algoritmasının Oluşturulması

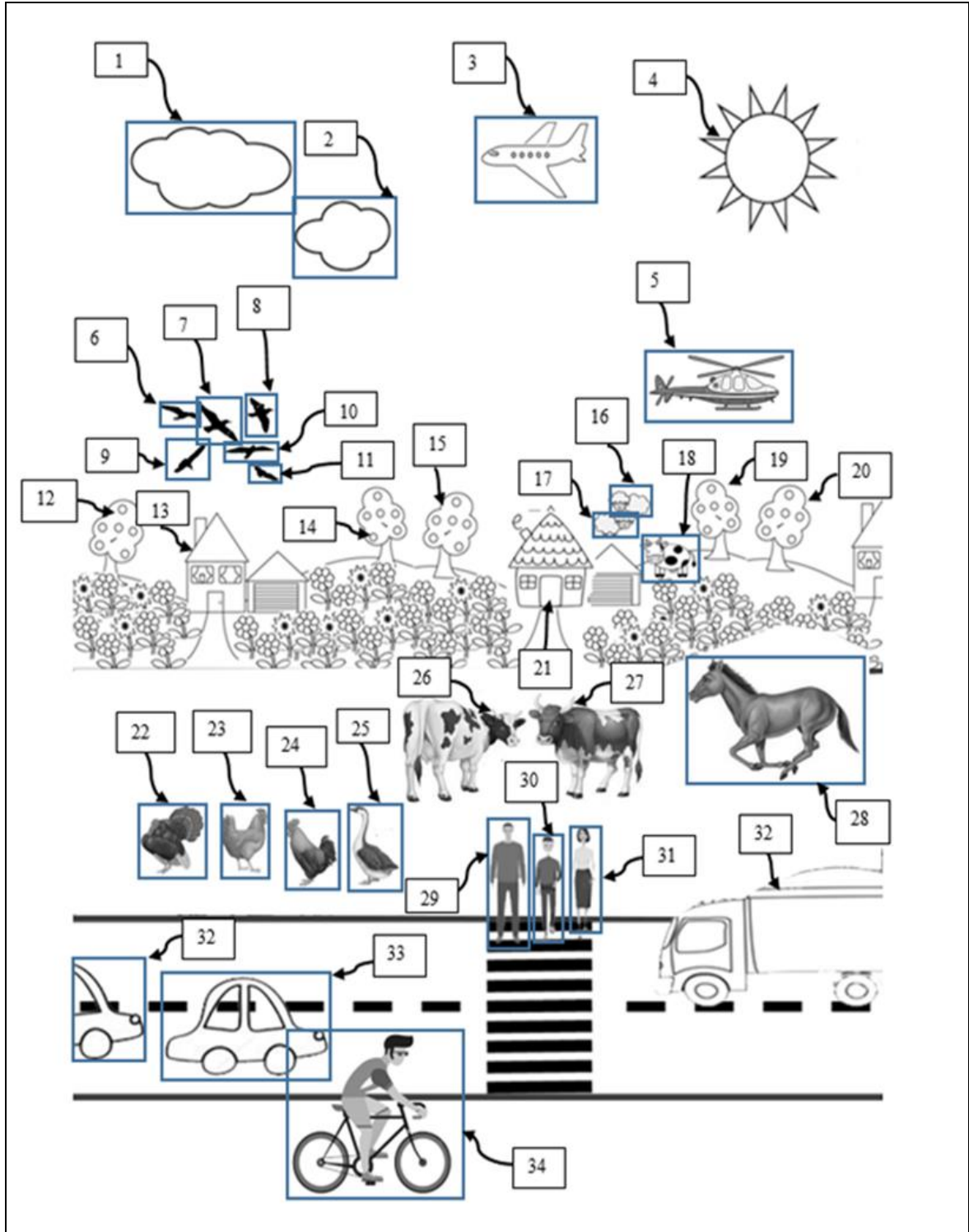
Tez çalışmasının amacı ardışık sayısal görüntülerden aynı anda birden çok hareketli nesnenin gerçek zamanlı tespiti, tanımlanması ve sınıflandırılmasını yapmaktır. Bu kapsamda geliştirilen yöntemde ardışık sayısal görüntülerdeki renkli, renksiz her türlü sayısal görüntülerden hareketli nesnelerin tespiti, tanımlanması ve sınıflandırılması (NİTİTS) yöntemi geliştirilmiştir. Tez çalışmasının amacı kapsamında geliştirilen özgün NİTİTS yönteminde bahsedilen ardışık sayısal görüntüler, teknolojinin getirdiği olanaklar sayesinde değişik çözünürlüklerde ve saniyede farklı görüntü kareleri alan sayısal görüntülerdir. Ardışık sayısal görüntüler, sayısal kameralar yardımı ile herhangi bir gerçek dünya ortamından alınan bir veya birden fazla kanaldan oluşan sayısal görüntülerdir.

Tez çalışması kapsamında geliştirilen özgün NİTİTS yönteminin blok diyagramında farklı özellik ve çözünürlüklerdeki ardışık sayısal görüntülerden hareketli nesnelerin gerçek zamanlı olarak tespiti, takibi, tanımlanması ve sınıflandırılmasının nasıl yapıldığının blok şeması Şekil 4.3’de görülmektedir.



Şekil 4.3. Geliştirilen özgün NİTİTS yönteminin blok şeması

Şekil 4.3’de görüldüğü üzere geliştirilen özgün NİTİTS yönteminde sayısal kameralardan alınan ardışık sayısal görüntülerin alındıktan sonra birinci aşamada hareket tespitinin yapılmasıdır. İkinci aşamada hareketli nesnelerin tanımlanması ve sınıflandırılmasının yapılmasıdır. Tanımlanması ve sınıflandırılmasının yapılan hareketli nesnelerin işaretlenerek kullanıcıya gösterildiği blok şeması görülmektedir.



Şekil 4.4. Ardışık sayısal görüntülerdeki hareketli nesneler

Şekil 4.4’de tez çalışması sonucunda geliştirilen NTİTS yönteminde ardışık sayısal görüntülerden hareketli nesnelerin izlenmesi ve tanımlanması işleminin ardışık sayısal görüntülerde nasıl çalıştığı anlatılmıştır. Şöyle ki ardışık sayısal görüntüler aynı ortamdan aynı sayısal kamera ile alınan art arda gelen sayısal görüntülerdir. Ardışık sayısal görüntüler

bu tez kapsamında geliştirilen NİTİTS yöntemi ile hareketli nesneleri tespit ederken ardışık sayısal görüntülerdeki önceki ve sonraki görüntü karelerini kullanarak hareketi tespit eder. Bu nedenle Şekil 4.4’de görülen görüntü karesi ardışık sayısal görüntülerde sonraki karedir. Bu ardışık sayısal görüntüler OA algoritması ile işlenerek hareket eden nesnelere ait pikseller, piksellerin ışıklılık değerlerindeki değişimleri hız vektörleri olarak hesaplayarak tespit eder. Bu hız vektörlerini ifade eden piksellerin değerleri anlamlandırılarak gruplandırılır. Hareketli nesnelere ait pikseller elde edilir. Hareketli nesnelere ait bu piksel grupları ardışık sayısal görüntülerdeki ikinci görüntü karesinden alınarak hareketli nesnenin son görüntüsü alınarak DSA ile analiz edilir ve hareketli nesnenin tanımlanması ve sınıflandırması yapılır elde edilen sonuçlar kullanıcıya işaretlenerek gösterilir.

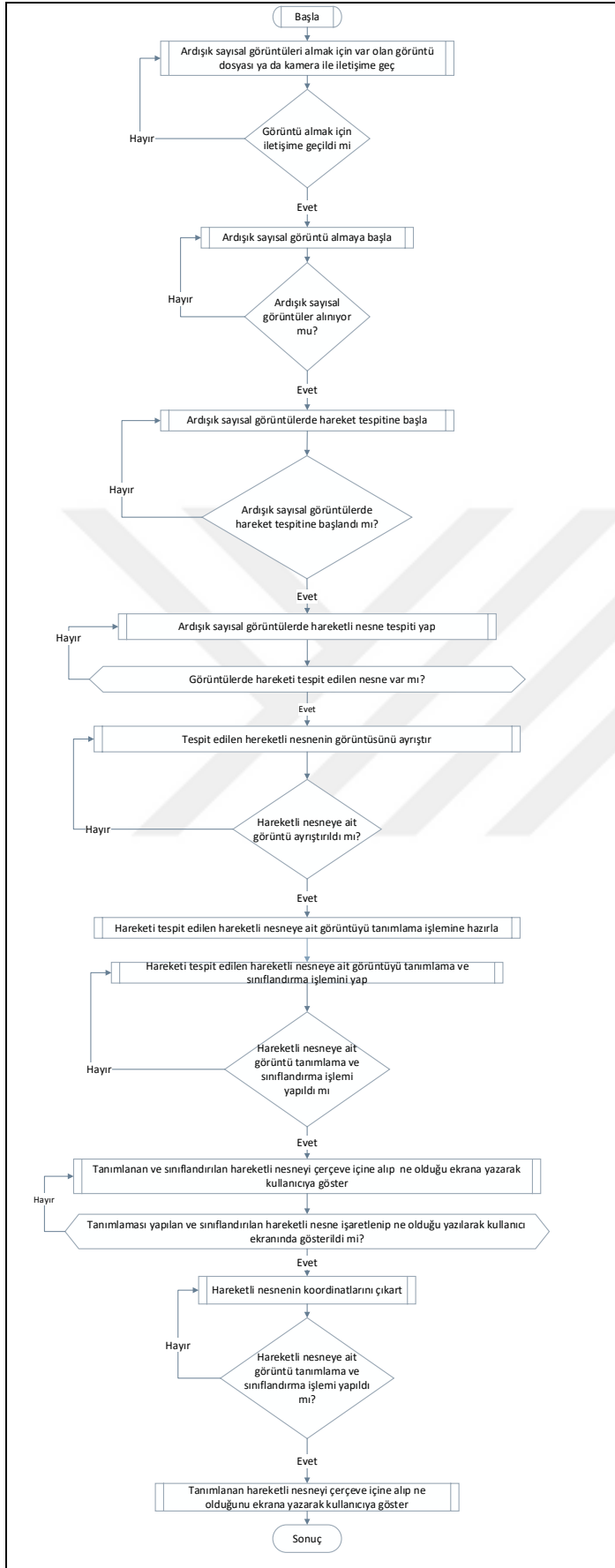
Şekil 4.4’deki görüntüde çerçeve içine alınan görüntülerdeki hareketli nesneleri çerçeve içine alarak tespit ettiği analiz sonuçlarını kullanıcıya gösterir. Hareketsiz nesneler ilgi alanı olmadığı için tanımlama ve sınıflandırılma işlemi yapılmaz. Şekil 4.4’de görülen nesnelerin hepsi numaralandırılmıştır. (1-3, 5-11, 16-18, 22-25, 28-31, 33-35) numaralı nesneler hareketli nesnelerdir. (4, 12-15, 19-21, 32) numaralı nesneler hareketsiz nesnelerdir. NTİTS yönteminde büyük, küçük nesne sınırlaması olmaksızın, ardışık sayısal görüntülerde var olan tüm hareketli nesneleri (1-3, 5-11, 16-18, 22-25, 28-31, 33-35) tespit eder. Hareketsiz olan nesnelerden (4, 12-15, 19-21, 32) ayırıştırır. Tespit ettiği hareketli nesneleri (1-3, 5-11, 16-18, 22-25, 28-31, 33-35) takibe alıp bu nesnelere ait görüntüleri, bu görüntülerin çözünürlüğünü değiştirmeden tüm özellikleri ile alıp tanımlamasını yapar ve sınıflandırmasını yapar. Hareket etmeyen nesneler tespit edilmediği için tanımlama ve sınıflandırma işlemi (4, 12-15, 19-21, 32) yapılmaz. Çünkü hareketsiz nesneler ve arka plan ilgi alanı dışındadır.

Tanımlama ve işlemlerini yaparken alınan ardışık sayısal görüntünün tüm piksellerini değil de sadece hareketli nesnelere ait olan pikselleri içeren görüntülerin tanımlama ve sınıflandırmasını yapar. Geliştirilen NTİTS yöntemi ilgi alanı olarak tespit edilen görüntülerdeki verilerde tanımlama ve sınıflandırma işlemi yaparken maksimum düzeyde giriş görüntüsünü koruduğu için nesnelerin tanımlanması ve sınıflandırılması işlemi yüksek doğruluk oranı ile gerçekleştirilir.

Çok yüksek çözünürlüklü ardışık sayısal görüntülerin işlenmesi ve faydalı bilginin çıkartılması için kullanılan yöntemlerde sayısal görüntüye ait tüm pikseller ile çalışmaları

için sistem gereksinimleri yüksek, kurulum maliyetleri ve iş yükü yüksektir. Bu sorunu aşmak için çok yüksek çözünürlüklü ardışık sayısal görüntüleri küçültme işlemine tabi tutarlar. Böylelikle görüntülerdeki pek çok bilgi kaybolur hatta bazı durumlarda küçük nesnelere ait görüntüler ya tamamen kaybolur ya da tanımlama ve sınıflandırma için görüntüye ait yeterli veri kalmaz. Bu nedenle tanımlama ve sınıflandırma işlemi yapılamaz. Bu tez çalışmasında geliştirilen NTİTS yöntemiyle, bütün görüntü yerine ilgi alanı olan hareketli nesnelere ait her zaman için ana görüntüden daha küçük görüntüler alınıp bu görüntülere de tanımlama ve sınıflandırma işlemi yapıldığı için görüntülerdeki veri kaybı en az düzeyde olur. Daha fazla sağlıklı verinin kullanılması sistemin doğruluk oranı ve güvenilirliğini artırır. Ayrıca iş yükü, sistem gereksinimleri ve kurulum maliyetleri azalır.





Şekil 4.5. NTITS yöntemini işlem akış diyagramı

Şekil 4.5’de geliştirilen NTİTS yönteminin işlem akış diyagramı görülmektedir. İşlem akış diyagramda geliştirilen yazılımın işlem adımları görülmektedir.

4.2. Tek Atış Çoklu Kutu Algılama (Single Shot Multibox Detector) (TAÇKA)

Sayısal kameralar, gerçek dünyadaki görüş alanındaki 3B’li görüntüyü 2B’li bir düzleme iz düşümünü alarak sayısal görüntüye dönüştürürler. Sayısal görüntüler bir ekranda yansıtılarak gerçek dünyanın 2B’li iz düşümleri bu ekrandan izlenir. Sayısal kameralardan alınan ardışık sayısal görüntülerde ki hareketli nesnelere ait piksellerin ışıklılık değerlerinde değişimler olur. Piksellerin ışıklılık değerlerinde değişim olmuyorsa kameranin görüş alanında bulunan nesnelerin şekillerinde bir değişiklik olmuyor ya da bir hareket olmuyordur. Ne zaman ki sahneye bir nesne girer ya da sahnede bulunan bir nesne hareket ederse sayısal kameradan alınan ardışık sayısal görüntülerde o nesnenin bulunduğu koordinatlara karşılık gelen piksellerin değerlerinde değişimler olur. Piksellerdeki ışıklılık değerlerindeki değişimler hareketlenmenin meydana gelmesinden dolayı olacağı için ardışık görüntülerde hareketi bulmak için piksel değerlerindeki değişimler hesaplanmalıdır. Hesaplanan piksel değerlerindeki değişimler tespit edildikten sonra ayrıştırılıp gruplandırılmalı ve ana görüntüden ayrıştırılarak tanımlanması ve sınıflandırılması insan gözünün ilgi alanına yoğunlaşarak çalışması prensibinden yola çıkılmıştır. Tıpkı topluluk önünde konuşma yapan konuşmacının ilgisini çeken hareketliliğin olduğu bölgeye gözünü kaydırması gibi.

Bu nedenle hızlı ve etkili bir şekilde hareketli nesnelerin tespiti tanımlanması ve sınıflandırılmasında görüş alanındaki hareketli alanların analizi ile yapmak sistem gereksinimleri ve hız bakımından daha verimli olacaktır. Bu nedenle, ardışık sayısal görüntülerde hareketli nesnelere karşılık gelen piksel gruplarının her biri ilgi alanıdır. İlgi alanlarını oluşturan bu grupların her biri analiz edilmeli ve harekete eden bu nesneler tanımlanmalıdır. Sayısal görüntüye sığabilen hareketli nesne sayısı sınırı olmamalıdır. Tanımlanma işlemi sadece hareket eden nesneye ait piksellerle yapılması tüm görüntü yerine harekete ait küçük görüntüler ile çalışılması sistem gereksinimlerini azaltırken işlem hızını artırır. Gereksiz veri yığınları ile çalışmaktan kurtarır. Buna bağlı olarak sistem gereksinimleri azalır. Kurulum maliyetini azaltır. Bütün görüntüde küçük bir alanı tanımlamaktansa sadece tanımlanacak olan alanın kesilip çıkartılması ile bu alanda olan nesnenin tanımla işleminin yapılması tanımlama hassasiyetini dolayısı ile doğruluk oranını

artırır. İşte bu özelliğinden dolayı gözetlenen bölgedeki kamera ne kadar yüksek çözünürlüklü ise önerilen yöntemin çalışma hassasiyeti de o kadar yüksek olmaktadır. İşte bu özelliğinden dolayı da henüz geliştirilmemiş çözünürlüklerdeki sayısal kamera çözünürlükleri için de geliştirilen yöntem çalışmaya hazırdır.

Yakalanan hareketli nesne çerçeve içine alınarak işaretlenir ve tespiti ve tanımlaması yapılan nesnenin ne olduğu yanına yazılarak kullanıcıya gösterilir. Hareketli nesnenin tespiti ve tanımlaması gerçek zamanlı ve sürekli olarak devam eder. Hareketin olmadığı bölgeler dikkate alınmayacağı için gerçek zamanlı hassas hareketli nesne tanımlaması yapılmaktadır. Buluş kapsamında önerilen yöntemin çalışmasında öncelikle kameradan alınan yüksek çözünürlüklü ardışık sayısal görüntülerde hareket tespiti için tarama yapılır. Görüntülerde bir hareketlenme olduğunda, hareketlenmenin olduğu bu bölgeler tespit edilerek ayrıştırılır. Ayrıştırılan bu bölgeler ana görüntüden kesilip çıkartılır, hareketli nesnelere ait küçük görüntüler elde edilir. Kesilip çıkartılan hareketli nesnelere ait bu küçük görüntüler analiz edilerek tanımlaması yapılır. Tanımlama işleminde tüm görüntü yerine harekete ait küçük görüntüler ile çalışılması sistem gereksinimlerini azaltırken işlem hızını ve tanımla hassasiyetini artırır. Tanımlaması yapılan hareketli nesne çerçeve içine alınarak ne olduğu yanına yazılır. Hareketli nesnenin tespiti ve tanımlaması sürekli olarak devam eder. Hareketin olmadığı bölgeler dikkate alınmayacağı için gereksiz bilgi yığınlarının işlenmesi ile uğraşılmamaktadır.

Sayısal görüntüyü alan yazılım sayısal görüntülerden piksellerin değerlerindeki değişimleri kullanarak hareketin olduğu bölgeleri tespit eder. Bu değişimler gerçek dünyadaki nesnelerin hareketlerine karşılık gelir. Görüntülerde hareket tespit edildikten sonra hareketin olduğu bölgeler kesilip çıkartılır. Kesilen hareketli nesnelere ait olan küçük görüntüler hareketli nesnelerin görüntüleridir. Bu görüntülerin tüm özelliklerini çıkartarak tanımlanması yapar.

Tespiti, tanımlaması ve sınıflandırılması yapılan varlık çerçeve içine alınarak işaretlenir ekranda hareketi tespit edilen tanımlaması yapılan nesnenin ne olduğunu yanına yazar kullanıcıya gösterir.

Sayısal görüntülerde belirli bir nesnenin araştırılması ve yerinin belirlenmesine nesne tespiti denir. Gerçek zamanlı nesne tespiti çalışmalardan birisi olan Viola Jones yüz tanıma

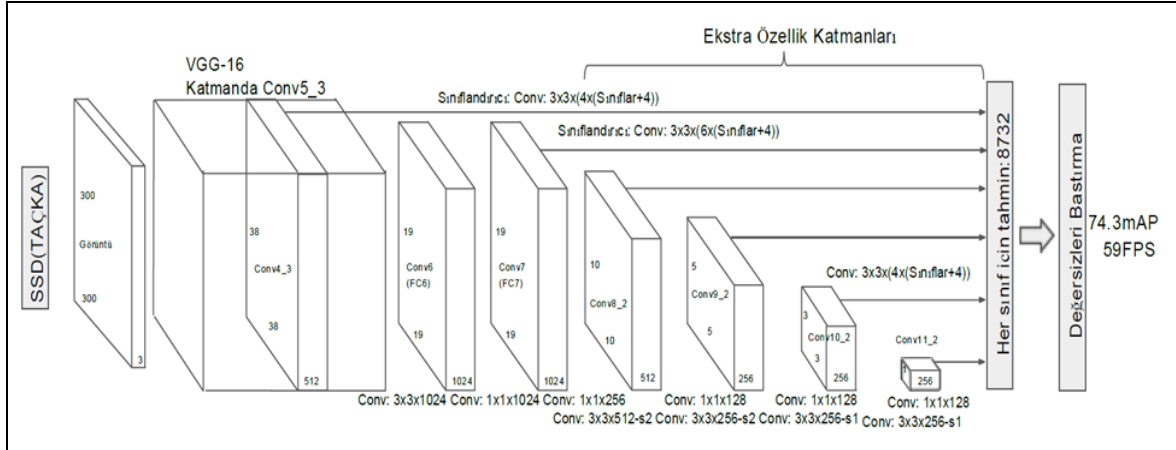
algoritmasıdır[149]. Ardışık sayısal görüntülerdeki hareketli nesnelerin tanımlanması insanın algısına yakın bir doğrulukta algılanıp tespit edilmesi makina öğrenmesinin konusudur.

DSA büyük verileri tek bir sinir ağı katmanı yerine pek çok katmanda, hesaplamaları tek bir seferde yapabilen hatta tanımlanması gereken parametreleri bile kendisi keşfedebilen, belki de daha iyi parametreler ile değerlendirmelerde bulunabilen bir sistemdir. Bu nedenle tez çalışmasında nesnelerin tanımlanmasında ve sınıflandırılması işlemlerinde DSA'ları kullanılarak yapılmıştır. DSA'lardan ESA sınıfında olan Tek Atış Çoklu Kutu Algılama (Single Shot Multibox Detector) (TAÇKA) kullanılmıştır. TAÇKA Liu ve ark. tarafından gerçek zamanlı nesne algılama tekniği olarak Kasım 2016 yılında geliştirilmiştir[122]. TAÇKA, arka plan bilgisini kullanan, tek seferde nesne tespiti yapabilen evrişim katmanlarına sahip ileri beslemeli bir ağıdır. Nesne tespiti ve sınıflandırmasında tüm işlemleri tek ileri yönlü hesaplamada yaptığı için “Singe Shot (Tek Atış)” denilmiştir. TAÇKA mimarisi temel ağ olarak VGG ağını kullanmıştır. VGG-16 ağının temel ağ olarak kullanılmasının nedeni, yüksek kaliteli görüntü sınıflandırma görevlerindeki güçlü performansı ve aktarımla öğrenmenin sonuçları iyileştirmeye yardımcı olduğu içindir. Orijinal VGG tamamen bağlı katmanlar yerine, bir dizi yardımcı evrişimli katmanları (conv6'dan itibaren) eklendi, böylece sayısal görüntüye ait özelliklerin birden çok ölçekte çıkarılmasına ve sonraki her katmana girişin boyutunun kademeli olarak azaltılmasına olanak sağlandı.

“TAÇKA, tek bir derin sinir ağı kullanarak görüntülerdeki nesneleri tespit etmek için sınırlayıcı kutular kullanır ve bu sınırlayıcı kutuların çıktı alanını, özellik haritası konumu başına çeşitli en boy oranları ve farklı ölçekler üzerinden bir dizi varsayılan kutuya ayırır”[122]. “YSA tahmin zamanında, ayrılan her varsayılan kutuda bulunan nesnelerin her bir kategorisinin varlığı için puanlar üretir ve kutunun nesnenin şekline en uygunu için ayarlamalar yapar. Ek olarak, ağ, çeşitli boyutlardaki nesneleri doğal olarak işlemek için farklı çözünürlüklere sahip birden çok özellik haritasından gelen tahminleri birleştirir”[122]. “TAÇKA modeli, teklif oluşturmaya ve sonraki piksel veya özellik yeniden örnekleme aşamasını tamamen ortadan kaldırdığı ve tüm hesaplamaları tek bir ağda yaptığı için nesne önerileri gerektiren yöntemlere göre basittir. Bu, TAÇKA'nın eğitilmesini ve algılama bileşeni gerektiren sistemlere entegre edilmesini kolaylaştırır. PASCAL VOC, MS COCO ve ILSVRC veri kümelerindeki deneysel sonuçlar, TAÇKA'nın hem eğitim hem de çıkarım

için birleşik bir yapı sağlarken, ek bir nesne önerme adımı kullanan yöntemlerle karşılaştırılabilir doğruluğa sahip olduğunu ve çok daha hızlı olduğunu doğrulamaktadır. Diğer tek aşamalı yöntemlerle karşılaştırıldığında SSD, daha küçük girdi görüntü boyutlarında bile çok daha iyi doğruluğa sahiptir. TAÇKA, 300×300 giriş görüntüsü için bir Nvidia Titan X ekran kartı ile 58 FPS'de VOC2007 testinde % 72,1 mAP elde eder ve 500×500 giriş için TAÇKA, % 75,1 mAP elde eder, Faster R-CNN modeli ile karşılaştırıldığında daha iyi performans göstermiştir[122]". mAP (mean Average Precision) doğruluğun ortalaması demektir.

Tüm ESA'larda ağa verilen girdi görüntüsü yeniden boyutlandırılarak standart girdi görüntüsü boyutu elde edilerek ağa verilmektedir. Bunun nedeni sayısal görüntünün evrişim işlemlerinden sonra her bir pikselinin ağda eğitilmesidir. Bu nedenle ağa verilecek olan girdi görüntüsü yeniden boyutlandırılarak standart bir girdi görüntüsü oluşturulur. Bu çalışmada da girdi görüntüsü olarak görüntüler yeniden boyutlandırılarak 300×300 'lük girdi görüntüsü kullanılmıştır. Bu girdi görüntüsü tüm ESA'larda ilk katman olan evrişim katmanına verilir. Bu katmanda birden fazla filtre kullanılarak girdi görüntüsü evrişim işlemine sokulur. Bu şekilde ESA'larda giriş görüntüsünden farklı öznelik haritaları(feature maps) çıkartılır.



Şekil 4.6. TAÇKA mimarisini oluşturan yapılar

Yukarıdaki Şekil 4.6'da VGG ağını kullanan TAÇKA mimarisi gösterilmektedir. İlk evrişim katmanı 38×38 boyutlarında ve 512 derinliğe sahiptir. 512 adet 38×38 boyutlarındaki görüntü evrişim filtreleri ile evriştirilerek nesnelere ait öznelikleri çıkarır. 512 farklı öznelik haritası özellikleri kullanılarak, çok noktadaki küçük nesneler için sınırlayıcı kutuyu tahmin etmek ve görüntü sınıflandırma için kullanılır. İkinci evrişim katmanı 19×19

boyutlarında ve 1024 derinliğe sahiptir. 1024 adet 19x19 boyutlarındaki evrişmiş nesnelere ait öznitelik haritaları vardır. 19x19 boyutlarındaki filtreler ilk katmandaki 38x38 boyutlarındaki filtrelere göre daha büyük nesnelere ait öznitelikleri çıkarmada daha iyidir. Evrişim katmanları 10x10, 5x5, 3x3 şeklinde bir sonraki evrişim katmanında giderek azalır en nihayetinde 1x1 boyutundaki evrişime kadar küçülür. Son evrişim katmanı olan 1x1 lik filtre çok büyük nesneler için uygundur ve bu nesneleri tekbir değer ile ifade eder. Her evrişim katmanının çıkışında nesne sınırlayıcı kutu bilgisi gönderilir.

Conv4_3'de boyutunda olduğundan ve 4 sınırlayıcı kutu olduğundan her sınırlayıcı kutu (sınıflar + 4) çıkışlara sahip olacaktır. Böylece, Conv4_3'de çıktı olur. 20 nesne sınıfı ve birde arka plan sınıfı olduğundan, çıktı $38 \times 38 \times 4 \times (21+4) = 144\,400$ olur. Sınırlayıcı kutu sayısı ise $38 \times 38 \times 4 = 5\,776$ olur.

Benzer şekilde tüm evrişim katmanları için aynı hesaplama yapılırsa

Conv7 için $19 \times 19 \times 6 = 2\,166$ kutu (her konum için 6 kutu)

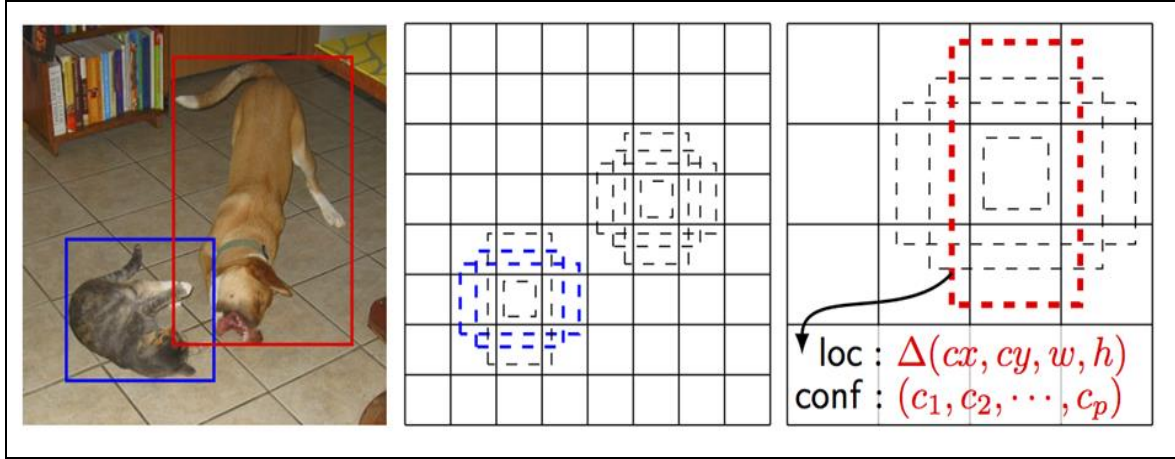
Conv8_2 için $10 \times 10 \times 6 = 600$ kutu (her konum için 6 kutu)

Conv9_2 için $5 \times 5 \times 6 = 150$ kutu (her konum için 6 kutu)

Conv10_2 için $3 \times 3 \times 4 = 36$ kutu (her konum için 4 kutu)

Conv11_2 için $1 \times 1 \times 4 = 4$ kutu (her konum için 4 kutu)

Tüm evrişim katmanlarından elde edilen sınırlayıcı kutuların sayıları toplandığında $5\,776 + 2\,166 + 150 + 36 + 4 = 8\,132$ kutu elde edilir. Hangi kutunun uygun sınırlayıcı kutu olacağı ise kesin referans kutusu ile bağlantı kutusu arasında en yüksek birlik üzerinde kesişime sahipse, birlik üzerinde kesişime sahip olan o kutu sınırlayıcı kutu olarak seçilir ve tespit edilen nesne bu kutu içinde gösterilir.



Şekil 4.7. a) nesne tespit kutuları ile giriş görüntüsü b) özellik kutu haritası c) özellik kutu haritası[122]

“TAÇKA’nın hareketli objeleri değişken ölçeklerde tespiti için çoklu ölçekteki öznitelik haritalarını ve varsayılan objelere ait alanları kapsayan kutuların ne şekilde seçildiği ve bu kutuların en uygun kutuya indirgenmesine ait örnek Yukarıdaki Şekil 4.7’de, gösterilmiştir. Yukarıdaki Şekil 4.7 a)’da köpek, 4x4 özellik haritası katmanında varsayılan kutu (kırmızı) ile eşleşmiştir. Ancak daha yüksek çözünürlüklü 8x8 özellik haritasındaki herhangi bir varsayılan kutu ile eşleşmez. Köpekten daha küçük olan kedi ise yalnızca 2 varsayılan kutuda (mavi) 8x8 özellikli harita katmanı tarafından algılanmıştır çünkü TAÇKA, küçük nesneleri daha yüksek çözünürlüklü katmanlarda Şekil 4.7’de (en soldaki katmanlar yani ilk evrişim katmanları) tespit edilebilir. Ancak bu katmanlar, sınıflandırma için daha az bilgilendirici olan kenarlar veya renk yamaları gibi düşük seviyeli özellikler içerir”[122]. Her konum için, 4 sınırlayıcı kutumuz var. Bu 4 sınırlayıcı kutu, farklı boyutlarda en, boy oranlarına sahiptir. Kutular yatay ve dikedir. Sınırlayıcı kutuların her biri için sınıf puanı nesnenin gerçek boyutlarını bulmak için sınırlayıcı kutu boyutları hesaplanır.

TAÇKA birden çok nesne kategorisini ele alacak şekilde genişletilmiştir. TAÇKA nesnenin gerçek sınır kutusu ve tahmin sınır kutusu arasındaki yerleştirme kaybını denklemler (4.1), (4.2) ve (4.3) ile hesaplanır.

Burada $x_{ij}^p = \{1, 0\}$, i -th varsayılan kutusunu, p kategorisindeki j -th zemin doğruluğu kutusuyla eşleştirmenin bir göstergesi olsun. $\sum_i x_{ij}^p \geq 1$ genel amaç kaybı fonksiyonu, yerleştirme kaybı (loc) ve güven kaybı fonksiyonlarının (conf) ağırlıklı toplamıdır.

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \quad (4.1)$$

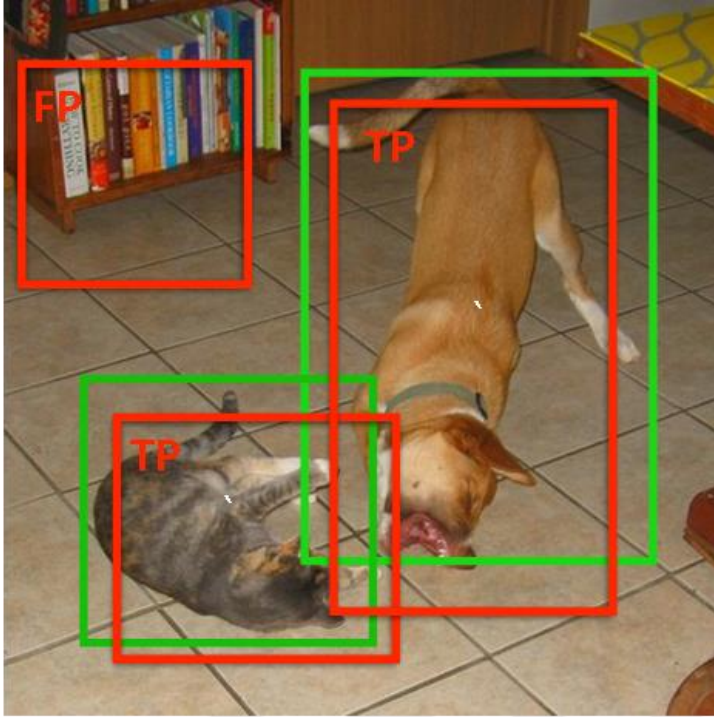
N , eşleşen varsayılan kutuların sayısıdır. $N = 0$ ise, kaybı 0 olarak alınır.

Tahmin edilen kutu l ile kesin doğruluk kutusu g arasındaki yerelleştirme kaybı yumuşatılmış $L1$, ile cx , cy ise varsayılan sınırlayıcı kutu d 'nin w genişliği ve h yüksekliği olarak tanımlanır.

$$\begin{aligned} L_{loc}(x, l, g) &= \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k smooth_{L1} \left(l_i^m - \hat{g}_j^m \right) \\ \hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h \\ \hat{g}_j^w &= \log \left(\frac{g_j^w}{d_i^w} \right) \quad \hat{g}_j^h = \log \left(\frac{g_j^h}{d_i^h} \right) \end{aligned} \quad (4.2)$$

$$L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log \left(\hat{c}_i^p \right) - \sum_{i \in Neg} \log \left(\hat{c}_i^0 \right) \rightarrow \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (4.3)$$

Burada N , pozitif eşleşen varsayılan kutuların sayısıdır, α yerleştirme kaybı ağırlığıdır. Her öznitelik haritası için çoklu kutularla, orijinal yer gerçeği kutularının dağılımıyla yakından eşleşen önceden hesaplanmış, sabit boyutlu sınırlayıcı kutular birlik üzerinde kesişime (Jaccard benzerlik indeksi) karşılık gelir ve bu değerlerin 0,5'ten büyük olanları kabul edilir. Yer gerçeğini tahmin etmekten sorumlu olan bağlantı kutusunun, yer gerçeği kutusunun ortasına yerleştirildiğinde maksimum birlik üzerinde kesişimi veren kutu olarak seçilir. Seçimde yer gerçeği kutuları ile nesnenin boyutlarının uyumu dikkate alınır. Resim 4.2'de yeşil renkli çerçeve içine alınan kedi ve köpekler görüntüdeki gerçek konumlarındadır. İçlerinde kedi ve köpek olan kırmızı çerçeveler ise yazılımın tespit ettiği nesneleri yani kedi ve köpeği ifade eder. Kırmızı ve yeşil çerçevelerin alanlarının birbirlerine benzerliği yani birlik üzerinde kesişimler 0,5'ten büyük olduğu için kesin referans kutuları olarak alınır. Sol tarafta kırmızı çerçeve ise yanlış olarak kabul edilir.



Resim 4.2. Varsayılan kutuların, kesin referans kutuları ile eşleştirilmesi

Her özellik haritası için varsayılan m kutuların ölçeği denklem (4.4)'deki gibi hesaplanır:

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1} (k-1), \quad k \in [1, m] \quad (4.4)$$

$s_{\min}$ değeri en alt katman için 0,2'dir. $s_{\max}$ değeri en üst katman için 0,9'dur ve diğer katmanlar düzenli aralıklarla yerleştirilmiştir. $a_r \in \left\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\right\}$ iken her bir varsayılan kutu için farklı genişlik denklem (4.5)'deki gibi ve yükseklik denklem (4.6)'daki gibi hesaplanır.

$$w_k^a = s_k \sqrt{a_r} \quad (4.5)$$

$$h_k^a = s_k / \sqrt{a_r} \quad (4.6)$$

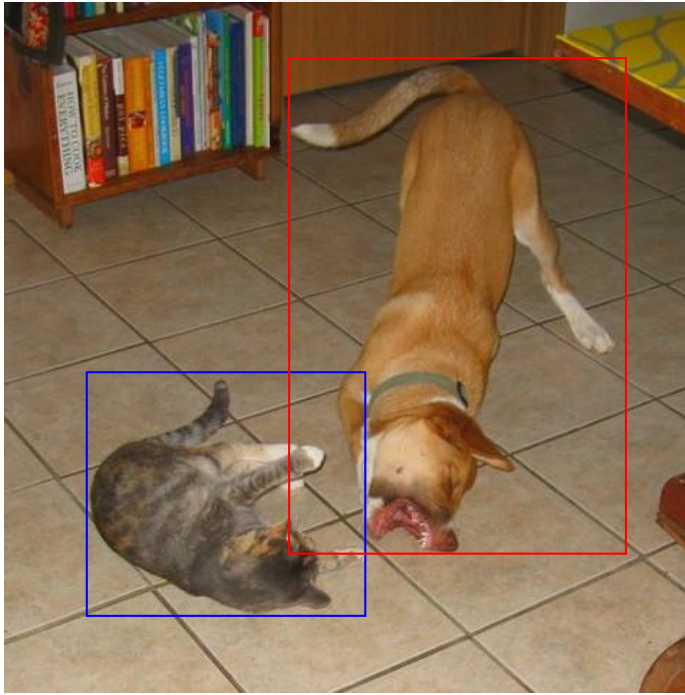
En boy oranı 1 için ölçeği (4.7)'deki gibi varsayılan bir kutu eklenir, özellik haritası konumu başına 6 varsayılan kutu ile sonuçlanır.

$$s_k' = \sqrt{s_k s_{k+1}} \quad (4.7)$$

Her varsayılan kutunun merkezi denklem (4.8)'deki gibidir. $i, j \in [0, |f_k|] \mid |f_k|, k'nıncı$ özellik haritasının boyutudur.

$$\left(\frac{i+0,5}{|f_k|}, \frac{j+0,5}{|f_k|} \right) \quad (4.8)$$

Pek çok özellik haritasının tüm konumlarından farklı ölçeklere ve en boy oranlarına sahip tüm varsayılan kutular için tahminleri birleştirerek, çeşitli girdi nesnesi boyutlarını ve şekillerini kapsayan çok çeşitli tahminler var. Örneğin, Resim 4.2'de köpek, 4x4 özellik haritasındaki varsayılan bir kutu ile eşleştirilir, ancak 8x8 özellik haritasındaki varsayılan kutulardan herhangi bir ile eşleşmez. Bunun nedeni, bu kutuların farklı ölçeklere sahip olması ve köpek kutusuyla eşleşmemesi ve bu nedenle eğitim sırasında negatif olarak değerlendirilmesidir.

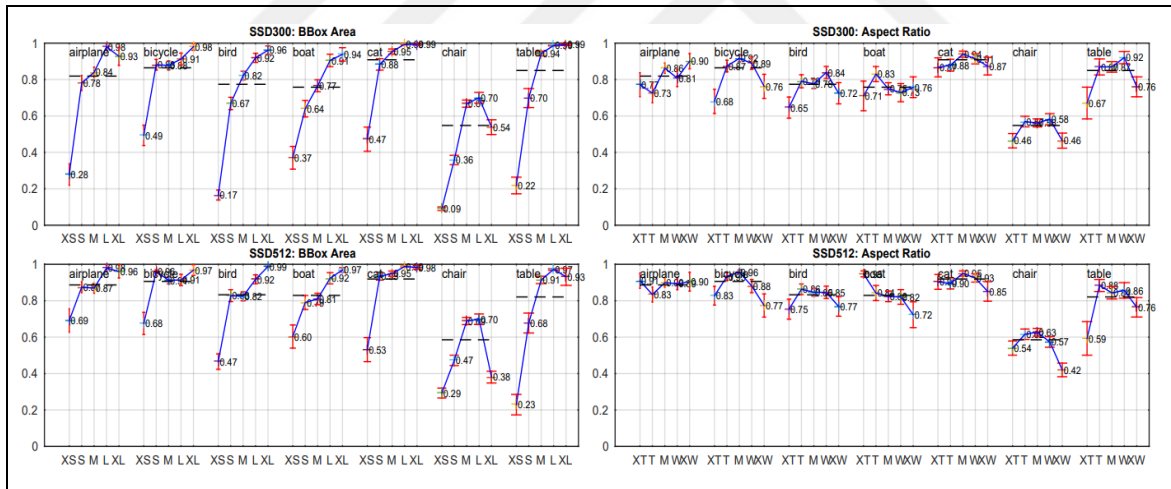


Resim 4.3. Seçilen sınırlayıcı kutunun nesneler üzerinde gösterimi

Varsayılan kutuların seçiminden elde edilen kesin referans kutularının nesneler üzerinde ki gösterimi Resim 4.3'de görülmektedir. Resim 4.3'de de görüldüğü üzere seçilen sınırlayıcı kutuların şekillerinin nesnelerin gerçek boyutlarına çok yakın olduğu görülür.

Çok fazla varsayılan kutu kullanılması hız üzerinde olumsuz bir etkisi olmasına rağmen, detektörün birden çok çözünürlükte çalışması daha iyi algılama ile sonuçlanır. İşlem zamanının% 80'i VGG-16 ağında harcanmaktadır: Bu, daha hızlı ve eşit derecede doğru bir ağ performansının olabileceği anlamına gelir. TAÇKA, nesneleri benzer kategorilere (örneğin hayvanlar) karıştırır. Bunun nedeni konumların birden fazla sınıf için paylaşılmasıdır[150].

TAÇKA-500 ağı 512x512 giriş görüntülerini kullanır. Pascal VOC2007'de% 76,8 ile mAP elde eder, ancak saniyedeki görüntü işleme hızı 22 fps'ye düşer. TAÇKA-300 ağı 300x300 giriş görüntüsü kullanır ve %74,3 mAP elde eder[150]. Pascal Görsel Nesne Sınıfları (Pascal VOC) nesne sınıflandırma için kullanılan herkese açık veri kümesidir. Aşağıdaki Şekil 4.11'deki VOC2007 test seti kullanılarak TAÇKA'dan elde edilen farklı büyüklüklerdeki nesneleri sınırlayıcı kutular ile nesnelerin en boy oranları sonuçları görülmektedir. Şekil 4.8 incelendiğinde küçük nesneleri bulmadaki başarısı görülecektir.



Şekil 4.8. TAÇKA'nın farklı büyüklüklerdeki nesnelerin VOC2007 test seti üzerindeki elde edilen sınırlayıcı kutuların ile nesnelerin en boy oranları sonuçları

Yukarıdaki Şekil 4.8'de TAÇKA yönteminin küçük nesnelerin ve farklı büyüklüklere sahip nesnelerin VOC2007 veri seti kullanılarak elde edilen sınırlayıcı kutuların, nesnelerin en boy oranlarını tahmininde elde edilen sonuçları gösterilmiştir.

İki atışlı algılama yöntemlerinin bölge önerisi ve ardından bu bölgelerin sınıflandırılması ve konum tahmininin iyileştirilmesi olmak üzere iki aşaması vardır. Tek atışta tespit, bölge teklif aşamasını atlar ve aynı anda nihai yerelleştirme ve sınıflandırma tahminini yapar. Çizelge 4.1'de Pascal Voc 2007 veri seti kullanılarak iki atışlı Daha hızlı R-CNN ve tek

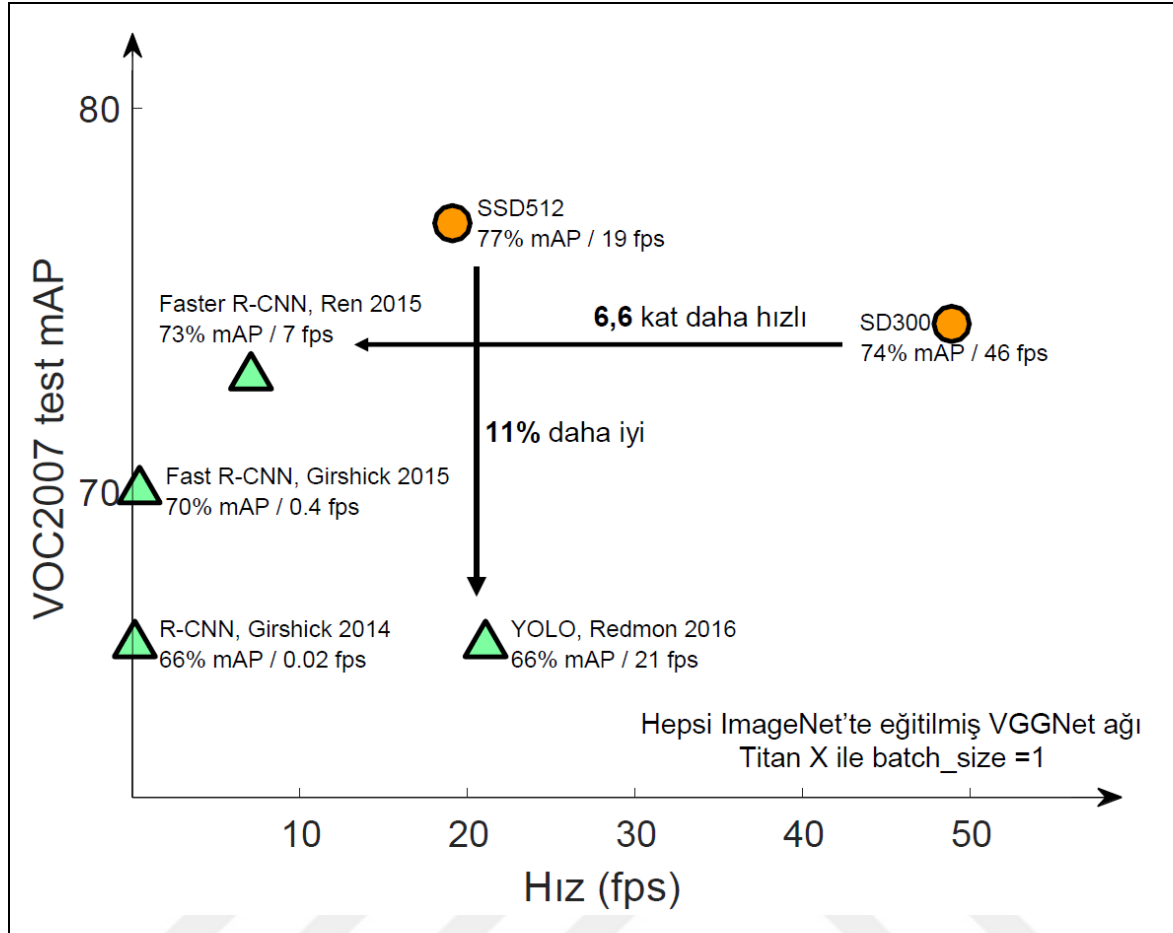
atışlı Hızlı YOLO, YOLO (VGG16) mimarilerine ait nesne tanıma sonuçları görülmektedir. Tablodaki sonuçlara göre TAÇKA gerçek zamanlı görüntü işleme için Daha hızlı R-CNN daha hızlı ve doğruluk oranı yüksektir. Hızlı YOLO mimarisi TAÇKA'dan hızlı olmasına rağmen doğruluk oranları olarak TAÇKA'dan daha az ortalama doğruluk oranı ile çalışır.

Çizelge 4.1. Pascal Voc 2007 veri seti kullanılarak elde edilen sonuçlar

Metot	mAP	fps	Batch size	Sınırlayıcı kutular	Görüntü çözünürlüğü
Daha hızlı R-CNN	73,2	7	1	~6000	~1000x600
Hızlı YOLO	52,7	155	1	98	448x448
YOLO (VGG16)	66,4	21	1	98	444x444
SSD300	74,3	46	1	8732	300x300
SSD512	76,8	19	1	24564	512x512
SSD300	74,3	59	8	8732	300x300
SSD512	76,8	22	8	24564	512x512

Yukarıdaki Çizelge 4.1 ve aşağıdaki Şekil 4.9'da TAÇKA yönteminin VOC2007 veri seti üzerinde, NVIDIA TitanX GPU kullanarak hız ve doğruluk oranları bakımından TAÇKA 300'ün 46 fps ile Daha hızlı RCNN, 7 fps'den 6,6 kat daha hızlı çalıştığı ve Daha hızlı RCNN'in %73 mAP ile çalışırken TAÇKA 300'un %74 mAP olarak çalışmaktadır. TAÇKA 512'nin 19 fps ile %77 mAP ile çalışırken Yolo, Remdon 2016 21 fps ile çalışırken %66 mAP ile çalışmaktadır. TAÇKA 512, Yolo, Remdon 2016'dan %11 mAP daha fazla güvenilirlik oranı ile çalışmaktadır. VGG-16 ağı üzerine kurulu TAÇKA ağıının hız performansı ve doğruluk oranı olarak oldukça başarılı olduğu görülmektedir.

TAÇKA daha hızlı eğitilir ve iki atışlı mimarilere göre daha az işlemle hızlı çıkarıma sahiptir. Tek atış da daha az hesaplama ile işlemleri yaptığı için sistem gereksinimleri ve tahmin başına enerji tüketimi daha azdır. Fosil yakıtlara daha az gereksinim duyduğu için çevre dostudur. Zaman ve enerji verimliliği, özellikle sistem gereksinimleri söz konusu olduğunda daha hızlı eğitim ve daha hızlı çıkarım özelliği, gerçek zamanlı uygulamalar için bir gereklilik olmaktadır. Bu da TAÇKA'yı gerçek zamanlı kullanım için tercih edilen nesne algılama yaklaşımı olarak öne çıkartır.



Şekil 4.9. TAÇKA'nın VOC2007 veri seti üzerinde hız ve doğruluk oranları bakımından karşılaştırılması görülmektedir

Yukarıdaki Şekil 4.9'da VOC2007 veri seti üzerinde TAÇKA300 ve TAÇKA500 ağlarının Daha hızlı R-CNN, Hızlı R-CNN, R-CNN, YOLO ağları ile çalışmasından elde edilen ortalama doğruluk oranları ve hızları gösterilmiştir. TAÇKA300 ağı, Daha hızlı R-CNN ağından 6,6 kat daha hızlı ve doğruluk oranı da daha fazla olarak ölçülmüştür.

4.3. NİTİS Yönteminin Yazılımının Geliştirilmesi

Bu tez çalışmasının amacı kapsamında yapılan çalışmalar sonucunda geliştirilen NTİS yönteminin yazılımı, Linux tabanlı işletim sistemi olarak Ubuntu 16.04 işletim sistemi kullanılmıştır. Yazılım programı olarak Python programlama dili, görüntü işleme uygulamaları için özel olarak geliştirilmiş OpenCV kütüphanesi, matematiksel hesaplamalarda için geliştirilmiş olan Numpy kütüphanesi, grafiksel ifadelerin çizilmesinde ve gösterilmesinde kullanılan Matplotlib kütüphanesi gibi birbirinden bağımsız ve farklı bileşenlerin bir araya getirildiği ve çalışmasının sağlandığı Linux platformu üzerinde

birleştirilmiş ve kullanılmıştır. Bu bölümde kullanılan bu yazılım paketleri hakkında bilgilendirme yapılacaktır.

4.3.1. Ubuntu 16.04 işletim sistemi

Geliştirilen NTİTS yönteminde işletim sistemi olarak Linux tabanlı tamamen ücretsiz sınırsız yazılım geliştirme yeteneği olarak bilinen bir işletim sistemi olan Ubuntu 16.04 işletim sistemi kullanılmıştır. NTİTS yöntemi ile ilgili yapıla tüm çalışmalar Ubuntu 16.04 işletim sistemi ortamında yapılmıştır. Ubuntu, Linux tabanlı tamamen açık kaynak kodlu ve tamamen ücretsiz erişimi kolay bir işletim sistemidir ve değişime açık fonksiyonel kullanımlar için oldukça uygundur. Linux, her şeyden önemlisi herhangi bir işletim sistemi ile entegre edilerek geliştirilecek olan yazılımın temel bileşeni olan herhangi bir donanım ile etkin bir şekilde haberleşmesine izin veren bir çekirdeğe sahiptir. Sahip olduğu bu çekirdek, kendisi işletim sisteminden ziyade yazılımlar ile donanımların arasında bilgi aktarımı ve bu bilgilerin işlenmesi sırasında haberleşmeyi sağlayan çok yönlü kullanıma uygun olan bir dizi bilgisayar talimatından oluşmaktadır[151].

4.3.2. Kullanılan programlama dili Python

Bu tez çalışmasında, yazılım dili olarak python programlama dili kullanılmıştır. Python programlama dili, Ubuntu 16.04 işletim sisteminde python2.7 ve python3 versiyonları ile ön yüklemeli olarak hazır halde vardır. Python dili, Hollandalı programcı Guido Van Rossum tarafından 1991 yılında geliştirilmeye başlanmıştır. Adını Monty Python's Flying Circus adlı komedi dizisinden almıştır. Nesne yönelimli (object oriented), yorumlanabilen (interpreted), modüler ve etkileşimli (interactive) bir programlama dilidir. Python dili, Unix, Linux, Windows, Mac OS, Amiga OS, Symbian gibi işletim sistemleri ile çalışmaktadır[152]. Python dili, Python Yazılım Vakfı tarafından desteklenmektedir ve gelişimi sürdürmektedir. Açık kaynak kodludur ve 2000 yılından buyana Python Lisansı (GPL ile uyumlu bir lisans) altında dağıtılmaktadır[153]. Açık kaynak kodlu olması kullanıcıya kolaylık sağlamaktadır[152, 153].

4.3.3. OpenCV kütüphanesi

OpenCV (Open Source Computer Vision), 1999 yılında Intel tarafından, Intel'in Rusya'daki laboratuvarlarında geliştirilmeye başlanmıştır[154]. Günümüzde BSD lisansı ile dağıtılan açık kaynak kodlu, platform bağımsız bir bilgisayarla görme ve makine öğrenme kütüphanesidir[155, 156]. OpenCV, kelimesi açılımında Open olan ilk kelime aynen alınarak Computer Vision kelimelerinin baş harflerinin birleştirilmesi ile elde edilmiştir. OpenCV, bilgisayarla görme uygulamalarında her kesimin üzerinde ortak bir yapıda buluşmasını sağlamak ve ticarileşen ürünlerde makinaların kullanılmasını yaygınlaştırmak için inşa edilmiştir[156]. C ve C++ programla dilleri kullanılarak geliştirilmiştir. OpenCV, esas olarak bilgisayarlı görü uygulamalarına yönelik olarak geliştirilmiş programlamalarda yaygın olarak kullanılan fonksiyonları içeren bir kütüphanedir. OpenCV kütüphanesi ilk başta Intel firması tarafından geliştirildi, daha sonra Willow Garage ve ardından Itseez tarafından desteklenmiştir[155, 156]. Opencv kütüphanesi Windows, Linux, Android, MacOS gibi işletim sistemleri ile çalışabilen platform bağımsız bir kütüphanedir. OpenCV kütüphanesi C,C++, Matlab, Python, Java, C# vb. gibi pek çok programlama dili ile kullanılmaktadır[155, 156]. Programlama dili desteği olmayan dillerde ise açık kaynak kodlu olması nedeni ile bu diller için kullanılan katmanlar (wrapper) ile bu dillere entegre edilerek kullanılabilir[156]. OpenCv kütüphanesinde 2500'den fazla görüntü işleme algoritması bulunmaktadır[156]. 2010 yılında OpenCV'ye GPU hızlandırma sağlayan yeni bir modül eklendi[155]. GPU modülü, kütüphanenin işlevselliği için önemlidir aktif olarak gelişimi devam etmektedir[155].

4.3.4. NumPY kütüphanesi

NumPY kütüphanesinin ismi Numerical Python kelimelerinin kısaltılmasıyla elde edilmiştir. NumPy kütüphanesi bilimsel hesaplamaları kolaylaştırmak ve hızlandırmak amacı ile python dili için yazılmış bir matematik kütüphanesidir. Numpy'ın temelini numpy dizileri oluşturur[157]. Makine öğrenmesi, görüntü işleme ve YZ uygulamalarındaki hesaplama işlemleri ve dönüşüm işlemlerinde kullanılan kod yapısı ile numpy kütüphanesi daha az kod ve daha sade yazılması için tasarlanmıştır. Matematiksel işlemleri çok hızlı yapmasından dolayı programcılar tarafından yoğun olarak kullanılır.

4.3.5. Matplotlib kütüphanesi

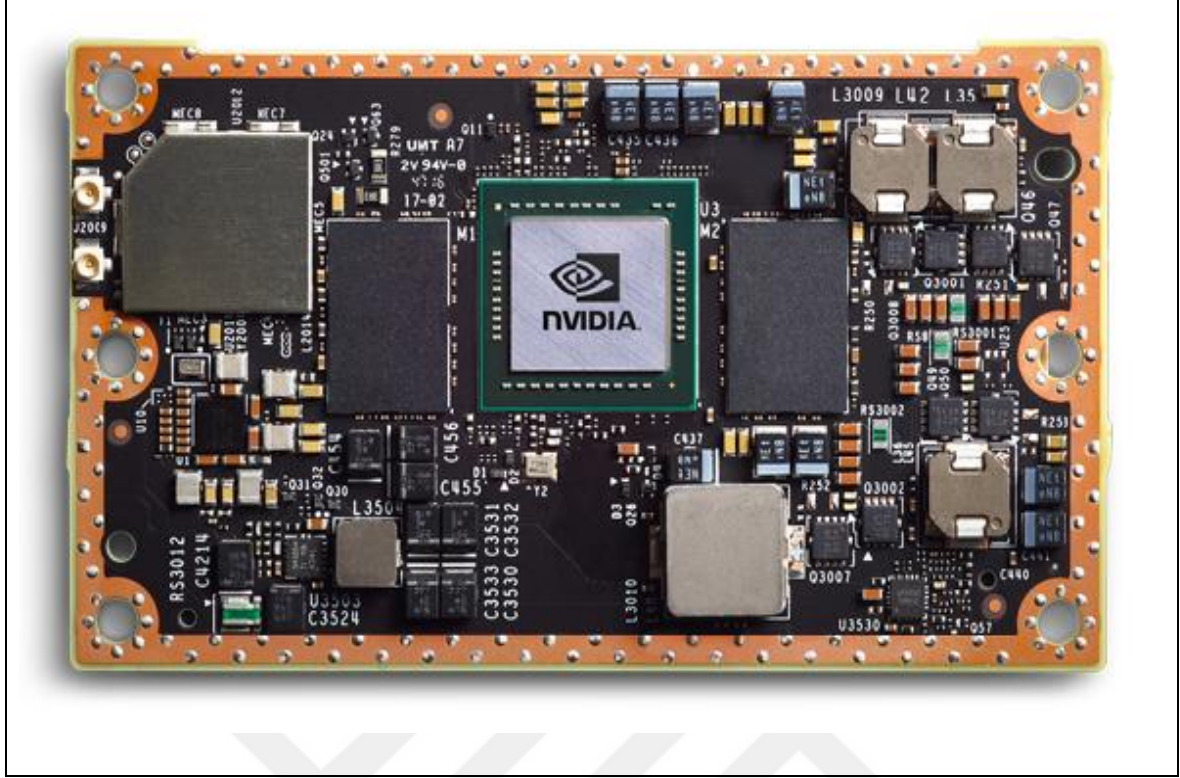
Bilimsel arařtırmalarda matematik ile ilgili işlemler için geliştirilmiştir çok geniş bir kullanım alanına sahiptir ve tüm grafik çizimleri yapabilen bir kütüphanedir. Matematic Plot Library kelimelerinin kısaltılması ile elde edilmiştir. Matematiksel ifadelerden elde edilen verilerin görselleştirilmesi için kullanılır. Python programlama dili ve Numpy için 2B ve 3B[157] grafikler için kullanılan çizim kütüphanesidir. İşlemleri oldukça kolaylaştırır.

4.4. Geliştirilen NTİTS Yönteminin Çalıştırılmasında Kullanılan Donanım

Bu tez çalışmasının amacı kapsamında yapılan çalışmalarda ardışık sayısal görüntülerden, gerçek zamanlı olarak hareketli nesnelerin tespiti izlenmesi sınıflandırılması yapılmıştır. Tez çalışmasını amacı kapsamında yapılan çalışmalar sonucunda NTİTS yöntemi geliştirilmiştir. Geliştirilen NTİTS yöntemi iki aşamalıdır. Birinci aşamada OA algoritması kullanılmıştır. OA algoritması ile ardışık sayısal görüntülerden hareketli nesnelere ait görüntüler elde edilmiştir. İkinci aşamada, birinci aşamada elde edilen hareketli nesnelere ait görüntüler YZ'nın bir alt dalı olan DSA ile analiz edilmiştir. DSA ile hareketli nesnelere ait sayısal görüntüler analiz edilerek hareketli nesnelerin tanımlama ve sınıflandırma yapabilmesi için görüntüye ait tüm özellikler çıkartır. DÖ ile daha önceden öğrenilmiş veriler sayesinde DSA algoritmaları paralel çalışabilen işlemciler GPU'lar sayesinde sayısal görüntü karelerini aynı anda yüzlerce, binlerce hatta on binlerce filtreden geçirip filtreleme işleminin sonucunu eşzamanlı olarak yapabilir. Bu GPU'ların çekirdek sayısı kadar komut satırını aynı anda işlemesi sayesinde gerçek zamanlı DSA algoritmaları çalıştırılabilmektedir. Yani görüntüler çok sayıdaki filtrelerle eş zamanlı olarak filtreden geçirilerek görüntüye ait özellikler elde edilir. Görüntüye ait özellikler eğitilmiş veri seti ile DSA ile karşılaştırılarak tanımlaması ve sınıflandırılması gerçek zamanlı olarak yapılabilir. DSA'ların gerçek zamanlı görüntü işleme uygulamaları gelişen teknolojiye bağlı olarak GPU'ların hızları ve çekirdek sayıları daha da arttıkça buna bağlı olarak çok yüksek çözünürlüklü ardışık sayısal görüntülerin kullanılması yaygınlaşacaktır.

4.4.1. Jetson TX2 geliştirme kiti

Tez çalışması kapsamında geliştirilen NTİTS yönteminde hem OA algoritmalarının çözümünde hem de DSA'larının çalıştırılması için Nvidia tarafından geliştirilen Jetson TX2



Resim 4.5. Jetson TX2 modülü

Aşağıdaki Tablo 4.2’de Jetson™ TX2 Kitinin teknik özellikleri verilmiştir.

NVIDIA Jetson TX2 geliştirme kiti, gömülü platformu için geliştirme yapılabilmesi için donanımla uyumlu JetPack kurulum paketi oluşturmuştur. JetPack kurulum paketi kullanıcılara, YZ uygulamaları yapmak için DÖA’larda kullanılan TensorRT, cuDNN, NVIDIA DIGITS™ Workflow, Bilgisayarla görü için NVIDIA VisionWorks, OpenCV, GPU Hesaplamaları için NVIDIA CUDA, CUDA Kütüphaneleri, Çoklu görüntüleme ortamları için ISP Desteği, Camera imaging, Video CODEC, ayrıca, ROS uyumluluğu, OpenGL gibi gelişmiş geliştirici kapsamlı yazılım araçlarını içeren bir ortam sunar.

Çizelge 4.2. Jetson™ TX2 Kitinin teknik özellikleri

YZ Performansı	1.33 TFLOPs
GPU	256 Çekirdek NVIDIA® Pascal™ GPU mimarisi ile 256 NVIDIA® CUDA® Çekirdeği
CPU	Dual-Core NVIDIA® Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore HMP Dual Denver 2/2 MB L2+Quad ARM® A57/2MB L2
Video	4Kx2K 60Hz Encode (HEVC) 4Kx2K 60Hz Decode (12-Bit Support)
Hafıza	8GB 128-Bit LPDDR4 Memory 18666MHz-59.7GB/s
Görüntü	2xDSI, 2xDP 1.2/HDMI 2.0/eDP 1.4
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)
PCIE	Gen2 1x4+1x1 OR 2x1+1x2
Veri Depolama	32GB eMMC 5.1, SDIO, SATA
Harici Bağlantılar	CAN, UART, SPI, I2c, I2S, GPIOs
USB	USB3.0+USB2.0
Bağlanabilirlik	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
Boyutlar	50mmx87mm (400-Pin Karttan-Karta Bağlantı Uyumlu)
Güç	7.5W/15W

4.4.2. Kamera

Web Camera veya kısaltılmışı Webcam; Ağ anlamındaki Web sözcüğüne, kamera anlamındaki İngilizce Camera sözcüğünün kısaltması Cam'ın eklenmesiyle Webcam kısaltması olarak kullanılmaktadır. Webcam'ler ile doğrudan anlık olarak fotoğraf ve video çekim yapılabilir.

İlk Webcam, Cambridge Üniversitesi, Bilgisayar Bilimleri Bölümü'nde "XCoffee" ve "Trojan Room Coffe Pot" adıyla 1991 yılında geliştirilmiştir[158]. Bu kamera Cambridge Üniversitesinde kahve makinesini gözetlenmesini amacı ile her saniyede bir 124x124 piksellik görüntü karesi alabilen kamera geliştirilmiştir. Webcam'ler, bilgisayarlara genellikle USB kablosu ile bağlanır. Webcam'in objektifine gelen ışık demektir, optik düzenden geçtikten sonra ışık algılayıcı üzerine düşürülür. Burada ışık demeti sayısal sinyale dönüştürülür ve sayısal görüntü elde edilir. Elde edilen bu sayısal görüntü

bilgisayardan ekrana aktararak gerçek ortamın 2B'lu izdüşümü elde edilir. Tez çalışması kapsamında değişik çözünürlüklü görüntülerle çalışılmıştır. Bu değişik çözünürlüklü görüntüler önceden kayıt edilmiş sayısal görüntüleri ve Webcam'den alınan gerçek zamanlı görüntüleri de içermektedir. Gerçek zamanlı hareketli nesnelerin tespiti takibi tanımlaması ve sınıflandırmasının yapılabilmesi için sisteme entegre edilmiş sayısal kameraya da ihtiyaç vardır. Erişiminin kolay, kullanımının yaygın olması, maliyetinin düşük olması enerji sarfiyatının az olması, erişiminin kolay olması, tak kullan özelliği sayesinde kullanıcı dostu olması gibi nedenlerden dolayı yapılan çalışmalarda gerçek zamanlı sayısal görüntü işleme uygulamalarında Webcam kullanılmıştır.

4.4.3. Klavye

Klavye; üzerinde harfler, rakamlar, özel karakterler ve özel fonksiyon tuşlarının bulunduğu, bilgisayara komutları ve sayısal değerleri yazmamızı sağlayan harici bir giriş birimidir. Q klavye 1714 yılında Henry Mille adlı bir İngiliz mühendis tarafından görme engelliler için yaptığı bir makinaya dayanır[159]. 1868 yılında Christopher Latham Sholes patenti alınan ilk ticari klavyede en çok kullanılan harfleri belirli konumlara koyarak klavyenin işlevselliğini arttırmıştır[159]. Klavye, Jetson TX2 kitinini çalıştırmak ve iletişim kurmak geliştirilen NTİTS programını çalıştırmak ve yazılıma ulaşmak gerekli durumlarda müdahale etmek için kullanılmıştır.

4.4.4. Fare

Kullanılan ilk bilgisayar faresi prototipi Douglas Engelbart tarafından grafik kullanıcı arabirimi (GUI) "Windows" ile kullanılmak üzere 1964 yılında yapıldı[160]. Altta dikey olarak monte edilmiş iki disk bulunan basit bir mekanik cihazdır. Mükemmel yatay veya dikey çizgiler çizmek farenin hareketi ile mümkün olmuştu[160, 161]. Düz bir yüzeyde hareket ettirildikçe fare, hareketi ekrandaki imlecin hareketine dönüştürür. İmlecin hareketini sağlar. Ekrandaki öğeleri seçmeyi veya bir yerden başka bir yere taşımayı sağlayan bir giriş elemanıdır[161]. İki farklı tekerlek ile koordinat düzlemindeki hareketleri algılayan, bilgisayarın kursörünü hareket ettiren bu cihaza "fare" ismi, bilgisayar ile haberleşmede kullanılan kablosunun farenin kuyruğuna benzetilmesi sebebiyle bu isimle anılmaktadır[160]. Fare, Jetson TX2 kitinini çalıştırmak ve iletişim kurmak geliştirilen

NTİTS programını çalıştırmak ve yazılıma ulaşmak gerekli durumlarda müdahale etmek için kullanılmıştır.

4.4.5. Ekran

Elektronik ortamda, ışık yoluyla oluşturulan görüntülerin saydam olmayan düz yüzeylerde gösterilen aygıtların genel adıdır. 1973 yılında Don Lancaster 16 satırda 32 büyük harf karakterini gösterebilen bir ekranı icat etti[162]. Ekranlar bilgisayara girilen değerleri gördüğümüz ve bilgisayarın ürettiği sonuçları bize gösterdiği vazgeçilmez donanımlardandır. Teknolojini gelişmesi ekranlarda da adeta yeni bir çığır açmıştır. Yüksek çözünürlüklü esnek led ekranlar geliştirilmiştir. Ekranların kullanımı her geçen gün yaygınlaşmaktadır. Ekran, Jetson TX2 kitinini çalıştırılması, Jetson TX2 kitinini çalıştırılmasından elde edilen sonuçların kullanıcıya gösterilmesi için Jetson TX2 kiti ile iletişim kurmak için kullanılmıştır.

Tez çalışması kapsamında geliştirilen NTİTS yönteminde kullanılan Jetson TX' geliştirme kiti, Jetson TX' geliştirme kiti adaptörü, ekran, klavye, fare, webcam, donanımlar Şekil 4.5'da görülmektedir.



Resim 4.6. NTİTS yönteminde kullanılan donanımlar

5. GELİŞTİRİLEN NTİTS YÖNTEMİNİN ÇALIŞTIRILMASINDAN ELDE EDİLEN SONUÇLARIN İRDELENMESİ

Tez çalışmasının amacı kapsamında özgün Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Gerçek Zamanlı Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması (NTİTS) yöntemi geliştirilmiştir. Geliştirilen NTİTS yöntemi değişik çözünürlükteki ardışık sayısal görüntülerde çok sayıdaki hareketli nesneleri tespiti eder izler, tanımlar ve sınıflandırmasını yapar. NTİTS yöntemi değişik çözünürlükteki ardışık sayısal görüntülerde çok sayıdaki hareketli nesneleri tespiti eder izler, tanımlar ve sınıflandırmasını iki aşamada gerçekleştirir. Birinci aşamadan NTİTS yöntemi ile hareketli nesneleri, hareketsiz nesnelerden ve zeminden OA algoritması kullanılarak ayrıştırılır. Ayrıştırılan hareketli nesnelere ait bu görüntüler veri kaybı olmadan ait olduğu ana görüntüden kesilerek alınır. Böylelikle Ana görüntüde hareket eden kaç tane nesne var ise o nesnelere ait ana görüntüden her zaman için daha küçük sayısal görüntüler elde edilir. Elde edilen bu hareketli nesnelere ait görüntülerde sayı sınırı yoktur. Ana görüntüye sığan maksimum sayıdaki hareketli nesne OA ile tespit edilir ve bu nesneler ana görüntüden kesilerek alınır.

İkinci aşamada ana görüntüden kesilerek alınan hareketli nesnelere ait görüntülerin tanımlaması sınıflandırılması yapılır. Tanımlama ve sınıflandırma işlemleri DSA mimarilerinden biri olan ESA sınıfında olan TAÇKA metodu ile hareketli nesnelere ait sayısal görüntüler tanımlanarak sınıflandırılır.

OA ardışık sayısal görüntülerdeki her bir pikselin ışıklılık değerlerindeki değişimleri ardışık sayısal görüntülerdeki önceki görüntü ile sonraki görüntülere ait piksellerin değerlerinde meydana gelen değişimleri nesnelerin hareketlerinin yönünü ve şiddetini veren hız vektörleri olarak hesaplar. OA'dan elde edilen bu hareketli nesnelere ait görüntüler hız vektörlerine Bu bölümde, tez çalışması kapsamında geliştirilen NTİTS yönteminin çalışması sonucunda elde edilen veriler irdelenmiştir.

Aşağıdaki, Resim 5.1. a) ve b) 'de Londra'da Piccadilly Circus meydanının ardışık iki görüntüsü görülmektedir. Bu meydan yoğun bir buluşma yeri ve başlı başına bir turistik cazibe merkezi olan kavşaktır. Piccadilly Circus kavşağında çok yüksek çözünürlüklü bir sayısal kameradan alınan Resim 5.1. a) ve Resim 5.1 b) 'de gösterilen 3840x2160 piksel 4K çözünürlüğündeki ardışık iki görüntünün boyutlarının sayfaya sığabilmesi için küçültülmüş

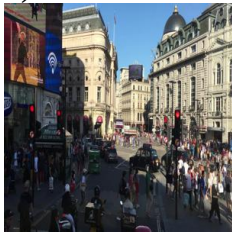
olmasına rağmen kaliteli ve net olduğu görülmektedir. Bu görüntülerde meydana yürüyen insanlar, hareket halindeki araçlar, duran araçlar, trafik ışıkları, tabelalar, ışıklı ekranlardaki görüntüler vb. gibi çok sayıda hareketli, hareketsiz, canlı, cansız pek çok nesne görülmektedir. Resim 5.1. c)'de ise Şekil 5.1. b)'deki 3840x2160 piksel, 4K çözünürlüğündeki görüntünün yeniden boyutlandırılarak 300x300 piksel boyutundaki görüntüye küçültülmüş hali görülmektedir. Resim 5.1. c) dikkatlice incelendiğinde bu görüntünün Resim 5.1. a) ve b)'deki görüntülerle mukayese edildiğinde görüntü netliğinin ve kalitesinin çok daha az olduğu görüntü nesnelerin görüntülerinin bulanıklaştığı kalabalığın yoğun olduğu yerlerde bulunan insanlara ait detayların kaybolduğu ve görüntünün gözün görme kalitesinin ne kadar düşük olduğu ve Resim 5.1. c)'deki görüntünün gözü yorduğu görülmektedir. Çünkü 3840x2160 piksel ile ifade edilen 4K kalitesinde ki bir sayısal görüntü 8.294.400 adet pikselden oluşurken, bu görüntü yeniden boyutlandırılarak 300x300 piksel boyutundaki görüntüye küçültüldüğünde 90.000 piksel ile ifade edilir. Yani görüntünün içerdiği sayısal veri miktarı 92,16 kat azaltılmış olur. Bu nedenle görüntü netliği azalır, görüntüye ait pek çok detay ortadan kaybolur bu nedenle bazı küçük nesneler ya tamamen kaybolur ya da ne olduğu anlaşılamayacak kadar küçüldüğü için ne olduğu bilinemez ya da başka nesnelerden ayırt etmesi mümkün olmayabilir.



a)



b)



c)

Resim 5.1. a),b) sayısal kameradan alınan 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntü görülmektedir, c) 3840x2160 piksel 4K çözünürlüğündeki sayısal görüntünün yeniden boyutlandırılarak 300x300 piksel çözünürlüğündeki görüntü

Sayısal kamera yardımıyla 3B'li gerçek ortamdan alınan 3840x2160 piksel 4K çözünürlüğündeki renkli sayısal görüntüler Resim 5.1 a) ve b) görülmektedir. Bu renkli iki ardışık sayısal görüntüdeki hareketli alanlarının tespit edilmesi OA algoritması ile gerçekleştirilir. OA her bir pikselin ışıklılık değerlerindeki değişimlerin hesaplar. Bu görüntü için $2 \times 3 \times 3840 \times 2160 = 49.766.400$ byte'lık veri demektir. Böyle büyük görüntülerdeki hareketli nesnelerin veri kaybı olmadan işlenmesi hareketli nesnelerin tanımlaması ve sınıflandırılması işlemlerinin başarısı açısından oldukça önemlidir. Nesnelerin hareketlerinin yönünü ve şiddetini hız vektörleri olarak hesaplar. Bu hız vektörlerinin olduğu bölgeler hareketin olduğu yerleri belirtir. Hız vektörlerinin yoğun olduğu yerlerde hareketli piksellerin çok olduğu anlamındadır. Vektörlerin büyük olduğu piksellerin hızının diğerlerine göre fazla olduğunu ifade eder.



Resim 5.2. 3840x2160 piksel 4K çözünürlüğündeki hareket eden nesnelerin OA'dan elde edilen hız vektörleri

Resim 5.1 a) ve b)'deki 3840x2160 piksel 4K çözünürlükteki ardışık iki sayısal görüntünün OA'dan elde edilen hareket vektörlerinin Resim 5.1 b) üzerine çizdirilerek hareket eden nesnelere ait hız vektörlerinin Resim 5.2'de hareketli nesneler üzerine çizdirilerek gösterilmiştir. Resim 5.2'ye yakından bakıldığında OA vektörlerinin mavi oklarla hareket eden nesnelerin etrafında oldukları daha iyi görülecektir. Hareketin olmadığı nesnelerin üzerinde okların olmadığı, hareketin olduğu nesnelerin üzerinde mavi okların olduğu görülecektir. Hareketin çok olduğu nesnelerin üzerinde mavi okların daha belirgin ve büyük

olduğu yani hareket eden nesnelerden hızlı olanların etraflarındaki hız vektörlerinin daha büyük ve belirgin olduğu görülecektir.

Resim 5.3'e yakından bakıldığında 3840x2160 piksel 4K çözünürlüğündeki Resim 5.1'deki ardışık sayısal iki görüntünün OA ile analizinden elde edilmiş hareket vektörlerinin Resim 5.2'de çizdirilmiş olan görüntüsüne yakından bakıldığında bazı bölgelerde hareketli nesnelerin olmadığı yerlerde optik akış vektörlerinin çizdirilmiş olduğu görülmektedir. Bu bölgelerdeki Led ışıklı ekrandaki görüntüdeki değişimler ve binanın yüzeyine gelip yansıyan güneş ışınlarının meydana getirdiği yansımaların sebep olduğu değişimleri OA algoritması hesaplamakta ve bunları hız vektörleri olarak ifade ettiğini göstermektedir.



Resim 5.3. 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün hareketli nesneler nesnelerin olmadığı durumda optik akış vektörleri

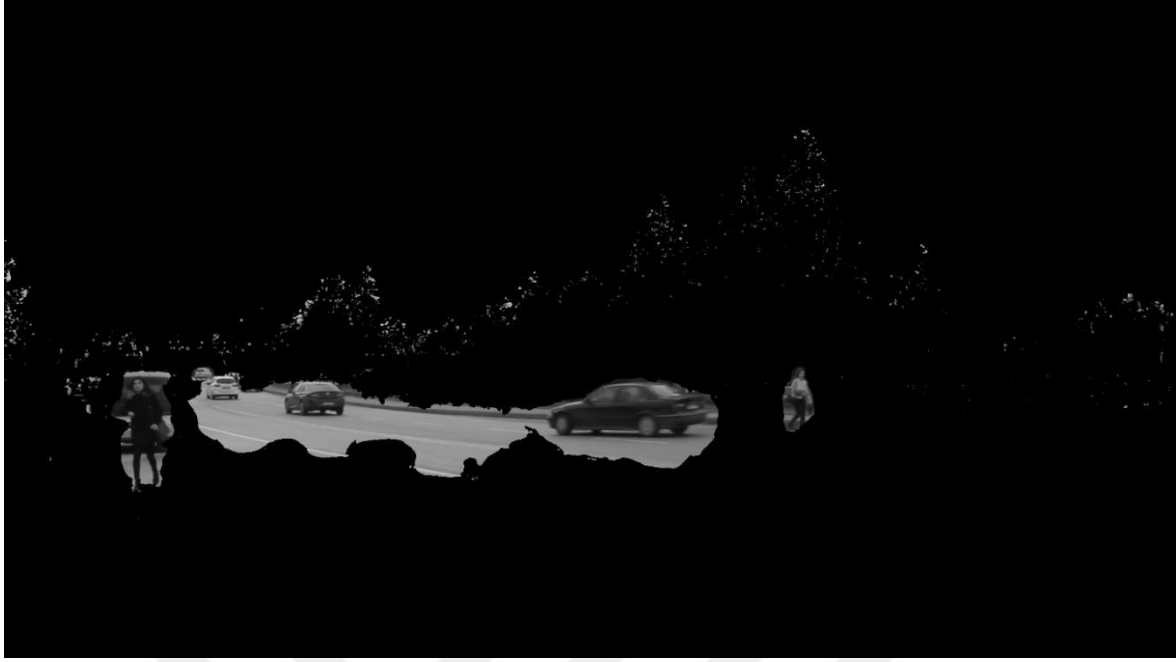
Aşağıdaki, Resim 5.4'e bakıldığında yine Şekil 5.1'deki 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün OA ile analizi sonucu elde edilmiş olan Şekil 5.2'nin yakından bakılarak hareketli nesnelerin ve hareketsiz nesnelerin olduğu alandaki optik akış vektörleri görülmektedir. Resim 5.4'de trafik ışıklarının olduğu yerde karşıdan karşıya geçen yayaların hareketleri yayaların üzerinde mavi oklarla gösterilmiştir. Kırmızı ışıktaki yayaların geçmesini bekleyen hareketsiz duran motosiklet sürücülerinin etrafında hareket vektörlerini ifade eden mavi oklar yoktur. OA'nın etkin bir şekilde hareket eden nesneler ile hareket etmeyen nesneleri ayırt ettiği görülmektedir. Yine ışıktaki bekleyen

gri renkteki araba hareketsiz olduđu için etrafında OA hız vektörlerini ifade eden mavi oklar yoktur ancak gri arabanın arkasından, gri arabaya doğru hareket etmekte olan siyah arabanın etrafında, siyah arabanın hareketini gösteren OA hız vektörleri yani mavi oklar görülmektedir. Hareket eden nesnelerin hızları ne kadar büyük ise (yani piksellerin ışıklılık değışimleri) hız o kadar büyük olur. Hareket ne kadar büyük ise hareketi ifade eden vektörler de o kadar büyük olur. Resim 5.4'e dikkatli bakıldığında ışıklarda karşıdan karşıya geçen yayaların etrafında mavi okların olduđu görülmektedir. Işıktaki bekleyen hareketsiz motorlu kuryelerin etrafında hareket olmadığı için mavi oklar yoktur.



Resim 5.4. 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün hareket eden ve hareket etmeyen nesnelerin olduđu bölgeye yakından bakıldığında optik akış vektörleri

Tez çalışması kapsamında geliştirilen NTİTS yönteminin ilgi alanı harekettir. OA algoritması ile ardışık sayısal görüntülerde hareket eden nesnelere ait hız vektörleri tespit edilmektedir. Tespit edilen bu hız vektörlerine karşılık gelen pikseller gruplandırılarak anlamlandırılmaktadır. Aşağıdaki, Resim 5.5'de 1920x1080 çözünürlüğündeki sayısal görüntülere ait OA algoritmasından elde edilen hareketli nesnelere ait hız vektörlerin karşılık geldiği piksellerin gruplandırılarak anlamlandırıldığı görüntü görülmektedir.

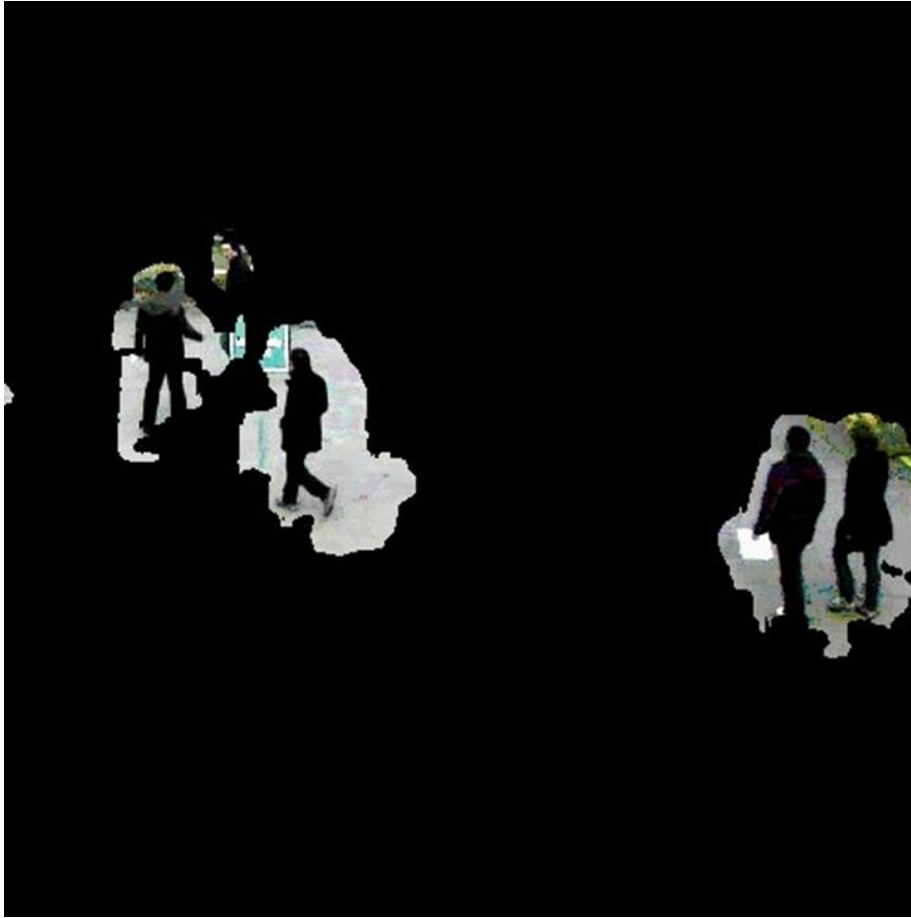


Resim 5.5. 1920x1080 çözünürlükteki sayısal görüntülerde OA algoritmasından elde edilen hareketli nesnelere ait hız vektörlerin karşılık geldiği piksellere ait görüntü

Yukarıdaki, Resim 5.5’de 1920x1080 piksel çözünürlüğündeki ardışık sayısal görüntüde OA algoritmasından elde edilen hareketli nesnelere ait vektörlerin karşılık geldiği piksellerin anlamlandırılarak gruplandırılması sonucunda bu piksel gruplarında bulunan görüntüleri göstermektedir. Resim 5.5’deki görüntünün sol tarafında karşıdan karşıya geçen kadın ve kadının arkasından ilerleyen bir araç görülmektedir. Yolda hareket halinde sıra ile giden araçlar görülmektedir. Resmin sağ tarafında ise kaldırımın kenarında yürüyen başka bir kadın görülmektedir. Şeklin üst tarafında küçük küçük beyaz alanlar görülmektedir. Bu parçalı küçük piksel grupları yol kenarında bulunan ağaçların dalları ve yapraklarıdır. Yol kenarındaki bu ağaçların dalları ve yaprakları rüzgar ile hareket etmektedir. Ağaçların dalları ve yapraklarının hareketi OA algoritması ile tespit edilmiştir. Şöyle ki OA algoritması her bir pikseldeki ışıklılık değerlerindeki değişimleri hesaplamaktadır. Piksellerin ışıklılık değerlerindeki değişimlere karşılık gelen vektörler ağaç dallarının ve yapraklarının rüzgardaki dalgalanma hareketine karşılık geldiği alanları da tespit etmiştir. Bu alanlar hız vektörlerinin ikili (binary) görüntülerde inceltme, gürültü giderme gibi görüntü işleme yöntemleri ile işlenerek faydalı bilginin olduğu hareketli nesnelere ait değerli bilgiler elde edilir.

Aşağıdaki, Resim 5.6’da ardışık sayısal görüntülerde yolda yürüyen insanların OA yöntemi ile hareketli alanların hız vektörlerinin ikili (binary) görüntülerde inceltme, gürültü giderme

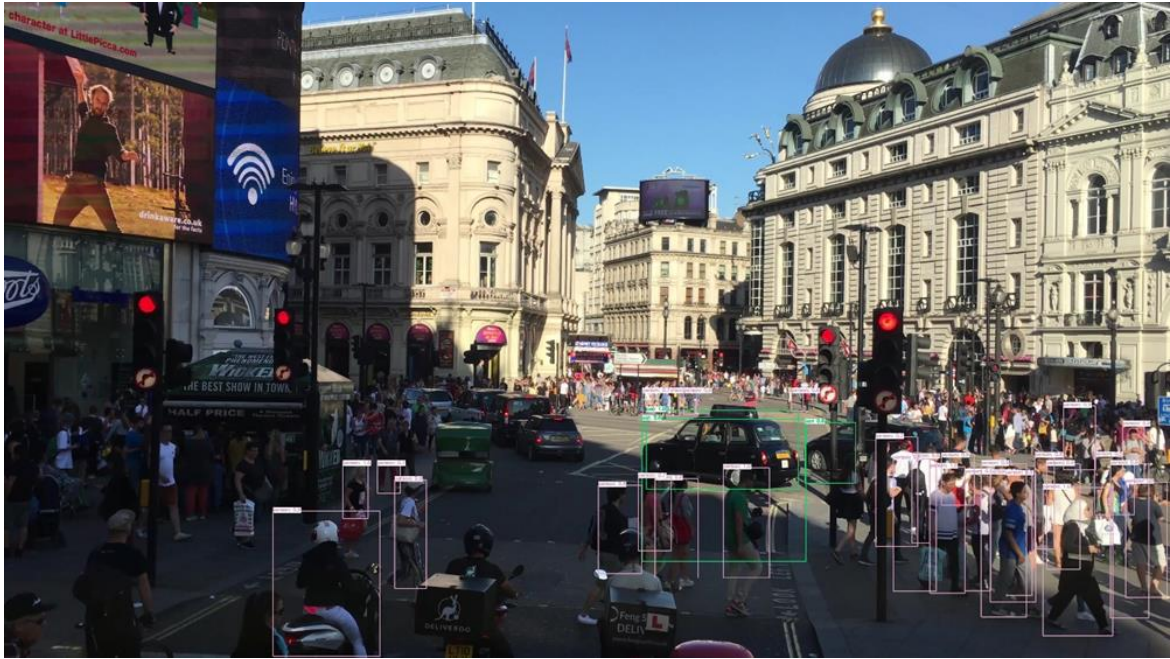
gibi görüntü işleme yöntemleri ile işlenerek faydalı bilginin olduğu hareketli nesnelere ait görüntülerin elde edilmesi görülmektedir. Resim 5.6 dikkatli bir şekilde incelendiğinde sol tarafta yürüyen insanlar görülmektedir. Resim 5.6 'da değişimin olduğu piksel grupları arka plandan ve hareketsiz olan diğer nesnelerden ayrıştırılarak yolda yürüyen insanlara karşılık geldiğinden tespit edilmiştir. Yoldaki trafik tabelasının arkasında kalan insan yürürken tabelanın arkasından çıktığı an görülmektedir. Yürürken tam tabelanın arkasından çıkarken dizini kaldırdığı andaki görüntüsü tespit edilmiştir. Resim 5.6'da sağ tarafta yürüyen iki yaya görülmektedir.



Resim 5.6. Yolda yürüyen insanların OA yöntemi ile tespit edilmesi

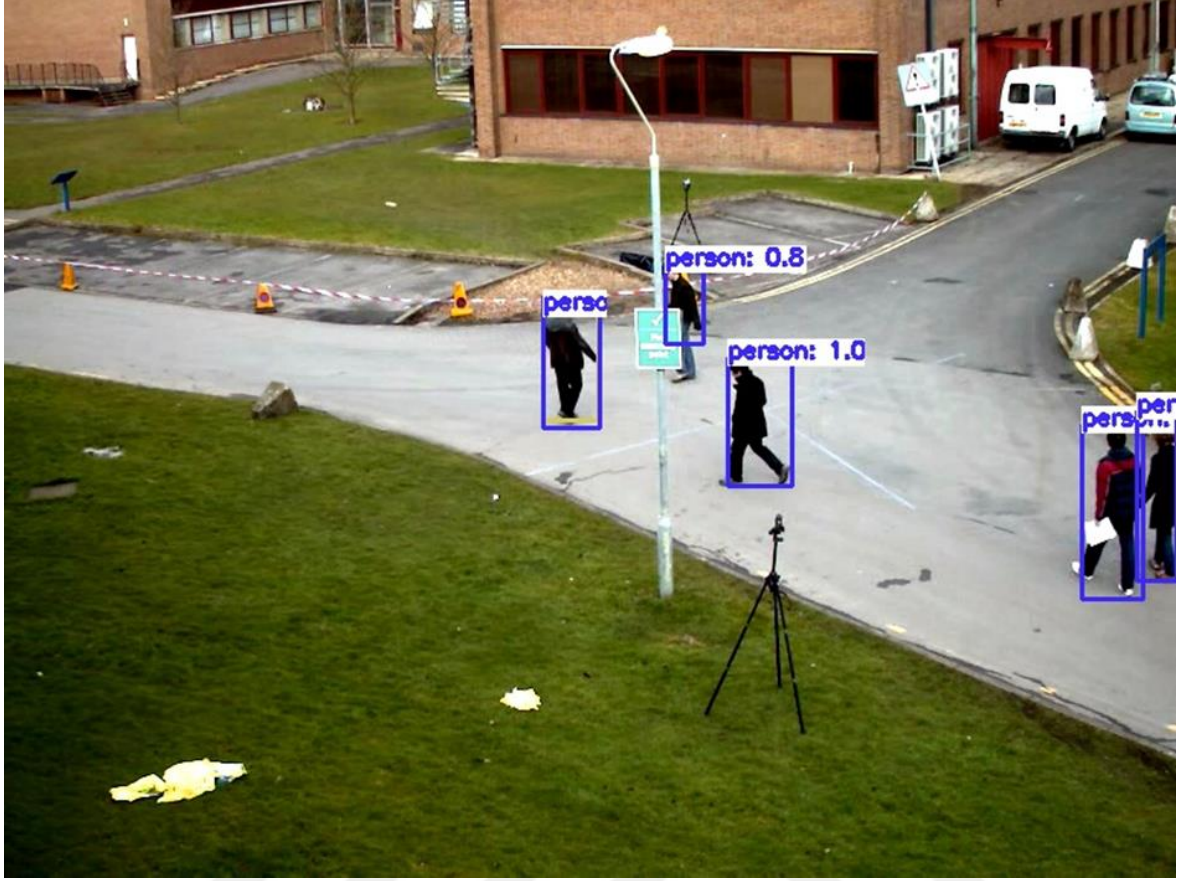
Aşağıdaki Resim 5.7'de Resim 7.1. a) ve b)'de Londra'da Piccadilly Circus meydanından alınmış 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal iki görüntünün NTİTS yöntemi ile işlenerek elde edilen sonuç görüntüsü görülmektedir. Bu görüntüde meydana yürüyen insanlar, hareket halindeki araçlar, duran araçlar, trafik ışıkları, tabelalar, ekranlardaki görüntüler vb. gibi hareketli hareketsiz nesneler görülmektedir. Tez çalışması kapsamında geliştirilen NTİTS yazılımı hareketli nesneleri hareketsiz nesnelerden OA ile

hareketli nesneler hareketsiz nesnelerden ve arka plandan ayrıştırılarak tanımlamasını ve sınıflandırmasını bir ESA olan TAÇKA ile analizinden elde ettiği sonuçları görüntü üzerine yazarak kullanıcıya gösterilmiştir. Resim 5.7’de görüleceği üzere ardışık sayısal görüntülere sığabilen çok sayıdaki canlı cansız hareketli tez çalışması kapsamında geliştirilen NTİTS yöntemi ile başarılı bir şekilde tespit edilmiştir. Jetson TX2 geliştirme kiti NTİTS yazılımı ile 3840x2160 piksel 4K çözünürlüğündeki ardışık sayısal görüntüleri gerçek zamanlı olarak işlenmemiştir.



Resim 5.7. 3840x2160 piksel 4K çözünürlüğündeki sayısal görüntüde hareketli nesnelerin hareketsiz nesnelerden ayrıştırılıp tespit edilip tanımlanması ve sınıflandırılması gösterilmiştir

Aşağıdaki, Resim 5.8’de yolda yürüyen insanlar NTİTS yöntem ile tespit ediliyor. Resim 5.8’de sağ tarafta yürüyen iki insan vardır. Bu iki insanın sağındaki kadının boyun bitim yerinden itibaren sağ omuzunun olduğu bir kısmı görüntüde olmamasına rağmen DSA bu hareketli nesnenin insan olduğunu tespit ediyor. Görüntü dikkatlice incelenecek olursa görüntülerdeki kişilerin detayları azdır. Kollar bacaklar kafa gibi organların bilgilerini kullanarak insan olduğu tespitini başarılı bir şekilde yapıyor.



Resim 5.8. Yolda yürüyen insanlar NTİTS yöntemi ile tespit ediliyor

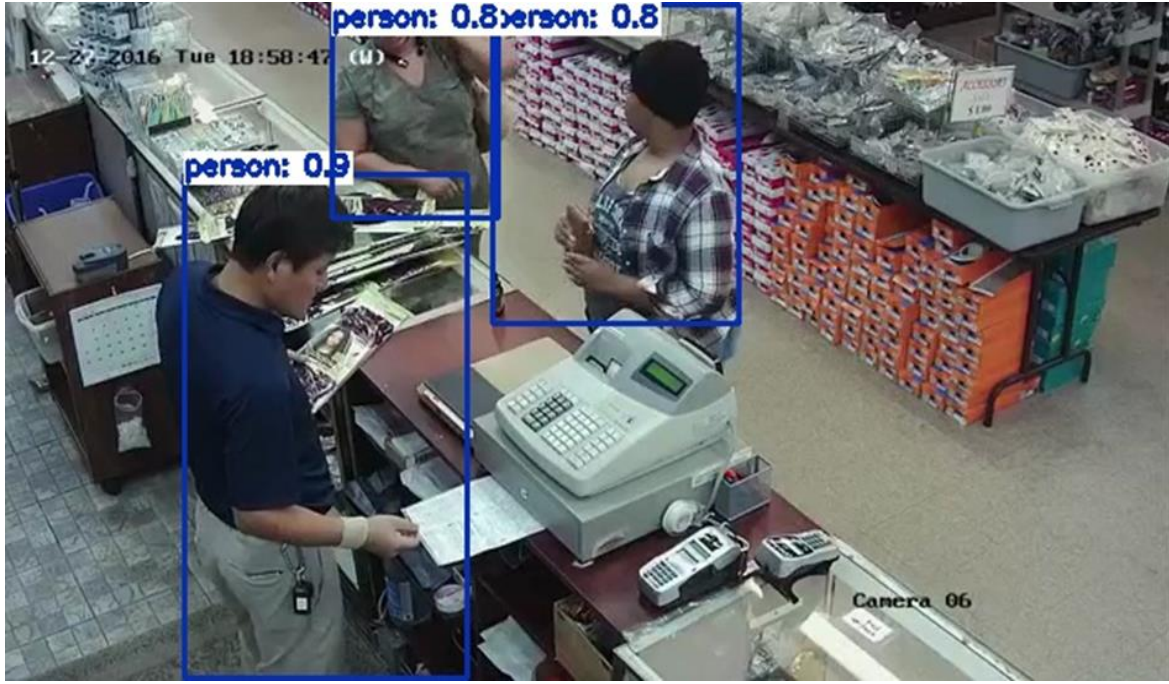
Aşağıdaki, Resim 5.9’da görüntü Şekil 5.5’deki sağ tarafta kaldırımda yürüyen kadına ait görüntüdür. 1920x1080 çözünürlükteki ardışık sayısal görüntülerin NTİTS yöntemi ile analizi sonucunda OA algoritmasından elde edilen hareketli nesnelere ait görüntülerin TAÇKA ile analizi sonucunda 63x12 pikse boyutlarında ki görüntünün analizi sonucunda görüntüdeki hareketli nesnenin yürüyen kadının olduğunu tespit edildiği Resim 5.9’da gösterilmiştir.



Resim 5.9. NTİTS ile analizi sonucunda görüntüdeki insanın tanımlanması

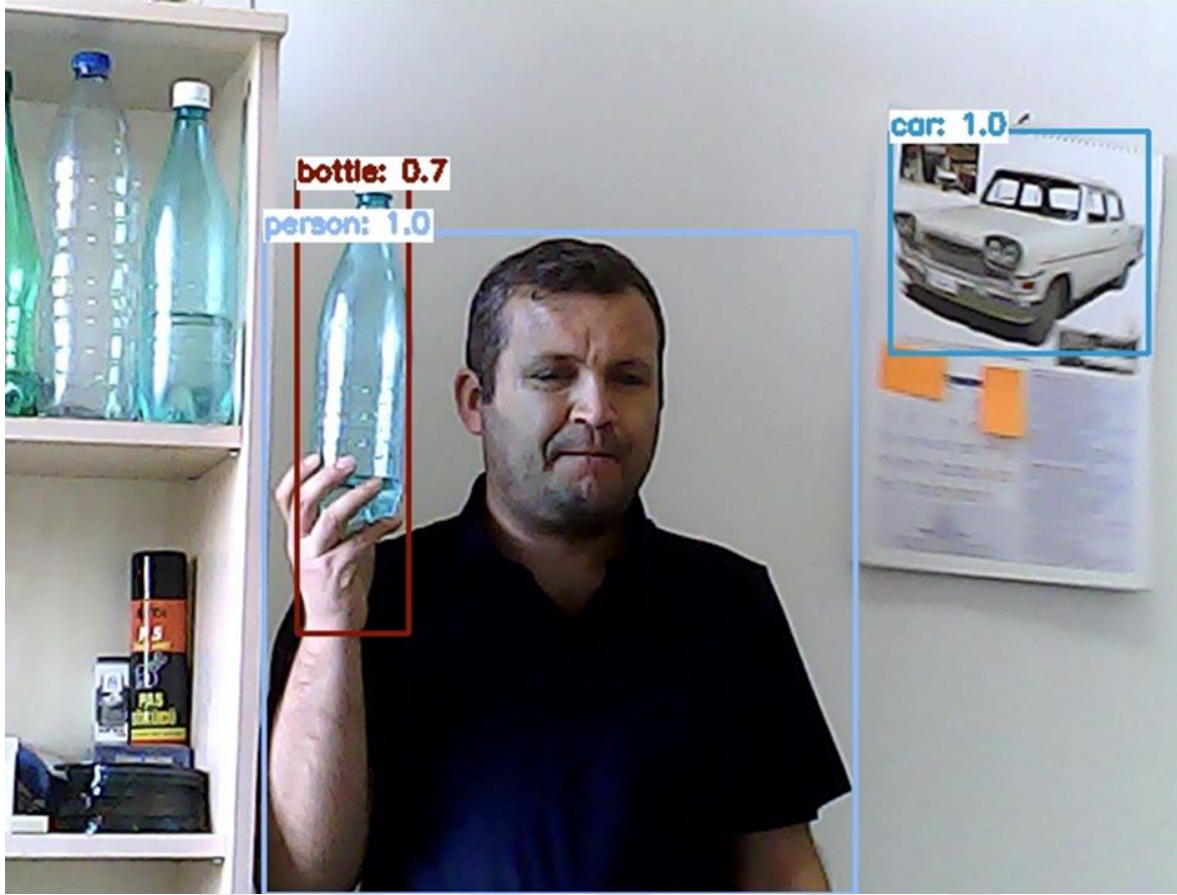
Aşağıdaki Resim 5.10’da bir mağazada kasiyer ve iki arkadaşın alışveriş yaparken ki görüntüleri görülmektedir. Bu görüntüde kasiyer vücudunun hepsini hareket ettirmeden

sabit durduğu yerde sağ eli ile listeye bakarken NTİTS yöntemi kasiyerin hareketli nesne olarak bir kısmını almış ve kasiyerin diz kapaklarından aşağısı yoktur. Geliştirilen sistem yüzde 90 insan olduğunu belirtmiştir. Kasiyerin önündeki iki müşteri ise birbirleri ile konuşmaktadır ve bu müşterilerinde belden aşağısı görünmemektedir NTİTS yazılımı yüzde 80 insan olduğunu tespit etmiştir.



Resim 5.10. Bir mağazadaki insanların NTİTS yöntemi ile tespit edilmesi

Aşağıdaki, Resim 5.11’de sol tarafta rafta duran şişeler vardır. Resmin ortasındaki insanın elinde su şişesi vardır. İnsan hareket halinde iken NTİTS yöntemi insanı belden aşağısı olamamasına rağmen insanın yüzünden ve ellinden tespit ederek yakalamış ve yüzde yüz insan demiştir. İnsanın elinde tuttuğu su şişesini hareketli nesne olarak tespit ederken insanın el bileğinin hareketi ile bilek ile birlikte su şişesini tespit etmiştir. Sınırlayıcı kutular su şişesinin yerini tespit ederken insan elini bileğine kadar olan bölgeyi seçmiştir. Birlik üzerinde kesişim oranı 0,5’den büyük olduğu için doğru bölge olarak alınmıştır. Tanımlama kısmında ise yüzde yetmiş şişe tanımlaması yapılmıştır. Şekil 5.11’in sağ tarafındaki duvarda asılı duran takvim karşılıklı kapı ve pencerenin açık olduğundan hava akımına bağlı olarak hareket etmektedir. Takvimin bu hareketi NTİTS yöntemi tarafından tespit edilerek takvim üzerindeki devrim otomobili tespit edilmiş ve yüzde yüz otomobil olduğu belirtilmiştir.



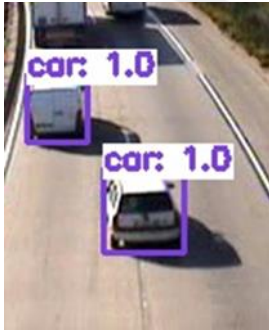
Resim 5.11. Hareketli nesnelerin hareketsiz nesnelerden ve arka plandan ayrıştırılması

1920x1080 çözünürlüklü sayısal kamera ile dış ortamda elde edilmiş ardışık sayısal görüntüler NTİTS yöntemiyle tespit edilen 111x138 pikselden oluşan hareketli alan Resim 5.12’de görülmektedir. Resim 5.12’deki görüntüdeki hareketli bölgede yürümekte olan kadın görülmekte ve kadının arkasında otoparkta park halinde duran otomobiller görülmektedir. NTİTS yöntemiyle tespit edilen 111x138 piksellik alanda sadece hareket halinde olan kadın olmasına rağmen hareket halindeki kadının olduğu bölgede bulunan NTİTS yönteminin tanımlamasını yapabileceği nesneler var ise onlarında tanımlamasını yapmaktadır. Resim 5.12 dikkatli bir biçimde incelenecek olursa park halinde olan otomobillere ait görüntü verisi çok az olmasına rağmen otomobil olarak tespitini yapabilmıştır.



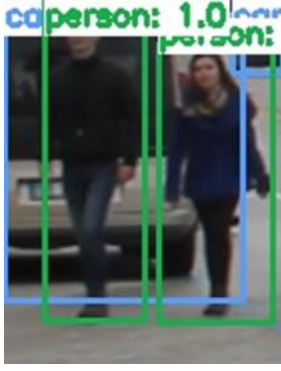
Resim 5.12. Yürümekte olan kadın

133x164 piksellik hareketli alan Resim 5.13’de görülmektedir. Resim 5.13’deki görüntüdeki otoyolda seyahat halindeki araçlar vardır. Resim 5.13’de hareketli bölgede seyahat halindeki iki otomobili NTİTS yöntemi tespit edip tanımlamasını %100 oranında otomobil olarak yapmıştır.



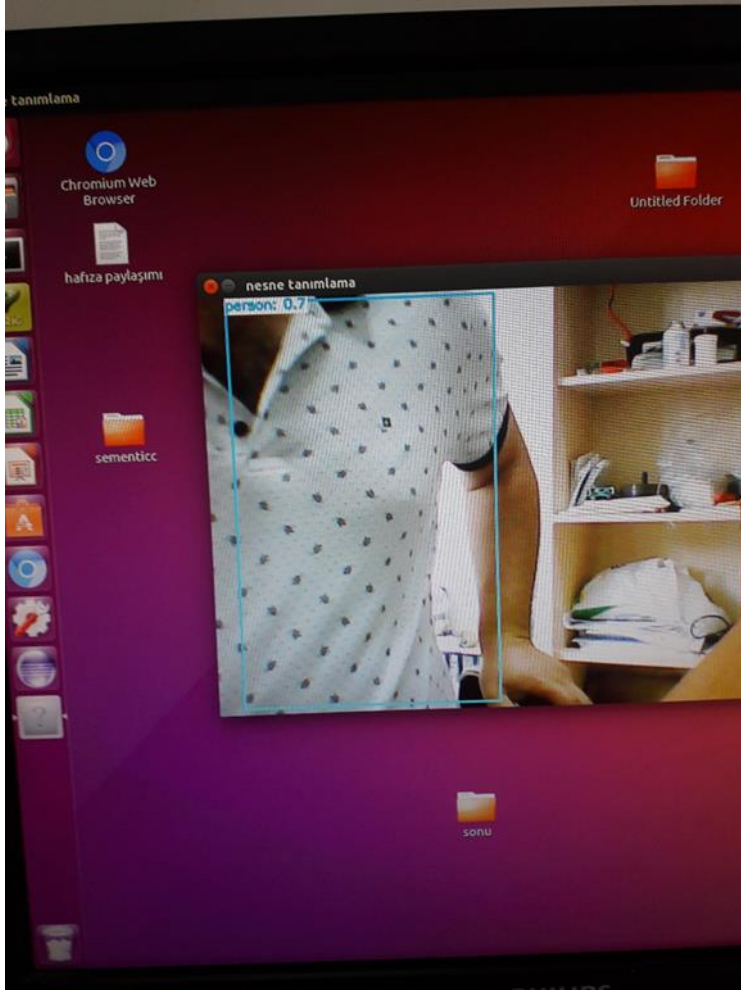
Resim 5.13. Geliştirilen NTİTS yönteminin çalıştırılmasıyla otoyolda seyahat halinde giden otomobillerin tespiti tanımlanması ve sınıflandırılmasının yapılması

Resim 5.14’de 139x181 piksellik görüntü görülmektedir. Resim 5.14’de, 1920x1080 çözünürlüklü sayısal kamera ile dış ortamda elde edilmiş ardışık sayısal görüntüler NTİTS yöntemiyle tespit edilen hareketli alanda resim sağ tarafında yürüyen kadının kafasının olduğu hizada seyahat halindeki otomobil, yürüyen kadın ve adam, kadın ve adamı NTİTS yöntemi tespit etmektedir. Yürüyen kadın ve adamın arkasında park halinde bulunan otomobil ise yürüyen kadın ve adamın arkasında hareketli alanda olduğu için hareketsiz halinde olmasına rağmen NTİTS yöntemi tarafından tanımlaması yapılmaktadır. Resim 5.14’de tanımlamasının yapıldığı görülmektedir.

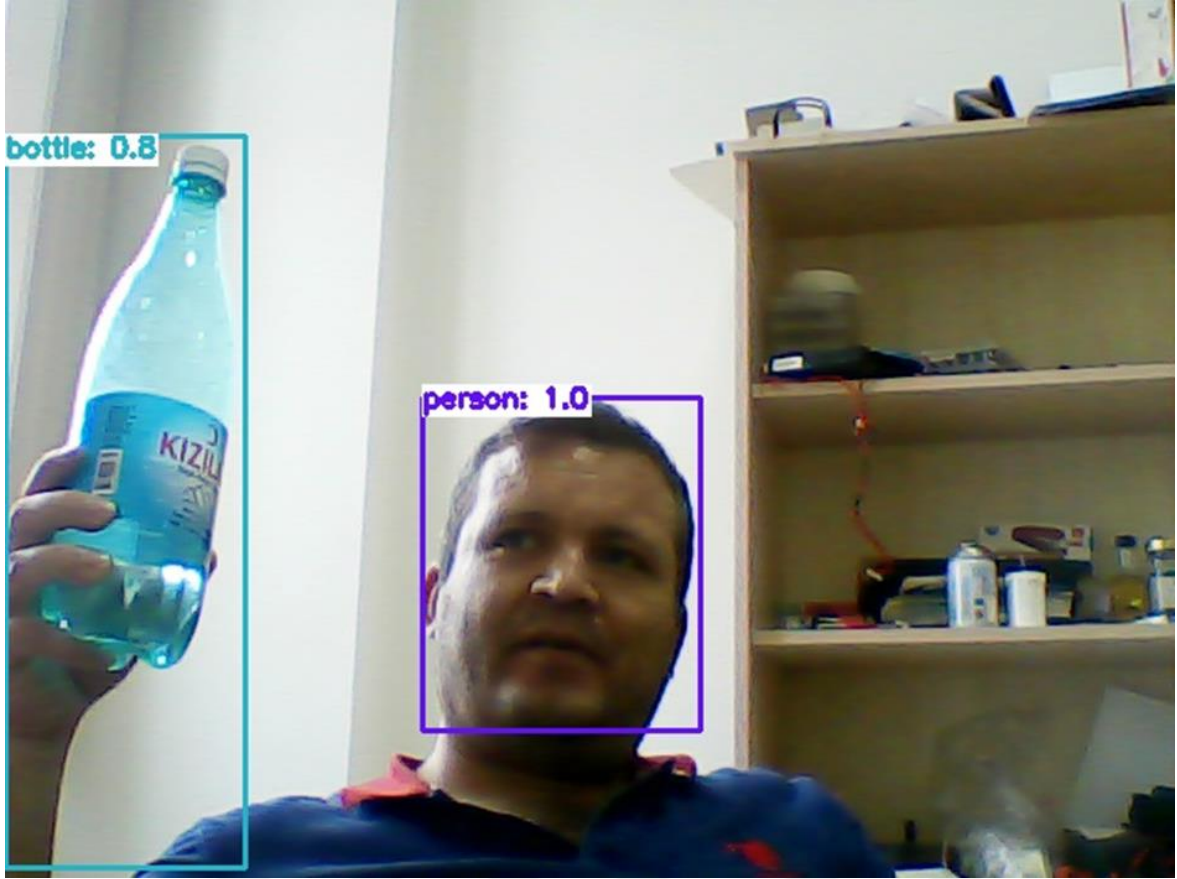


Resim 5.14. Yolun kenarında yürüyen insanlar

Aşağıdaki Resim 5.15’de turkuaz çerçeve içindeki alanda vücudunun sadece tek kolu, omuz, göbek kısmı arasında bulunan hareketli bölgede kalan alandaki hareketli bölgeyi %70 oranında insan olduğunu tespit edebilmektedir. Resim 5.15 DÖ’nün verileri tek bir katmanda yerine pek çok katmanda hesapları tek bir seferde yapabilen hatta tanımlanması gereken parametreleri bile kendisi keşfedebilen, belki de daha iyi parametreler ile değerlendirmelerde bulunabilen bir sistemdir. Analiz edilecek sayısal görüntüleri alıp aynı anda yüzlerce hatta binlerce GPU çekirdeğini paralel çalıştırarak görüntüye ait öz niteliklerini çıkararak bu öz nitelikleri daha önceden eğitilmiş veri seti ile karşılaştırıp nesnenin analizini yapıp nesnenin bir bölgesi olmasa bile tanımlamasını başarılı bir şekilde yapabilmektedir.

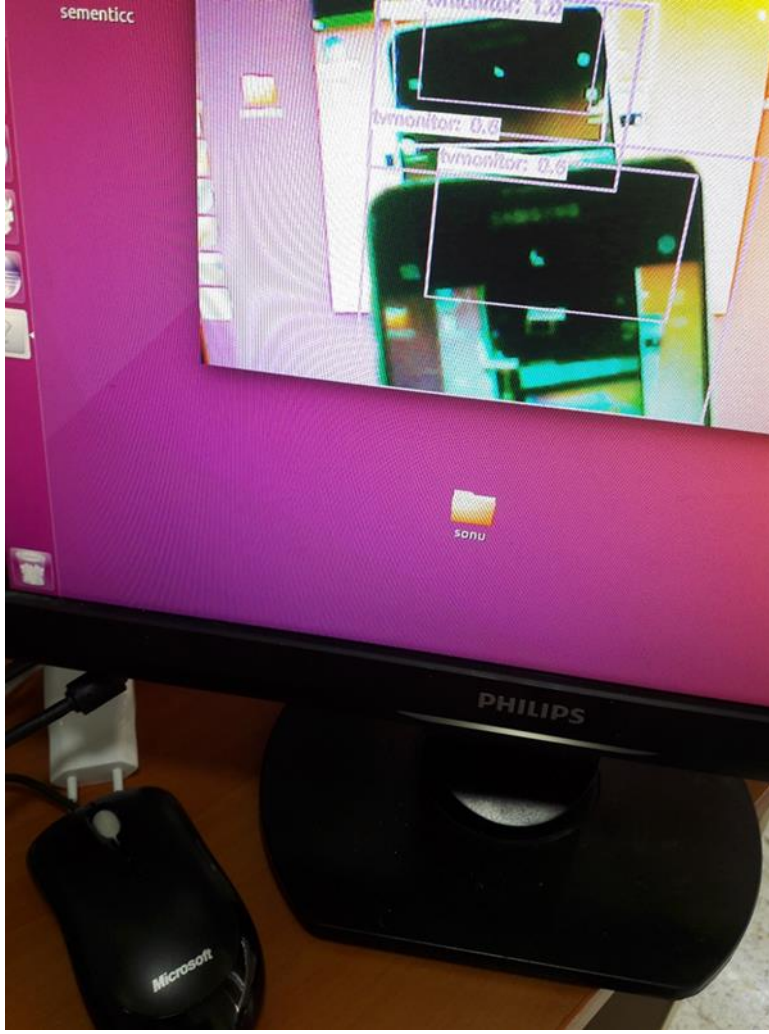


Resim 5.15. Vücudunun büyük bir kısmı olmayan hareketli insanın tanımlanması



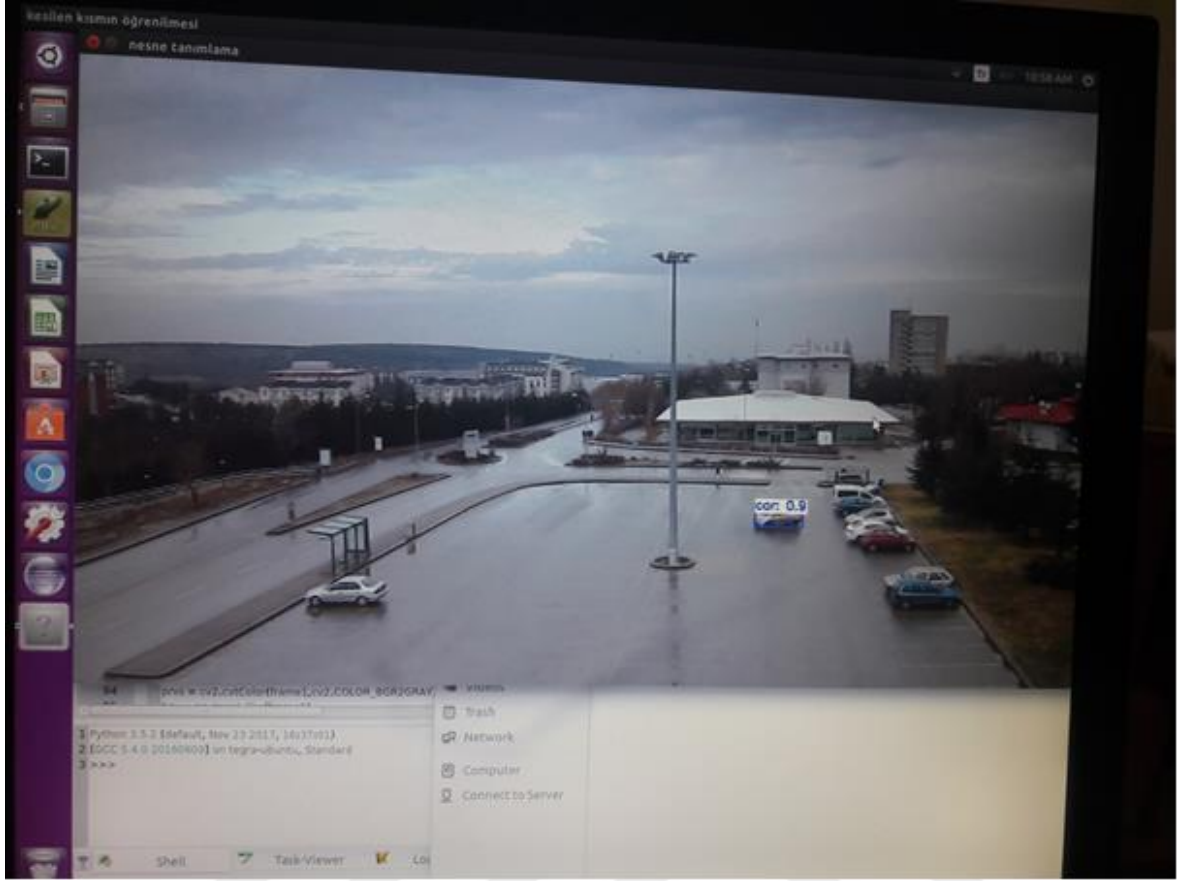
Resim 5.16. Sadece omuz bölgesi ve baş kısmı olan hareketli insanın ve elinde tuttuğu su şişesinin tanımlanması

Yukarıdaki Şekil 5.16'da görülen resimde sadece baş kısmının hareket ettiği hareketlinin %100 insan olarak tanımlanması ve kişinin elinde tuttuğu su şişesinin tanımlanması görülmektedir. Su şişesini tanımlanmasında TAÇKA yönteminin çoklu kutu atışı ile tespitinde yaptığı kutu uyarılama işlemi jaccard indeksine göre yüzde 50 sınırından büyük olduğu için kabul edilmiş ve bu alan içindeki şişenin üzerine yansıyan ışıklar vb. gibi bozucu etkilerden dolayı %70 oranında şişe olarak tespit etmiştir.



Resim 5.17. Ardışık sayısal görüntülerde iç içe geçmiş hareketli nesnelerin tanımlanması

Resim 5.17’de NTİTS sitemi çalışmakta ve çalışması kullanıcı ekranında görülmektedir. Kullanıcı ekranı ve NTİTS sisteminde kullanılan sayısal kamera arasına sayısal kamerası açık cep telefonu geldiğinde kamera cep telefonunu görmekte ve kameranın algıladığı cep telefonu NTİTS sistemiyle tv monitörü olarak tespit edilmekte ve sonucu kullanıcı ekranında gösterilmektedir. Kullanıcı ekranı da cep telefonu kamerası tarafından görüntülenmektedir ve cep telefonu ekranında NTİTS sisteminin üretmiş olduğu sonuç görülmektedir. Kullanıcı ve cep telefonu NTİTS sisteminde kullanılan kamera tarafından görüntülenmeye devam ettiğinden NTİTS sisteminin ürettiği sonuçlarla birlikte görüntü olarak NTİTS sistemine gönderilmektedir. NTİTS sistemi tekrar kameradan aldığı görüntüleri işleyerek hareketli alanların tespitini yapıp tanımlama ve sınıflandırması yaptığında elde edilen sonuç kullanıcı ekranı ve cep telefonunda görülmektedir. Resim 5.17’de de görüleceği üzere tez çalışması kapsamında geliştirilen NTİTİS yöntemi ardışık sayısal görüntülerde iç içe geçmiş hareketli nesnelerde tespit edilmektedir.



Resim 5.18. Otoparkta park etmeye çalışan otomobil

Yukarıdaki Şekil 5.18’de 1280x720 çözünürlükte yaklaşık 150m uzaklıkta açık otoparkta park etmeye çalışan otomobil geliştirilen NTITS yöntemi tarafından tespit edilip tanımlanması sınıflandırmasının yapıp kullanıcı ekranında gösterildiği sonuç resmi gösterilmektedir.

NTITS yönteminin geliştirilmesinde doğadan esinlenerek, insan gözünün çalışması prensibi modellenmiştir. Şöyle ki; insan gözü baktığı bölgedeki her yeri aynı çözünürlükte görmez. İnsan gözünün baktığı yerde ki görüş alanının içindeki merkezi çok yüksek çözünürlüklü olarak görür. Görüş alanının merkezinden kenarlara doğru gidildikçe, insan gözünün gördüğü görüntünün çözünürlüğü azalmaya başlar. En nihayetinde görüş alanının en uç noktalarında görüntü en düşük çözünürlükte olur. Görüş alanının uç bölgelerinde bulunan nesnelerin detaylarını insan gözü göremez siluet halinde görür. Bu en uç noktalarda insanın dikkatini çeken bir değişiklik olduğunda bilerek ya da gayri ihtiyari insan gözü o noktaya hareket ettirildiğinde bu uç bölge bir anda en yüksek çözünürlüklü olur iken az önceki görüş alanını merkezi olan çok yüksek çözünürlüklü olan bölgenin çözünürlüğü birden azalarak

çok daha az çözünürlüklü hale gelir. Az çözünürlüklü hale gelen bu bölgenin detayları kaybolur ve buranın görüntüsü siluet haline gelir. İnsan gözü sadece ilgi alanı olan bölgeyi en yüksek çözünürlüklü görerek bu alan ile ilgili tüm bilgiyi almaya çalışır. Böylelikle insan gözü, görüş alanında her bölgeyi eşit çözünürlüklü görmediği için beyne gereksiz bilgilerin gönderilmesini engeller ve beyni gereksiz verilerle yormaz.

İnsan gözünün bu çalışma prensibinden esinlenilerek sayısal kameralardan alınan sayısal görüntüleri içeren tüm pikseller dikkate alınacaktır ancak görüntünün tümüyle değil de sadece ilgi alanını içeren bölge ile çalışılacaktır. İlgi alanı olan bölge tespit edilerek ana görüntüden ayrıştırılacak ve bu ilgi alanındaki veriler analiz edilecektir. Analiz kısmında sadece gerekli olan belirli bir kısmı ile çalışılacağı için bu kısım her zaman ana görüntüden daha küçük olacaktır. Analiz edilecek görüntülerde veri kaybı hiç olmayacak ya da en minimum seviyede olacaktır. Böylelikle daha çok veriyle çalışılacak ve daha az iş yükü analiz işlemi tamamlanacaktır. Bu nedenle geliştirilen NTİTS yöntemi sistem gereksinimleri az seviyede tutarak gerçek zamanlı görüntü işleme mümkün olacak ayrıca kurulum maliyetleri azalacaktır.

Bunu için öncelikle ilgi alanının ne olduğunun kararı verilmesi gerekmektedir. İlgi alanının ne olduğunun kararı verildikten sonra ilgi alanının etkin bir şekilde tespit edilmesi gerekmektedir. Görüntü işleme problemini çözümüne uygun ilgi alanı algoritması farklı olabilir ya da aynı işlemi yapan farklı algoritmalar kullanılabilir. Önemli olan kullanılan algoritmanın etkin bir şekilde çalışarak ilgi alanını başarılı bir şekilde bulmasıdır. Bu tez çalışmasında hareketli nesnelerin tespiti, takibi, tanımlaması ve sınıflandırılması yapılacağı için ilgi alanı harekettir. Hareket edebilen canlı, cansız, endüktif, kapasitif her türlü nesne ya da nesnelerin tespiti tez çalışmasının ilgi alanıdır. Ardışık sayısal görüntülerdeki hareket eden nesneleri, hareket etmeyen nesnelerden ve arka plandan ayrıştırıp hareket eden nesnelere ait görüntüleri ana görüntüden küçük görüntüleri elde etmek bunların takibini tanımlamasını ve sınıflandırmasını yapmaktır. Bu nedenle ardışık sayısal görüntülerde ilgi alanı olan hareketi tespit etmek için hareket tespiti algoritmalar incelenmiştir. NTİTS yöntemi hareket tespitinin yapıldıktan sonra tespit edilen hareketli nesnenin takibi tanımlanması ve sınıflandırması yapılır. Bu işlemler geliştirilen NTİTS yazılımında iki aşamada gerçekleştirilir.

Hareket tespiti için yapıldığı birinci aşamada, ardışık sayısal görüntülerdeki hareketli nesnelere karşılık gelen piksellerdeki ışıklılık değerlerinde ki değişimler OA algoritması ile tespit edilmiştir. Piksellerin ışıklılık değerlerindeki değişimlerin olduğu pikseller ayrıştırılarak gruplandırılmıştır. Ayrıştırılan bu piksel gruplarından hareket eden nesnelere ait görüntüler elde edilmiştir. Piksellerdeki ışıklılık değerlerindeki değişimlerin tespit edilebilmesi hareket tespiti için önemli bir husustur. Hareket tespiti yapılırken hareketli nesnelerin boyutlarında da değişimler olabilmektedir. Hareket tespiti bu değişimlere de uyarlanabilir olması gerekmektedir. OA algoritmasının hareketli nesnelerin tespiti için uygun olduğu yapılan çalışmalarda görülmüştür. OA yöntemi sayısal görüntülerdeki hareketin analizinde üç boyutlu (3B) gerçek dünyadaki hareketin, 3B'li gerçek dünyadaki ortamın görünür görüntüsünün bulunduğu sahnenin, görüntüsünün 2B'li ortamdaki izdüşümünden yararlanarak, görüntünün anlık hız bilgisini tüm piksellerin ışıklılık değerlerini kullanarak çıkaran hızlarını vektörel olarak ifade eder. OA metodu günümüzde bilgisayarla görme uygulamaları, yapay zekâ, çarpışmadan kaçınma ve robotik gibi alanlarda sıklıkla kullanılmaktadır.

İkinci aşamada, birinci aşamada, elde edilen hareketli nesnelere ait görüntülerin tanımlanması ve sınıflandırılması işlemi yapılmıştır. Tanımlama ve sınıflandırma işleminde hareketli nesneye ait görüntü elde edildikten sonra, görüntüye ait tüm özellikler YZ'nin bir alt dalı olan DSA'lar ile analiz edilerek tanımlama ve sınıflandırma işlemi yapılmıştır. DÖ büyük verileri tek bir katmanda yerine pek çok katmanda hesapları tek bir seferde yapabilen hatta tanımlanması gereken parametreleri bile kendisi keşfedebilen, belki de daha iyi parametreler ile değerlendirmelerde bulunabilen bir sistemdir. Analiz edilecek sayısal görüntüleri alıp aynı anda yüzlerce hatta binlerce GPU çekirdeğini paralel çalıştırarak görüntüye ait öz niteliklerini çıkararak bu öz nitelikleri daha önceden eğitilmiş veri seti ile karşılaştırıp nesnenin analizini yapan sistemdir. DSA olarak ESA sınıfında olan TAÇKA kullanılmıştır. “TAÇKA, 300×300 giriş görüntüsü için bir Nvidia Titan X ekran kartı ile 58 FPS'de VOC2007 testinde % 72,1 mAP elde eder ve 500×500 giriş için SSD,% 75,1 mAP elde eder, Faster R-CNN modeli ile karşılaştırıldığında daha iyi performans göstermiştir[122]”

6. SONUÇ VE ÖNERİLER

Bu bölümde tez çalışması sonucunda geliştirilen NTİTS yönteminden elde edilen sonuçlar gösterilmiştir ve bundan sonraki yapılacak olan çalışmalarda nerelerde kullanılabileceğine dair önerilerde bulunulmuştur.

6.1. Sonuçlar

Bu tezde, Ardışık Sayısal Görüntülerde Çok Sayıda Hareketli Nesnenin Gerçek Zamanlı Tespiti, İzlenmesi, Tanımlanması ve Sınıflandırılması yöntemi geliştirilmiştir. Geliştirilen NTİTS yöntemi ile değişik çözünürlüklü, ardışık sayısal görüntülerde çok sayıda hareketli nesne, hareketsiz nesnelerden ve arka zeminden ayrıştırılarak çok sayıdaki hareketli nesnenin gerçek zamanlı olarak tespiti, izlenmesi, tanımlanması ve sınıflandırılmasını yapmıştır. Tez çalışmasının amacı kapsamında yapılan çalışmalar ve geliştirilen NTİTS yöntemi bilgisayarla görü konusu içindedir.

6.2. Gelecekte Yapılabilecek Uygulamalara Dair Öneriler

Geliştirilen NTİTS yönteminin birinci aşamasında ilgi alanı olan hareketli nesneleri tespit edip, ilgi alanındaki hareketli nesneleri oluşturan pikselleri ayrıştırarak hareketli nesnelere ait görüntüler elde etmesi ve bu görüntülerin ikinci aşamada derin öğrenme algoritmaları ile hareketli nesnelere ait birinci aşamada elde edilen görüntülerin analizi yapılmıştır. Bu sayede gereksiz piksel yığınları ile çalışmaktan kurtararak gerçek zamanlı görüntü işlemenin yapılabilmesini sağlamıştır. Birinci aşamadaki ilgi alanı algoritmasının kullanılması ve bu algorithmadan elde edilen sonuçlar ile çalışılması hem sistem gereksinimlerini azaltmaktadır hem de kurulum maliyetlerini azaltmaktadır. Bu nedenle tez çalışması kapsamında geliştirilen NTİTS yönteminin iki aşamalı yapısından elde edilen deneyimler görüntü işleme konusunda araştırma yapanlara yeni bir pencere açarak farklı uygulamalarda esin kaynağı olabilir. NTİTS yönteminin birinci ve ikinci aşamasında farklı görüntü işleme teknikleri kullanılabilir. Geliştirilen NTİTS yazılımında farklı tekniklerin kullanılması ve sisteme dahil edilmesi mümkündür. Farklı görüntü işleme tekniklerinin eklenmesi ve kullanılması tez çalışması kapsamında yapılan çalışmayı çok daha önemli kılacaktır.

Geliştirilen NTİTS yöntemi ile sürekli gözetlenmesi gereken stratejik önemli bölgelerin (Ülkelerin Başkanlık tesisleri, Ülkelerin Bakanlıkları, askeri tesisler, cephanelikler, sınır hatları, hava üsleri, hava alanları, limanlar, silolar, çiftlikler, elektrik santralleri, petrol rafinerileri, fabrikalar vb. gibi stratejik öneme sahip binalar ve tesisler) gözetlenmesinde nöbetçi (ler) yerine, nöbetçilerin olduğu noktalara konuşlandırılmış sabit veya hareketli her türlü sayısal kamera (lar)dan alınan çok yüksek çözünürlüklü ardışık sayısal görüntüleri gerçek zamanlı olarak analiz edilerek kritik önemli bölgede hareketli tüm nesnelerin otomatik olarak tespit edilmesi problemine çözüm önermiştir. Bu nesnelerin tanımlaması ve sınıflandırılması yapılırken YZ'nin bir alt dalı olan DÖA'ları kullanılmıştır. DÖA'ları birinci aşamada tespit edilen hareketli nesneleri birçok katmanda kullanılan hesapları tek bir seferde yapan, tanımlanması gereken parametreleri bile kendisi keşfeden, belki de daha iyi parametreler ile değerlendirmeler yaparak nesneleri tanımlamasını ve sınıflandırmasını yapar. Tespit edilip tanımlaması ve sınıflandırılması yapılan bu nesneler içinden istenmeyen zararlı unsurlar otomatik olarak kolaylıkla seçip ayrıştırılabilir. Stratejik önemli noktaya gözetleme için yerleştirilen nöbetçi (ler) de zaman ilerledikçe dikkati dağılır, dalgınlık, yorgunluk, bıkkınlık ve uyku gibi sorunlarla da mücadele etmeye başlar. Bir süre sonra nöbetçi bu sorunlardan yorulup yenik düşebilir. Stratejik öneme sahip noktalara sızmak isteyen zararlı unsurlar, gözetleme noktalarına yerleştirilmiş nöbetçilerin bu durumlarından faydalanarak saldırabilirler. Bu buluşla bu tür istenmeyen problemlerin çözümüne katkısı olacağı düşünülmektedir.

Dron teknolojisindeki gelişmelerle yüksek çözünürlüklü kamera teknolojisindeki gelişmelere paralel olarak kullanımlarının yaygınlaşması ve ucuzlaması sonucunda kolayca elde edilebilir olmuş ve yaygın kullanım alanı bulmuştur. Bu durum stratejik öneme sahip binaları ve tesisleri teröristlerin dronlarla saldırı tehditlerine açık hale getirmiştir.

Örneğin, dronlar patlayıcı maddelerle donatılıp, uzaktan kumanda ile patlatılarak terörist eylemlerde kullanılabilir. Yüksek katılımlı önemli etkinliklerde kimyasal gaz kullanıp toplu olarak çok sayıda kişiye zarar verebilir. Çok sayıda dronlar, mini ve mikro ihalarla patlayıcı taşıyıp büyük önemli tesislere örneğin, nükleer santraller, hidroelektrik santralleri gibi tesislere zarar verebilirler. Ayrıca dronların ucuzlamasıyla kişisel olarak hobi amaçlı kullanımları da yaygınlaşmıştır. Ancak hobi amaçlı kullanımlarda, şahısların kontrolsüz bir şekilde bunları kullanması hava alanı gibi bölgelerde uçuş güvenliğini tehdit etmektedir. Uçak motoruna çarpan bir dron ne yazık ki telafisi mümkün olmayan can ve mal kayıplarına

sebepe olabilmektedir. Tez kapsamında yapılan çalışmalarda yapa zekâ öğrenmesinin alt dalı olan derin öğrenme algoritmaları ile dronları, mini ve mikro ihaları, uçabilen diğere nesneleri, kuş sürülerini ayırt edilip tanımlaması ve sınıflandırmasını gerçek zamanlı olarak yapabilmesi sayesinde böyle problemlerin çözümüne katkısı olacağı düşünülmektedir.

Bir üretim tesisinde, üretilen ürünlerin üretim sürecinde üretim devam ederken, ürünlerin şekli uygunluklarının kontrol edilmesinde kullanılabilir. Üretim esnasında, üretilen ürünlerin şekli uygunluklarının kontrolünde kullanılabilir. Üretim aşamalarında üretilen ürünlerin şekli uygunluklarını denetleyerek kusurlu olan ürünler otomatik olarak tespit edilebilir. Standart dışı üretilen hatalı ürünleri seçerek kalite kontrol sürecinde kullanılabilir. Robotlarda insan algısına yakın doğruluk oranı ile gördüğünü yorumlama ve yoruma göre tepki yetisi kazandırabileceği için insanımsı robot teknolojilerinde kullanılabilir.



KAYNAKLAR

1. Castaño-Díez, D., Moser, D., Schoenegger, A., Pruggnaller, S., and Frangakis, A. S. (2008). Performance evaluation of image processing algorithms on the GPU. *Journal of Structural Biology*, 164(1), 153-160
2. Kayran, A. H. ve Ekşioğlu, E. M. (2004). *Bilgisayar uygulamalarıyla sayısal işaret işleme*. (Birinci Baskı). İstanbul: Birsen Yayınevi.
3. Gonzalez, R. C., and Woods, R. E. (2001). *Digital image processing*, Addison-Wesley: Longman Publishing, 1-249.
4. Bozkurt, K. (2009). *Retina görüntülerinin ayrıştırılması*. Yüksek Lisans Tezi, Düzce Üniversitesi, Fen Bilimleri Enstitüsü, Düzce, 1-39.
5. Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2004). *Digital Image Processing Using Matlab*. (First Edition). Upper Saddle River, NJ : Pearson/Prentice Hall. 1-56.
6. da Silva, E. A. B., and Mendonça, G. V. (2005). *Digital image processing*. in *The Electrical Engineering Handbook*, Wai-Kai Chen (Editor). Elsevier - Academic Press, 891-910.
7. Fitzgerald, A. E. (1945). *Basic Electrical Engineering: Circuits, Machines, Electronics*: (First edition). USA:McGraw-Hill, 145
8. İnternet: About Digital Cinema Initiatives, LLC. (2020). Web: <https://www.dcinovies.com/>.15 Mart 2020'de alınmıştır.
9. Zhang, A.B. and Gourley, D. (2009). *Creating Digital Collections A pratical guide*, A.B. Zhang and D. Gourley, (Editors). (Second edition). Oxford:Chandos. 55-72.
10. McFarlane, M.D. (1972). *Digital pictures fifty years ago*. Proceedings of the IEEE. 60(7), 768-770.
11. Kobayashi, K. (2002). Birth of a Digital Phototelegraph-the Bartlane System. *Journal of the Institute of Image Electronics Engineers of Japan*, 31, 244-249.
12. Shinde, B. (2011). The Origins of Digital Image Processing & Application areas in Digital Image Processing Medical Images. *IOSR Journal of Engineering*, 1, 66-71.
13. Tyagi, V. (2018). *Understanding Digital Image Processing*. (First edition). ebook, 12-79.
14. Kent, A. and Williams, J.G. (1987). *Encyclopedia of Computer Science and Technology*. Volume 18 - Supplement 3, NewYork and Basel: Marcel Dekker, 20-386
15. İnternet: Nasa. (2020). *Making New Reality: Computers in Simulations and Image Processing*. Computers in Spaceflight: The NASA Experience 2020. Web: <https://history.nasa.gov/computers/Ch9-3.html> 18 Kasım 2020'de alınmıştır.
16. Tabassum, S.N. (2012) *Digital Computing Image Processing Network Technology*. International Journal of Scientific and Engineering Research, 3(11), 909-913
17. İnternet: Uri, J. (2019). 55 years ago: ranger 7 photographs the moon. *NASA History*, Web: <https://www.nasa.gov/feature/55-years-ago-ranger-7-photographs-the-moon> 10 Ekim 2020'de alınmıştır.

18. İnternet: Cormack, A. M. and Hounsfield, G. N. (1979). The Nobel Prize in Physiology or Medicine 1979. *Nobel Fizyoloji veya Tıp Ödülü*. Web: <https://www.nobelprize.org/prizes/medicine/1979/summary/> 20 Ekim 2020'de alınmıştır.
19. Zhou, S.K., D. Rueckert, and Fichtinger, G. (2020). *Handbook of Medical Image Computing and Computer Assisted Intervention*. Academic Press: Mara Conner, 1-24.
20. Dougherty, G. (2009). *Digital image processing for medical applications*. (First edition). United States of America: Cambridge University Press, 1-405
21. Bora, D. J. (2019). Big data analytics in healthcare: a critical analysis. *Big Data Analytics for Intelligent Healthcare Management*, 43-57
22. Risha, K.P. and Chempak, K.A. (2016). Novel Method of Detecting Moving Object in Video. *Procedia Technology*, 24, 1055-1060.
23. Wu, Y. (2019). *Optical Flow and Motion Analysis*, EECS432-Advanced Computer Vision Notes Series 6. (First edition). Evaston:Northwestern University, 1-12.
24. Kulchandani, J.S. and Dangarwala, K.J. (2015, January). Moving object detection: Review of recent research trends. *2015 International Conference on Pervasive Computing (ICPC)*. Pune, India, 1-5.
25. Guo, J., Wang, R.B., Zhang, Y. and Li, Y. (2017, January). A new moving object detection method based on frame-difference and background subtraction. *IOP Conference Series: Materials Science and Engineering, 2017 3rd International Conference on Applied Materials and Manufacturing Technology (ICAMMT 2017)*. Changsha, China. 242(1), 1-4.
26. Temizel, A., Halıcı, T., Loğlu B. and Taşkaya, Temizel T. (2011). *Chapter: 34 Experiences on Image and Video Processing with CUDA and OpenCL*, in *GPU Computing Gems Emerald*. (First edition) Burlington: Elsevier, (Editor: Hwu, W.-m.W.), 547-567.
27. Verma, R. (2017). A review of object detection and tracking methods. *International Journal of Advance Engineering and Research Development*, 4(10), 569-578.
28. Prajapati, D. and Galiyawala, H. (2015). A Review on Moving Object Detection and Tracking, *International Journal of Computer Application*, 5(3), 168-165.
29. Horn, B. K. P. and Schunck, B. G. (1981). Determining Optical Flow. *Artificial Intelligence*, 17(1-3), 185-203.
30. Amandeep and Goyal, E.M. (2015). Review : moving object detection techniques. *International Journal of Computer Science and Mobile Computing*, 4(9), 345-349.
31. Gibson, J. J. (1950). *The perception of the visual world*. (First edition). USA:Houghton Mifflin, 1-127
32. Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994a). Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1), 43-77.
33. Özüntürk, E. (2007). *Optik akış ile hareket tespiti*, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, 9-24.
34. Dur, E., Temeltaş, H. ve Kurnaz, S. (2009). Optik akışın hesaplanması ve yapay sinir ağları ile yorumlanarak mobil robotlar için engel tespiti ve kaçınma davranışında kullanılması. *Havacılık ve Uzay Teknolojileri Dergisi*, 4, 77-87.

35. İnternet: Florian, R. (2013). Optic flow. *Scholarpedia*, 8(7), 30724. doi:10.4249/scholarpedia.30724 Web: http://www.scholarpedia.org/article/Optic_flow adresinden 3 Mart 2019'da alınmıştır.
36. Hildreth, E. and Ullman, S. (1982). The measurement of visual motion. *Massachusetts Institute of Technology Artificial Intelligence Laboratory*, 699, 1-22.
37. İnternet: Narasimha, S., (2006). Computer vision, in computer vision, *Lecture Notes*. Web: <https://studylib.net/doc/16054810/computer-vision-lecture-%2319-spring-2006-15-385--685-instr...> adresinden 15 Ocak 2020'de alınmıştır.
38. Barron, J.L., Fleet, D.J. and Beauchemin, S.S. (1994b). Performance of optical flow techniques, *International Journal of Computer Vision*, 12(1), 43-77.
39. İnternet: Türk Dil Kurumu Sözlükleri. (2021). Öğrenmek, *Bilim ve Sanat Terimleri Sözlüğü*, Web: <https://sozluk.gov.tr/> 16 Haziran 2021'de alınmıştır.
40. İnternet: Türk Dil Kurumu Sözlükleri. (2021). Öğrenme, *Bilim ve Sanat Terimleri Sözlüğü*, Web: <https://sozluk.gov.tr/> 16 Haziran 2021'de alınmıştır.
41. Elmas, Ç. (2018) *Yapay Zeka Uygulamaları*. (Dördüncü baskı). Ankara:Seçkin, 25-250.
42. İnternet: Türk Dil Kurumu Sözlükleri. (2021). Zeka, *Bilim ve Sanat Terimleri Sözlüğü*, Web: <https://sozluk.gov.tr/> 16 Haziran 2021'de alınmıştır.
43. Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. (First edition). USA:Prentice Hall, 23.
44. Zurada, J.M. (1992). *Introduction to Artificial Neural Systems*. (First edition). St. Paul, New York, Los Angeles, San Francisco:West Group, 16-368
45. McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5, 115-133.
46. Hebb, D.O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. (First edition). New York:Wiley and Sons, 11-198.
47. Sze, V., Chen Y., Yang, T. and Emer, J. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105, 2295-2329.
48. McCulloch, W.S. and W. Pitts, (1988, January). *A logical calculus of the ideas immanent in nervous activity*, in *Neurocomputing: foundations of research*, A.A. James and R. Edward (Editors). USA:MIT Press, 15-27.
49. Turing, A.M. (1947). Rounding-off errors in matrix processes, *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1), 287-308.
50. Turing, A. M. (1950). I.-Computing machinery and intelligence, *Mind a Quarterly Review of Psychology and Philosophy*, 59(236), 433-460.
51. İnternet: Fogg, A. (2018). A History of Deep Learning; *import.io*, Web: <https://www.import.io/post/history-of-deep-learning/> 26 Mart 2020'de alınmıştır.
52. Rosenblatt, F. (1957). The Perceptron: A Perceiving and Recognizing Automaton. NY: Cornell Aeronautical Laboratory. Report 85-460-1, 1-24.
53. Rosenblatt, F., (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386-408.
54. Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3), 574-591.

55. Kelley, H.J. (1960). Gradient Theory of Optimal Flight Paths, *ARS Journal*, 30(10), 947-954.
56. Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(4), 364-378.
57. Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1), 215-243.
58. Fukushima, K. (1980). Neocognitron A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics*, 36(4), 193-202.
59. Tombe, R. and Viriri, S. (2020). Effective processing of convolutional neural networks for computer vision: a tutorial and survey. *IETE Technical Review*, 1-14.
60. Fukushima, K. and Miyake, S. (1982). Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. In: Amari S., Arbib M.A. (Editors). *Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics*, Berlin, Heidelberg. 45, 267-285.
61. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *PNAS Proceedings of the National Academy of Sciences of the United States of America*, 79(8), 2554-2558.
62. Dechter, R. (1986, August). Learning while searching in constraint-satisfaction-problems, *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI Press, 178-183.
63. Rumelhart, D., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating errors, *Nature*, 323, 533-536.
64. Sejnowski, T., and Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1(1), 145-168.
65. Watkins, C.J.C.H. (1989) *Learning from Delayed Rewards*,. Doctoral dissertation King's College, Cambridge, UK
66. Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990, January). Handwritten digit recognition with a back-propagation network. D. Touretzky (Editors). *Advances in Neural Information Processing Systems (NIPS 1989)*, Denver, 2, 396-403
67. Schmidhuber, J. (1993). *Network architectures, objective functions, and chain rule*, in *nstitut für Informatik*, Technische Universität München: Technische Universität München , 1-186
68. Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 2673-2681.
69. Lecun, Y. and Bottou, L. and Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, 86(11), 2278-2324.
70. Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012, December). Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, New York, USA. 1, 1097-1105.

71. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014, June). Generative adversarial networks. Ghahramani, Z., Welling, M., Cortes, C., Lawrence N. and Weinberger, K.Q. (Editors). *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA. 2, 2672-2680.
72. İnternet: MissingLink.ai. (2020). *Convolutional Neural Network Tutorial: From Basic to Advanced*, Web: <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/> 10 Ekim 2020'de alınmıştır.
73. Yamashita, R., Nishio, M., Do, R.K.G. and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), 611-629.
74. Fu Hongping, Niu Z., Zhang C. and Chen J. (2016). Visual cortex inspired cnn model for feature construction in text analysis. *Frontiers in Computational Neuroscience*, 10, 1-10.
75. Lindsay, G.W., (2020). Convolutional neural networks as a model of the visual system: past, present, and future. *Journal of Cognitive Neuroscience*, 1-15.
76. Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., and Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLOS Computational Biology*, 15(4), 1-16.
77. Hubel, D.H. and Wiesel, T.N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160, 106-154.
78. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L.D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4), 541-551.
79. Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012, December). Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NewYork, USA. 1, 1097-1105.
80. Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *Computer Vision and Patern Recognition arXiv.oerg*, 1-14.
81. İnternet: Saha, S. (2018). A comprehensive guide to convolutional neural networks - the eli5 way, Web: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> 29 Eylül 2020'de alınmıştır.
82. Nixon, M.S. and Aguado, A.S. (2012). *Feature extraction and image processing for computer vision*, (Third Edition), Academic Press:Oxford, 83-136.
83. Ramachandran, P., Zoph, B. and Le, Q.V. (2017). Swish: a self-gated activation function, *arXiv: Neural and Evolutionary Computing*, 1-12, arXiv:1710.05941v1
84. Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: comparison of trends in practice and research for deep learning. *Machine Learning ArXiv*, 1-17

85. Nair, V., and Hinton, G.E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of ICML*, 27, 807-814.
86. İnternet: Karakuş, B.A. (2018). Derin sinir ağları için aktivasyon fonksiyonları, *Büyük Veri ve Yapay Zeka Laboratuvarı*, Web: <http://buyukveri.firat.edu.tr/2018/04/17/derin-sinir-aglari-icin-aktivasyon-fonksiyonlari/> 10 Haziran 2020'de alınmıştır.
87. He, K., Zhang, X., Ren, S., and Sun, J. (2015, December). Delving deep into rectifiers: surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, USA, 1026-1034.
88. Sharma, O. (2020). A novel activation function in convolutional neural network for image classification in deep learning. *Data Science and Analytics*, 1229, 120-130.
89. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 15(1), 1929-1958.
90. Carass, A., Roy, S., Gherman, A., Reinhold, J. C., Jesson, A., Arbel, T., Maier, O., Handels, H., Ghafoorian, M., Platel, B., Birenbaum, A., Greenspan, H., Pham, D. L., Crainiceanu, C. M., Calabresi, P. A., Prince, J. L., Roncal, W., Shinohara, R. T., and Oguz, I. (2020). Evaluating white matter lesion segmentations with refined sørensens-dice analysis. *Scientific Reports*, 10(1), 8242.
91. Jaccard, P. (1912). The Distribution of the Flora of the Alpine Zone. *New Phytologist*, 11(2), 37-50.
92. Kosub, S. (2019). A note on the triangle inequality for the Jaccard distance. *Pattern Recognition Letters*, *arXiv*, 120, 36-38.
93. İnternet: ImageNet, (2021). About imagenet, *Stanford Vision Lab, Stanford University, Princeton University*, Web: <http://image-net.org/about-overview> adresinden 02.03.2021'de alınmıştır.
94. Zeiler, M.D. and Rergus F, (2014, September). Visualizing and understanding convolutional neural networks, *ECCV: European Conference on Computer Vision Computer Vision-ECCV 2014*, Zurich, Switzerland, 8689, 818-833.
95. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2014, September). Going deeper with convolutions, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 1-9.
96. He, K., Zhang, X., Ren, S., and Sun, J. (2016, June). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 770-778. Doi:10.1109/cvpr.2016.90
97. Romano, F. (2015). *Learning Python*. (First edition). Birmingham: Packt Publishing, 2-68
98. İnternet: Python Software Foundation. (2020). *General Python FAQ*. Web: <https://docs.python.org/3/faq/general.html> adresinden 26 Temmuz 2020'de alınmıştır.
99. İnternet: Albatross. (2020). History of C++. *cplusplus.com*. Web: www.cplusplus.com/info/history/ adresinden 25 Temmuz 2020'de alınmıştır.
100. Stroustrup, B. (1993). A history of C++: 1979–1991, *ACM SIGPLAN Notices*. 28(3), 271-297.

101. Schildt, H., (2021). *Java The Complete Reference*, (Eight Edition). New York: Oracle Press, 3-17.
102. İnternet: Oracle, (2020). Java teknolojisi hakkında bilgi edinin. Web: <https://java.com/tr/about/> adresinden 26 Temmuz 2020'de alınmıştır.
103. Arslan, İ. (2019). *R ile İstatistiksel Programlama*. (Üçüncü Baskı). İstanbul:Pusula, 1-30
104. Rungta, K. (2019). *Learn R Programming in 1 Day: Complete Guide for Beginners*. Independently published, 1-43
105. Ierusalimschy, R. (2016). *Programming in Lua, (Fourth Edition)*. Lua.Org, 1-48
106. İnternet: Lua.org. (2020). Lua About, web: <http://www.lua.org/about.html> adresinden 23 Temmuz 2020'de alınmıştır.
107. Ierusalimschy, R., de Figueiredo, L.H. and Celes, W. (2007, June). The evolution of Lua, *HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*, Association for Computing Machinery, NewYork, NY, USA, 1-26.
108. Kamiński, B. and Szufel, P. (2018). *Julia 1.0 Programming cookbook, over 100 numerical and distributed computing recipes for your daily data science workflow*, (First edition), Birmingham:Packt Publishing, 7-401
109. Salceanu, A. (2018). *Julia programming projects: learn julia 1.x by building apps for data analysis, visualization, machine learning, and the web*. (First edition). Birmingham:Packt Publishing, 6-38.
110. Jia, Y., Shelhamer, E., Donahue, J. Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T. (2014, November). Caffe: convolutional architecture for fast feature embedding. *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, New York, NY, USA, 1-4.
111. İnternet: Brownlee, J. (2016). *Popular Deep Learning Libraries*, Web: <https://machinelearningmastery.com/popular-deep-learning-libraries/>.adresinde 8 Mayıs 2020'de alınmıştır.
112. İnternet: Keras. (2020). *Keras Documentation*, Web: <https://keras.io/> adresinden 5 Mayıs 2021'de alınmıştır.
113. Chollet, F. (2017). *Deep learning with Python*. (First edition), New York:Manning Publications, 6-241
114. Bastien, F., Lamblin, P., Pascanu, R. Bergstra, J. Goodfellow, I., Bergeron, A., Bouchard, N. Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. Presented at the Deep Learning Workshop, *arXiv*, 1-10.

115. Theano Development Team: Al-Rfou, R., Alain, G., Almahairi, A., Angermüller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J.B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., Brébisson, A., Breuleux, O., Carrier, P.L., Cho, K., Chorowski, J., Christiano, P.F., Cooijmans, T., Côté, M.A., Côté, M., Courville, A.C., Dauphin, Y.N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Kahou, S.E., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I.J., Graham, M., Gülçehre, Ç., Hamel, P., Harlouchet, I., Heng, J.P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrançois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J.A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.A., Mastropietro, O., McGibbon, R., Memisevic, R., Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C.J., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I.V., Serdyuk, D., Shabanian, S., Simon, É., Spieckermann, S., Subramanyam, S.R., Sygnowski, J., Tanguay, J., Tulder, G., Turian, J.P., Urban, S., Vincent, P., Visin, F., Vries, H., Warde-Farley, D., Webb, D.J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv*, 1-19,
116. İnternet: Ronan, Clément, Koray and Soumith. (2019). What is Torch?, *A Scientific Computing Framework For Luajit Get Started* .Web: <http://torch.ch/> 8 Mayıs 2019'da alınmıştır.
117. Collobert, R., Kavukcuoglu, K. and Farabet, C. (2011). Torch7: A Matlab-like Environment for Machine Learning. *NIPS 2011*, 1-6
118. Collobert, R., Bengio, S. and Marthoz, J. (2002). Torch: A Modular Machine Learning Software Library. *IDIAP Research Report*, 02(46). 2-6.
119. İnternet: NVIDIA. (2019). *NVIDIA DIGITS Interactive Deep Learning GPU Training*, Web: <https://developer.nvidia.com/digits> 8 Mayıs 2019'da alınmıştır.
120. İnternet: Deeplearning4j. (2020). *Deeplearning4j Tutorials*. Web: <https://deeplearning4j.konduit.ai/getting-started/tutorials> adresinden 26 Temmuz 2020'de alınmıştır.
121. İnternet: Eraly, S., Rahman, F., Azeem, S.U., Altena, R., Gonzalez E., Stoyakin A., Shyrma Y., Shulinok G., Dubs P., Semeniv, O., Gurbanov, R. and Tuzhykov, A. (2021). *Eclipse Deeplearning4j*. Web: <https://projects.eclipse.org/projects/technology.deeplearning4j> adresinden 2 Şubat 2021'de alınmıştır.
122. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., and Berg, A. (2016). SSD: single shot multibox detector. *ECCV arXiv*., 21-37.
123. Alali, M.I., Mhaidat, K., and Aljarrah, I. (2013, December). Implementing image processing algorithms in FPGA hardware. *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Amman, Jordan, 1-5.
124. İnternet: (2020). Biyomimikri nedir? *Biyomimikri Doğadan Gelen İnovasyon*, Web: <http://www.biyomimikri.com/tanim> adresinden 20 Temmuz 2020'de alınmıştır.
125. Nkandu, M.I. and Alibaba, H.Z. (2018). Biomimicry as an alternative approach to sustainability, *Architecture Research*, 8(1), 1-11.

126. Hargroves, K.J. and Smith, M.H. (2006). Innovation inspired by nature: Biomimicry, *ECOS*, 129, 27-29.
127. Aziz, M.S. and El sherif, A.Y. (2016). Biomimicry as an approach for bio-inspired structure with the aid of computation, *Alexandria Engineering Journal*, 55(1), 707-714.
128. El-Zeiny, R.M.A. (2012). Biomimicry as a problem solving methodology in interior architecture. *Procedia - Social and Behavioral Sciences*, 50, 502-512.
129. Benyus, J.M. (2009). *Biomimicry: innovation inspired by nature*. HarperCollins e-books, 1-280
130. Vincent, J.F.V., Bogatyreva, O.A., Bogatyrev, N.R., Bowyer, A., and Pahl, A.K. (2006). Biomimetics: its practice and theory, *Journal of the Royal Society Interface*, 3(9), 471-482.
131. Oguntona, O.A. and Aigbavboa, C.O. (2017). Biomimicry principles as evaluation criteria of sustainability in the construction industry. *Energy Procedia*, 142, 2491-2497.
132. Gardner, G. (2012). Using Biomimicry to Engage Students in a Design-Based Learning Activity. *The American Biology Teacher*. 74(3), 182-184.
133. Kumar, A. (2015). Learning through Biomimicry, *National Conference on Inspired Learning*, Delhi, India, 153-156.
134. Ashokan, V.S. (2014). *Mimicking nature: A solution for sustainable development*. (First edition), India: Partridge, 1-250
135. Singh A., and Nayyar, N. (2015). Biomimicry-an alternative solution to sustainable buildings. *Journal of Civil Engineering and Environmental Technology*, 2(14), 96-101.
136. Gaissert, N., Mugrauer, R., Mugrauer, G., Jebens, A., Jebens, K., and Knubben E.M. (2014). Inventing a micro aerial vehicle inspired by the mechanics of dragonfly flight. *Springer Berlin Heidelberg*, 90-100.
137. Usherwood, J.R. and Lehmann, F.O. (2008). Phasing of dragonfly wings can improve aerodynamic efficiency by removing swirl. *Journal of the Royal Society, Interface*, 5(28), 1303-1307.
138. Ren, L. and Li, X. (2013). Functional characteristics of dragonfly wings and its bionic investigation progress. *Science China Technological Sciences*, 56(4), 884-897.
139. Lestari, D. (2020). Biomimicry learning as inspiration for product design innovation in industrial revolution 4.0., *International Journal of Creative and Arts Studies*, 7(1), 1-18.
140. El-Naggar, A. and Shalaby, A. (2015, December). Biomimicry to network on chip: router heart rate. *2015 27th International Conference on Microelectronics (ICM)*, Casablanca, Morocco, 162-165.
141. Fayemi, P., Maranzana, N., Aoussat A. and Bersano, G. (May, 2014). Bio-inspired design characterisation and its links with problem solving tools, *DS 77: Proceedings of the DESIGN 2014 13th International Design Conference*, Dubrovnik - Croatia, 173-182.
142. Cıgızoğlu, M. (2011). Biyomimetik tepkisel yüzey örtüsü tasarımı için bir model önerisi: hexa-myosis, (Yüksek Lisans Tezi), *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul. 9.

143. Minsolmaz, Yeler, G. and Yeler, S. (2017). Models from nature for innovative building skins. *Kırklareli Üniversitesi Mühendislik ve Fen Bilimleri Dergisi*, 3(2), 142-165.
144. Panero, J. and Zelnik, M. (1979). *Human dimension & interior space: a source book of design reference standards*. (First Edition). NewYork:Whitney Library of Design, 265-297
145. Zhisheng, W., Yukari N., Dan, Z., Jiahui, L., and Nianyu, Z. (2019, August). Based on creative thinking to museum lighting design influences to visitors emotional response levels theory research, *7th International Forum on Industrial Design IOP Conference Series: Materials Science and Engineering*, Orlando, Florida, 1-7.
146. Tara, A., Lawson, G. and Renata, A. (2021). Measuring magnitude of change by high-rise buildings in visual amenity conflicts in brisbane, *Landscape and Urban Planning*, 205, 1-12.
147. Lungaro, P., Sjoberg, R., Valero, A.J.F., Mittal, A., and Tollmar, K. (2018). Gaze-aware streaming solutions for the next generation of mobile vr experiences, *IEEE Transactions on Visualization and Computer Graphics*, 24(4), 1535-1544.
148. Bek, R. (2018). *The perceptual and psychological effects of artificial lighting on peripheral vision in humans*, (Master Dissertation), KTH, School of Architecture and the Built Environment (ABE), 5.
149. Viola, P. and Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 511-518.
150. İnternet: Forson, E. (2017). Understanding ssd multibox - real-time object detection in deep learning, *Towards Data Science*, Web: <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab> adresinden 04 Mart 2021'de alınmıştır.
151. İnternet: Yeren, A. (2021). *Ubuntu nedir?* Web: <https://www.mediatick.com.tr/tr/blog/ubuntu-nedir> adresinden 8 Mayıs 2021'de alınmıştır.
152. İnternet: Python Yazılım Vakfı. (2020). Python kurulumu, *Python*, Web: https://wiki.ubuntu-tr.net/index.php?title=Python_kurulumu adresinden 26 Eylül 2020'de alınmıştır.
153. İnternet: Yaz Bel Topluluğu. (2017). Python Hakkında. *Python Belgeleri*, Web: https://belgeler.yazbel.com/python/#_python_hakk adresinden 01 Şubat 2021'de alınmıştır.
154. Kaehler, A. and Bradski, G. (2016). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. (First edition), USA:O'reilly, 1-175
155. Pulli, K., Baksheev, A., Korniyakov, K., and Eruhimov, V. (2012). Realtime computer vision with opencv. *Queue*, 10(4), 40-56.
156. İnternet: OpenCV Team. (2020). *About OpenCV*. Web: <https://opencv.org/about/> adresinden 20 Temmuz 2020'de alınmıştır.
157. Vaughan, L., (2018). *Impractical Python Projects: Playful Programming Activities to Make You Smarter*. (First edition), San Francisco:William Pollock. 426.

158. İnternet: Stafford-Fraser, Q. (1955). *The Trojan Room Coffee Pot*. Web: <https://www.cl.cam.ac.uk/coffee/qsfc/coffee.html> adresinden 10 Mayıs 2020'de alınmıştır.
159. Bilici, N., Pehlivanlı, R. and Demirkılınç, S. (2017). Innovation and global issues in social sciences. *InGlobe Institution. extended abstracts book*, 1-156.
160. İnternet: History-Computer.com. (2020). Complete history of the computer mouse, *Computer mouse*, Web: <https://history-computer.com/ModernComputer/Basis/mouse.html> adresinden 19 temmuz 2020'de alınmıştır.
161. İnternet: Hall, M. (2021). Douglas Engelbart. *Encyclopedia Britannica*. web: <https://www.britannica.com/biography/Douglas-Engelbart> adresinden 3 Mart 2021'de alınmıştır.
162. Lancaster, D. (1976). *TV Typewriter Cookbook* (First edition). New York: Gernsback Publications, 5-126.





GAZİ GELECEKTİR..