EVRİŞİMLİ SİNİR AĞI ÖZELLİKLERİNE DAYANAN KORELASYON FİLTRELEME VE VERİ İLİŞKİLENDİRME İLE ÇOKLU NESNE TAKİBİ

Elnura ARSLAN

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EVRİŞİMLİ SİNİR AĞI ÖZELLİKLERİNE DAYANAN KORELASYON FİLTRELEME VE VERİ İLİŞKİLENDİRME İLE ÇOKLU NESNE TAKİBİ

Elnura ARSLAN
0000-0002-7999-0072

Dr. Öğr. Üyesi Ceyda Nur ÖZTÜRK
(Danışman)

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2021
Her Hakkı Saklıdır

ÖZET

Yüksek Lisans Tezi

EVRİŞİMLİ SİNİR AĞI ÖZELLİKLERİNE DAYANAN KORELASYON FİLTRELEME VE VERİ İLİŞKİLENDİRME İLE ÇOKLU NESNE TAKİBİ

Elnura ARSLAN

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Ceyda Nur ÖZTÜRK

Nesne takibi art arda görüntüler içerisinde nesne tespitine ihtiyaç duyulmaksızın nesne konumlarının belirlenmesini sağlar. Çekirdek korelasyon filtresi Fourier uzayında hızlı ve etkili nesne takibini gerçekleştirmektedir. Evrişimli sinir ağlarının ürettiği derin özelliklerin korelasyon filtresi ile kullanımı takip başarısını arttırmaktadır. Bu tez kapsamında üç farklı deney çalışması yapılmıştır. İlk çalışmada OTB-100 veri seti üzerinde farklı nesne takibi yöntemlerinin performans karşılaştırması gerçekleştirilmiştir. İkinci çalışmada çekirdek korelasyon filtresi için farklı görünüm modellerinin takip başarısına etkisi analiz edilmiştir. Üçüncü çalışmada ise 2D MOT 15 veri seti üzerinde yayaların tespit edilmesi, takip edilmesi, takip ve tespit edilen yayaların ilişkilendirilmesi denenmiştir. Yayaların tespit edilmesi için Yolo kullanılmıştır. Tespit edilen yayaların takip edilmesi için ise yönelimli eğimlerin histogramı (histogram of oriented gradients - HOG) tabanlı çekirdek korelasyon filtresi çalıştırılmıştır. Yayaların ilişkilendirilmesi için derin özellikler, renk histogramı ve çevreleyici kutulardan faydalanılmıştır. Yöntemlerin tekli nesne takip performansı başarı, kesinlik ve saniye başına düşen çerçeve ölçütleri ile çoklu nesne takip performansı ise tespitlerin hata oranı birleşimi ve kimlik değişim sayısı gibi ölçütler ile hesaplanmıştır. Sonuçlar göstermiştir ki çekirdek korelasyon filtresi derin özellikler ile birlikte kullanıldığında nesne takibi başarısı artmaktadır. Evrişimli sinir ağlarının daha çok son katmanlarından oluşturulan görünüm modelleri nesne takibini etkin kılmaktadır. Çekirdek korelasyon filtresi HOG özellikleri ile kullanıldığında gerçek zamanlı nesne takibi mümkün olmaktadır. Çoklu nesne takibinde derin özellikler ile yayaların ilişkilendirilmesi kimlik bilgilerinin değişim miktarını azaltmaktadır.

Anahtar Kelimeler: Çoklu Nesne Takibi, Çekirdek Korelasyon Filtresi, Derin Özellikler, Veri İlişkilendirme, Gerçek Zamanlı Takip
2021, ix + 97 sayfa

ABSTRACT

MSc Thesis

MULTIPLE OBJECT TRACKING WITH DATA ASSOCIATION AND CORRELATION FILTER BASED ON CONVOLUTIONAL NEURAL NETWORK FEATURES

Elnura ARSLAN

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronic Engineering

Supervisor: Assist. Prof. Ceyda Nur ÖZTÜRK

Object tracking allows determination of object positions in successive images without object detection. The kernelized correlation filter performs fast and efficient object tracking in Fourier space. Using deep features that are produced by convolutional neural networks with correlation filter increases tracking success. Three experimental studies were carried out in this thesis. In the first study, performance comparison of different object tracking methods was performed using OTB-100 dataset. In the second study, the effect of different appearance models on tracking success was analyzed for kernelized correlation filter. In the third study, detecting and tracking pedestrians, and associating the tracked and detected pedestrians using 2D MOT 15 dataset were tried. Yolo was used to detect pedestrians. In order to track the detected pedestrians, histogram of oriented gradients (HOG) based kernelized correlation filter was run. Deep features, color histogram, and bounding boxes were used to associate pedestrians. Single object tracking performances of the methods were computed with success, precision, and frames per second metrics, while multi-object tracking performances were computed with metrics such as a combined error rate of detections and number of identity changes. The results showed that the success of object tracking increases when the kernelized correlation filter is used with deep features. Appearance models, which are mostly retrieved from the last layers of convolutional neural networks, enable effective object tracking. Real-time object tracking is possible when the kernelized correlation filter is used with HOG features. Associating pedestrians with deep features in multi-object tracking reduces number of identity switches.

Key words: Multiple Object Tracking, Kernelized Correlation Filter, Deep Features, Data Association, Real-time Tracking
2021, ix + 97 sayfa

TEŞEKKÜR

Tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam Sayın Dr. Öğr. Üyesi Ceyda Nur ÖZTÜRK'e sonsuz teşekkürlerimi sunarım.

Tez çalışmam sırasında desteğini esirgemeyen Özdilek Ev Tekstil San. ve Tic. AŞ Özveri Ar-Ge Merkezi'ne çok teşekkür ederim.

Tez araştırmalarımda bana yardımcı olan, desteğini esirgemeyen ve manevi olarak sürekli yanımda olan Sayın İlyas Musaoğlu'na çok teşekkür ederim.

Tez çalışmam boyunca her zaman yanımda olan, maddi ve manevi desteklerini hiçbir zaman esirgemeyen ve hayatım boyunca beni her koşulda motive eden aileme en içten teşekkür ve sevgilerimi sunarım.

Elnura ARSLAN
28/07/2021

İÇİNDEKİLER

Sayfa

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
KISALTMALAR DİZİNİ vi
ŞEKİLLER DİZİNİ vii
ÇİZELGELER DİZİNİ ix
1. GİRİŞ 1
2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI 3
2.1. Nesne Tespitinde Kullanılan Klasik Yöntemler 3
2.2. Nesne Tespitinde Kullanılan Modern Yöntemler 3
2.3. Yapay Sinir Ağı 4
2.3.1. Derin Öğrenme 6
2.3.2. Evrişimli Sinir Ağları 7
2.4. Nesne Takibi Yöntemleri 12
2.4.1. İlkel Nesne Takip Yöntemleri 12
2.4.2. Karesel Hata Toplamının Minimum Çıktısı 14
2.4.3. Çekirdek Korelasyon Filtresi 14
2.4.4. Bayes Yöntemi 15
2.4.5. Görünüm Modelleri ile Takip 20
2.5. Veri İlişkilendirme 21
2.5.1. Doğrusal Atama 23
2.5.2. Kesişimin Birleşime Oranı 24
2.5.3. Macar Algoritması 24
2.5.4. Küresel En Yakın Komşu 28
2.5.5. Çizge Temelli Yaklaşımlar 29
2.5.6. Derin Özellikler ile Basit Çevrim İçi ve Gerçek Zamanlı Takip 32
2.6. Çoklu Nesne Takibi Kıyaslama 35
3. MATERYAL VE YÖNTEM 40
3.1. Veri Seti 40
3.2. Yazılım ve Donanım 42
3.3. Yolo 43
3.4. Visual Geometry Group-19 52
3.5. Çekirdek Korelasyon Filtresi 53
3.6. Veri İlişkilendirme 56
3.7. Çoklu Yaya Takibi Yöntemlerinin Birleştirilmesi 57
3.8. Başarı Ölçüm Kriterleri ve MOT Kıyaslama 62
4. BULGULAR VE TARTIŞMA 63
4.1. Farklı Takip Algoritmaları Kıyaslama 63
4.1.1. Mosse ile Tekli Nesne Takibi Sonuçları 63
4.1.2. Kalman Filtresi ile Tekli Nesne Takibi Sonuçları 65
4.1.3. Çekirdek Korelasyon Filtresi ile Tekli Nesne Takibi Sonuçları 67
4.1.4. Hiyerarşik Çekirdek Korelasyon Filtresi ile Tekli Nesne Takibi Sonuçları 69
4.1.5. Tekli Nesne Takibi Sonuçlarını Karşılaştırma 72

4.1.6. OpenCV Takip Yöntemleri ile Karşılaştırma 73
4.2. Farklı Özniteliklerin Takibe Etkisi 75
4.3. Çoklu Yaya Takibi 77
4.3.1. Derin Özellikler ile Veri İlişkilendirme 79
4.3.2. Renk Histogramı Özellikleri ile Veri İlişkilendirme 81
4.3.3. Çevreleyici Kutular ile Veri İlişkilendirme 82
4.3.4. Çoklu Yaya Takibi Yöntemleri Karşılaştırması 84
5. SONUÇ 89

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
BN	Yığın Normalleştirme
CNN	Evrişimli Sinir Ağı
FC	Tamamen Bağlı
FP	Yanlış Pozitif
FPS	Saniyede İşlenen Çerçeve Sayısı
GMCP	Genelleştirilmiş Minimum Klik Problemi
GMMCP	Genelleştirilmiş Maksimum Çoklu Klik Problemi
GPU	Grafik İşlemci Birimi
GT	Gerçek Değer
HOG	Yönelimli Eğimlerin Histogramı
IOU	Kesişimlerin Birleşime Oranı
KCF	Çekirdek Korelasyon Filtresi
KF	Kalman Filtresi
MOSSE	Karesel Hata Toplamının Minimum Çıktısı
MOT	Çoklu Nesne Kıyaslama
MOTA	Çoklu Nesne Takibi Doğruluk
MOTP	Çoklu Nesne Takibi Kesinlik
PAN	Yol Toplama Ağı
RGB	Kırmızı Yeşil Mavi
SAM	Uzamsal Dikkat Modülü
SORT	Basit Çevrimiçi ve Gerçek Zamanlı İzleme
TP	Doğru Pozitif

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1. Nesne tespitinde dönüm noktaları 4
Şekil 2.2. Yapay zeka, makine öğrenmesi, yapay sinir ağları, derin öğrenme ilişkisi ... 5
Şekil 2.3. Tek nöronlu yapay sinir ağı modeli 5
Şekil 2.4. Çok katmanlı yapay sinir ağı modeli 6
Şekil 2.5. Bilinen ilk denetimli ileri beslemeli çok katmanlı derin öğrenme modeli 7
Şekil 2.6. LeNet mimarisi 8
Şekil 2.7. Konvolüsyon işlemi 9
Şekil 2.8. Havuzlama işlemi 10
Şekil 2.9. Aktivasyon fonksiyonları 11
Şekil 2.10. Evrişimli sinir ağı katmanlarının görselleştirilmesi 12
Şekil 2.11. Kalman filtresi 16
Şekil 2.12. Kalman filtresi tahmin ve güncelleme adımı 17
Şekil 2.13. Parçacık filtresi genel yapısı 19
Şekil 2.14. Çoklu nesne takibinde veri ilişkilendirme 22
Şekil 2.15. Kesişimin birleşime oranı 24
Şekil 2.16. Macar algoritması örnek atama maliyetleri 25
Şekil 2.17. Macar algoritması birinci adım 26
Şekil 2.18. Macar algoritması ikinci adım 26
Şekil 2.19. Macar algoritması üçüncü adım 26
Şekil 2.20. Macar algoritması dördüncü adım 27
Şekil 2.21. Macar algoritması üçüncü adım tekrarı 27
Şekil 2.22. Macar algoritması optimum atama 27
Şekil 2.23. Macar algoritması optimum atama maliyetleri 28
Şekil 2.24. Küresel en yakın komşu örnek atama 28
Şekil 2.25. Küresel en yakın komşu atama problemi 29
Şekil 2.26. 3 zaman adımı ve 9 gözlem içeren bir maliyet akışı ağı örneği 30
Şekil 2.27. Çift taraflı eşleştirme ve GMCP karşılaştırması 31
Şekil 2.28. GMCP varsayım düğümü ekleme 32
Şekil 2.29. Tespit ve hata atamaları 39
Şekil 3.1. OTB-100 veri seti 42
Şekil 3.2. Yolo modeli 43
Şekil 3.3. Yolov1 ağ mimarisi 44
Şekil 3.4. Yolo9000 eğitiminde WordTree hiyerarşik yapısı ile veri seti birleştirme 45
Şekil 3.5. Yolov1'den Yolov2'ye yol haritası 46
Şekil 3.6. Çevreleyici kutunun boyut öncelikleri ve konum tahmini ile bulunması 47
Şekil 3.7. Darknet-53 48
Şekil 3.8. Tek ve çift aşamalı nesne tespit ağları 48
Şekil 3.9. SAM 50
Şekil 3.10. PAN 51
Şekil 3.11. Farklı nesne tespiti yöntemlerinin hız ve doğruluğunun karşılaştırması 51
Şekil 3.12. VGG-19 derin öğrenme ağı 52
Şekil 3.13. VGG-19 katmanları 52
Şekil 3.14. Kayma matrisi gösterimi 55

Şekil 3.15. Derin özellikler ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı 59
Şekil 3.16. RGB renk histogramı ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı 60
Şekil 3.17. IOU ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı 61
Şekil 4.1. MOSSE OTB-100 kesinlik ve başarı sonucu 64
Şekil 4.2. Kalman OTB-100 kesinlik ve başarı sonucu 66
Şekil 4.3. Çekirdek korelasyon filtresi OTB-100 kesinlik ve başarı sonucu 67
Şekil 4.4. Hiyerarşik çekirdek korelasyon filtresi OTB-100 kesinlik ve başarı sonucu 70
Şekil 4.5. VGG-19 katmanları 70
Şekil 4.6. Tekli nesne takibi karşılaştırma 72
Şekil 4.7. OpenCV yöntemleri ile tekli nesne takibi karşılaştırma 73
Şekil 4.8. Tekli nesne takibi woman veri seti karşılaştırma 75
Şekil 4.9. Çekirdek korelasyon filtresi ile farklı görünüm modellerinin karşılaştırması kesinlik sonucu 76
Şekil 4.10. Çekirdek korelasyon filtresi ile farklı görünüm modellerinin karşılaştırması başarı sonucu 76
Şekil 4.11. ETH-Pedcross2 veri seti çoklu yaya takibi kimlik değişikliği problemi (a) derin özellikler (b) renk histogramı özellikleri (c) çevreleyici kutular ile veri ilişlilendirme yapıldığında takip edilen yayalar 87
Şekil 4.12. KITTI-13 çoklu yaya takibi kimlik değişikliği problemi (a) derin özellikler (b) renk histogramı özellikleri (c) çevreleyici kutular ile veri ilişlilendirme yapıldığında takip edilen yayalar 88

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 2.1. Nesne tespiti yönteminin nesne takibi üzerindeki etkisi 33
Çizelge 2.2. DeepSORT evrişimli sinir ağı modeli..... 34
Çizelge 2.3. MOT veri setindeki videolara ilişkin bilgiler..... 35
Çizelge 2.4. MOT değerlendirme ölçütlerine genel bakış..... 36
Çizelge 2.5. Karmaşıklık matrisi 37
Çizelge 3.1. 2D MOT 15 eğitim ve test veri seti 41
Çizelge 3.2. İki boyutlu tespit dosyası örnek verisi 42
Çizelge 3.3. Korelasyon ve evrişim farkları 54
Çizelge 4.1. MOSSE yöntemi OTB-100 sayısal sonuçlar..... 65
Çizelge 4.2. Kalman filtresi OTB-100 sayısal sonuçlar 66
Çizelge 4.3. Çekirdek korelasyon filtresi OTB-100 sayısal sonuçlar 68
Çizelge 4.4. Hiyerarşik çekirdek korelasyon filtresi OTB-100 sayısal sonuçlar 71
Çizelge 4.5. Tekli nesne takibi fps karşılaştırma 74
Çizelge 4.6. Farklı görünüm modelleri fps karşılaştırma 77
Çizelge 4.7. Derin özellikler ile kosinüs mesafesi kullanılarak veri ilişkilendirme yapıldığında çoklu yaya takibi sonuçları 80
Çizelge 4.8. Renk histogramı özellikleri ile kosinüs mesafesi kullanılarak veri ilişkilendirme yapıldığında çoklu yaya takibi sonuçları..... 82
Çizelge 4.9. Çevreleyici kutular ile IOU metriği kullanılarak çoklu yaya takibi sonuçları..... 83
Çizelge 4.10. Çoklu yaya takibinde kullanılan farklı veri ilişkilendirme yöntemlerinin performans karşılaştırması 84

1. GİRİŞ

Temel fikri çok eski tarihlere dayanan yapay zekâ insan beyninin makineler ile modellenmesine dayanır ve makinelerin deneyimlerinden öğrenmesi amaçlanır. Günümüzde otonom araçlardan savunma sanayisine, güvenlikten eğlenceye, tıptan turizm sektörüne kadar birçok alanda kullanımı mevcuttur. Gelecekte insan hayatının kolaylaşması, tıpta erken teşhislerin artması, otonom araçların yaygınlaşması gibi durumlar yapay zekânın kullanımının artması ile beklenen sonuçlardır. Yapay zekânın alt dalı olarak kabul edilen makine öğrenmesi verilerden öğrenmeyi amaçlar. Verileri matematiksel işlemler ile modelleyerek tahminlerde bulunur. Etiketlenmiş ya da etiketlenmemiş verilerin çeşitli makine öğrenmesi algoritmaları ile kullanılması anomali tespiti, nesne tespiti, hasta hücrelerin tespiti gibi birçok konuda karar mekanizmaları oluşturur. Yapay sinir ağı ise makine öğrenmesinde sıklıkla kullanılan bir öğrenme yöntemidir ve insan beyninin sinir yapısı düşünülerek modellenir. Yapay sinir ağı deneyimlerden öğrenir ve geleneksel algoritmalardan farklı olarak yapay sinir ağının öğrendiği modellerin insanlar tarafından anlaşılması güçtür. Derin öğrenmede çok katmanlı yapay sinir ağları kullanılır ve ilk derin öğrenme ağı Ivakhnenko ve Lapa (1966) tarafından oluşturulmuştur. O zamandan günümüze bu yöntem büyük bir gelişme göstermiş, ancak bu gelişme ile birlikte yüksek bir hesaplama gücü ihtiyacı belirmiştir. Grafik işlemci birimi (GPU) donanımının kapasitesinin artması ile günlük hayatta kullanımı da hız kazanmıştır. Evrişimli sinir ağı bir derin öğrenme yöntemidir ve bilgisayarla görme, görüntü işleme, ses işleme, doğal dil işleme gibi birçok alanda kullanımı mevcuttur. Çeşitli katmanlardan oluşan bu model görüntü üzerinde çalışırken kenar, çizgi, eğri gibi yerel özelliklerden nesneyi ötekilerden ayırt edecek genel özelliklere kadar birçok özellik belirler.

Nesne takibi bilgisayarla görme alanında önemli bir yere sahiptir. Hareketli ya da sabit kameralardan alınan bir dizi görüntü üzerinde hedef nesnelerin durumunun izlenmesi işlemi olarak tanımlanabilir. Nesne takibi çalışmalarını çevrim içi ve çevrim dışı yapılan çalışmalar olarak iki gruba ayırmak mümkündür. Gerçek zamanlı görüntüler üzerinde hedef nesnelerin takip edilmesi ve konumunun tahmin edilmesi işlemi çevrim içi takip olarak adlandırılmaktadır. Çevrim dışı takip işlemi ise var olan video görüntüleri üzerinde hedef nesnelerin konumunun izlenmesi olarak adlandırılmaktadır. Nesne

takibinin etkin olarak yer aldığı pek çok uygulama alanı bulunmaktadır. Bu uygulama alanlarına örnek olarak otonom araçlar ve robotlar, video izleme, insan makine etkileşimi, tıbbi uygulamalar, artırılmış gerçeklik, video indeksleme, trafik kontrolü ve benzeri verilebilir. Nesne takibi çalışmalarında sıklıkla karşılaşılan problemler aydınlatma değişiklikleri, hızlı hareket, poz değişimleri, kısmi tıkanmalar ve arka plan karmaşıklığı gibi durumlardır. Bu gibi durumlarda nesne takibi zorlaşmaktadır. Mevcut çalışmalar incelendiğinde her zorluk durumu için başarılı bir algoritma geliştirmenin zor olduğu ve bu nedenle çalışmaların belirli bir alt problemin çözümüne odaklandığı görülmektedir. Çoklu nesne takibi bu zorlu durumlar göz önünde tutularak görüntüdeki bütün nesnelerin doğru kimlik bilgileri ile takip edilmesidir. Korelasyon filtresi ise nesnelerin takip edilmesi için yaygın kullanılan bir takip algoritmasıdır.

Bu tezde derin öğrenme yöntemlerinden olan evrişimli sinir ağına dayanan bir model kullanarak yayaların öncelikle tespit edilmesi amaçlanmıştır. Tespit edilen yayalar daha sonra korelasyon filtresi ile takip edilmiş ve tekrar tespit edilen yayalar ile eşleştirmek için bu ağın özellik vektörlerinden faydalanılmıştır. Gerçekleştirilen sistemin önce tekli nesne takip başarımı sonra da çoklu nesne takip başarımı başarı, kesinlik, kimlik değişimi metrikleri kullanılarak değerlendirilmiştir.

Tezin takip eden bölümlerinden Bölüm 2'de çalışmanın alt yapısında bulunan Yolo, korelasyon filtresi, veri ilişkilendirme ve özellik modelleri hakkında ayrıntılı bilgiler sunulmaktadır. Bölüm 3'te kullanılan veri seti, donanım ve Yolov4 ile çekirdek korelasyon filtresi yöntemlerine ilişkin algoritmaların detayları mevcuttur. Bölüm 4'te gerçekleştirilen deneyler sunulmakta, performans ile başarı değerlendirmelerine yer verilmekte ve deney sonuçları yorumlanmaktadır. Bölüm 5'te ise çalışmanın nihai sonuçları özetlenmiş, genel değerlendirmelere yer verilmiş ve gelecekte bu alanda yapılabilecek çalışmalara değinilmiştir.

2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI

Nesne tespiti, nesne takibi ve veri ilişkilendirme adımlarını içeren çoklu nesne takibi için literatürde çeşitli yöntemler mevcuttur. Bu bölümde nesne tespiti klasik ve modern nesne tespiti yöntemleri şeklinde gruplandırılarak ele alınmıştır. Aynı zamanda yapay zekâ ve derin öğrenmenin nesne tespiti ve takibi için kullanımından bahsedilmiştir. Nesne tespiti ve takibine ek olarak çoklu nesne takibi çalışmaları için büyük öneme sahip veri ilişkilendirme yöntemlerine değinilmiştir.

2.1. Nesne Tespitinde Kullanılan Klasik Yöntemler

Nesne tespiti görüntü işleme alanında birçok uygulamada önem arz etmektedir. Literatürde bu alanda klasik olarak bilinen çeşitli çalışmalar mevcuttur. İki görüntü arasındaki geçici farkın bulunması ile nesne tespiti gerçekleştirilebilir ancak ışık değişimi ve gürültüye karşı hassas olması bu yöntemin başarısını düşürmektedir.

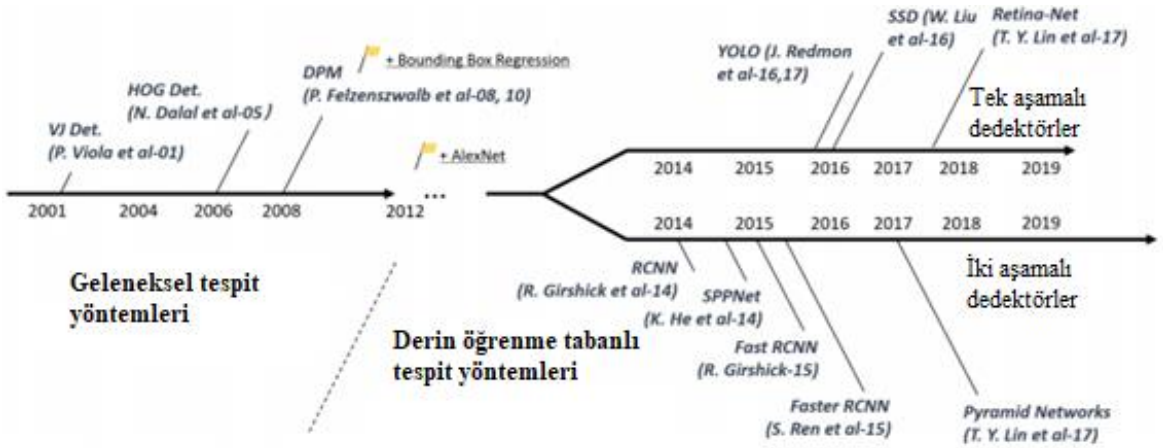
Bir diğer yöntem ise tespit edilecek nesnenin arka plandan çıkartılarak nesne tespitinin yapılmasıdır. Ancak bu yöntemde morfolojik işlemlerin uygulanması ile başarı elde edilse bile çoklu nesne tespiti için kullanışlı olmamaktadır. Shaikh ve diğerlerinin (2014) belirttiği gibi arka plan görüntüsünün dinamik olarak güncellenmesi ise daha gelişmiş bir yöntem olmakla beraber ortam değişiklikleri ve ışık değişimi gibi durumlara hassasiyetinden dolayı bu görüntüyü sürekli güncellemek oldukça önemlidir.

Bir diğer nesne tespiti yöntemi olan optik akış yönteminde arka plan görüntüdeki piksel hareketliliğine bakılarak çıkartılmaktadır. Ancak bu yöntemin de başarı oranı karmaşık arka plana sahip görüntülerde ve hareketli kamera çekimlerinde düşecektir.

2.2. Nesne Tespitinde Kullanılan Modern Yöntemler

Nesne tespiti yöntemleri bilgisayarların işlem kapasitesinin artması ve makine öğrenmesi algoritmaları ile gelişme göstermiştir. Zou ve diğerlerinin (2019) belirttiği gibi dönüm noktası olan 2012 yılına kadar kabul gören yöntemler yönelimli eğimlerin histogramı (HOG) özellik çıkarma ve Viola Jones integral görüntü hesaplama gibi

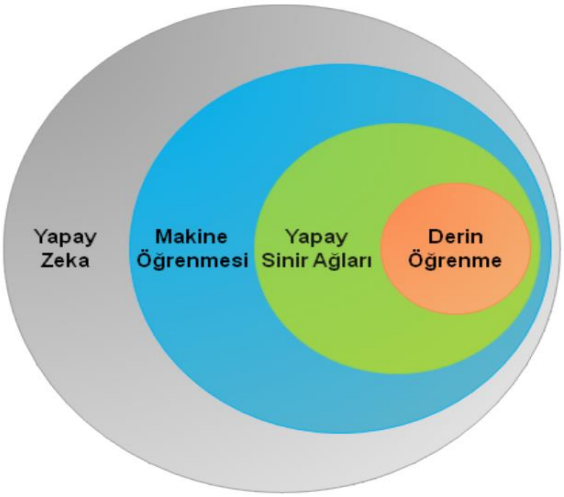
yöntemlerdir. 2012 yılında düzenlenen ILSVRC-2012 yarışmasını Alexnet'in kazanması ile derin öğrenmenin başarılı sonuçlar vereceği gösterilmiştir ve bu tarihten itibaren birçok alanda derin öğrenme kullanılmaya başlanmıştır (Şekil 2.1). Nesne tespiti ve konumlandırılması için günümüze kadar çeşitli ağ yapıları kullanılmıştır. Bu yapıları tek aşamalı dedektörler ve çift aşamalı dedektörler olarak sınıflandırmak mümkündür. Çift aşamalı dedektörlerde ilk aşamada bölge önerileri üretilirken sınıflandırma işlemi ikinci aşamada gerçekleştirilir. Tek aşamalı dedektörlerde ise bölgeler üretilmeden sınıflandırma ve regresyon işlemleri tek adımda gerçekleştirilir.



Şekil 2.1. Nesne tespitinde dönüm noktaları (Zou ve diğerleri, 2019)

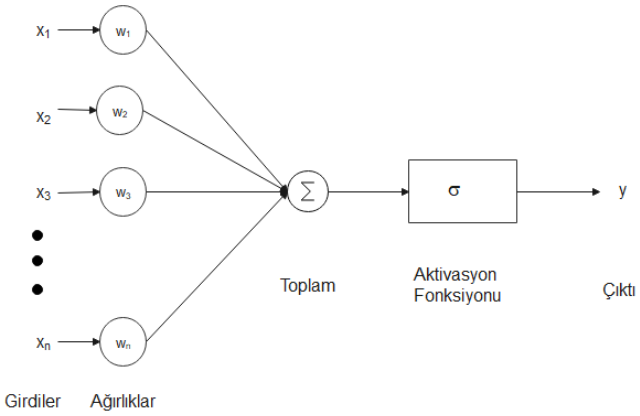
2.3. Yapay Sinir Ağı

Yapay sinir ağı 1943 yılında Warren McCulloch ve Walter Pitts tarafından gerçekleştirilen bir çalışma ile ilk olarak tanınmıştır ("History of Artificial Neural", 2021). Yapay sinir ağları yapay zekânın alt kümesinde yer alır (Şekil 2.2). İnsan beyninin sinir yapısının, deneyimlerinden öğrenme ve hatırlama yönteminin matematiksel modellenmesine verilen isimdir. İnsan beyninin öğrenme ve karar alma şeklinin bilgisayar ortamına aktarılması için beynin nöron ağının modellenmesi amaçlanır. Tahmin, sınıflandırma, üretim planlama, otonom araçlar, sahte bilgi tespiti, zararlı e-postaların belirlenmesi ve daha birçok alanda yapay sinir ağı kullanımı mevcuttur.



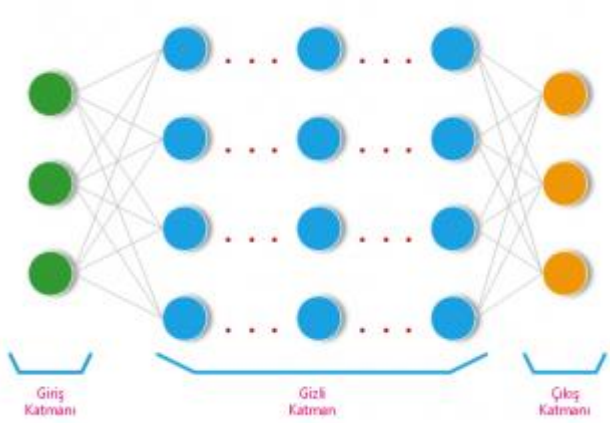
Şekil 2.2. Yapay zeka, makine öğrenmesi, yapay sinir ağları, derin öğrenme ilişkisi

Ayşe ve Berberler'in (2017) belirttiği gibi yapay sinir ağları tek katmanlı ve çok katmanlı olarak iki gruba ayrılır. Tek katmanlı yapay sinir ağlarında girdi ve çıktı katmanları bulunur. 1958 yılında Frank Rosenblatt tarafından geliştirilen tek nöronlu bir tek katmanlı sinir ağı modelidir (Şekil 2.3). Bu modelde çıktı değerini hesaplamak için girdi değerleri ağırlıkları ile çarpılarak toplanır ve bir eşik değeri ile karşılaştırılır. Girdi değerlerinin ağırlıklarla çarpımı ve toplanması sonucu elde edilen sayı eşik değerinden büyük bir sayı ise 1 değilse 0 olarak sonuç hesaplanır. Ancak tek katmanlı modellerin XOR problemi gibi problemleri çözemediği belirlenmiştir. Bu eksiklik çok katmanlı modellerin geliştirilmesiyle ortadan kalkmıştır.



Şekil 2.3. Tek nöronlu yapay sinir ağı modeli

Çok katmanlı modellerde farklı olarak bir ya da birden fazla ara katman bulunur (Şekil 2.4). Katmanlar arası bilgi aktarımı ileri yayılım ve geri yayılım adı verilen yöntemlerle gerçekleştirilir. İleri yayılım neticesinde hesaplanan hata değeri, geri yayılım işlemi sırasında hatayı en aza indirmek amacıyla ağırlıkların değerlerini güncelleme hesabında kullanılır.



Şekil 2.4. Çok katmanlı yapay sinir ağı modeli (Makinist, 2018)

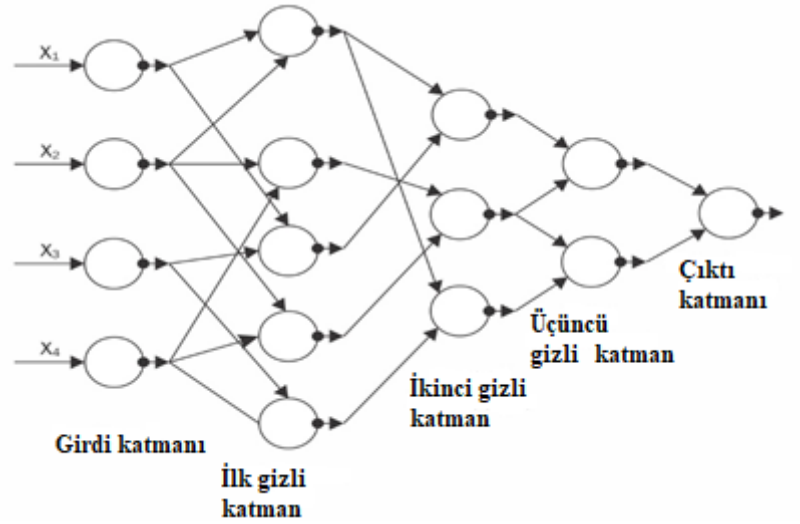
Yapay sinir ağlarını ileri beslemeli ağlar ve geri beslemeli ağlar olarak iki gruba ayırmak mümkündür. İleri beslemeli ağlarda giriş katmanından alınan girdi değerleri ağırlıklar ile çarpıldıktan sonra aktivasyon fonksiyonu uygulanarak gizli katmana, daha sonra da çıktı katmanına iletilir (Makinist, 2018). Geri beslemeli ağlarda ise girdi değerleri çıktı katmanına gelmeden önce geri yayılım yöntemi uygulanır ve hata oranı istenilen değere yaklaştığında ya da belirli sayıda bu adım tekrarlandıktan sonra işlem tamamlanır.

2.3.1.Derin Öğrenme

Derin öğrenme yapay sinir ağlarının alt dalı olarak kabul görmektedir. Schmidhuber'in (2015) belirttiği gibi yapay sinir ağlarında ara katman ve düğüm sayısı arttıkça hesaplama yükü de artış göstermiştir. Bununla beraber donanım yetersizliği bu alanda duraksamaya neden olmuştur. Ancak Grafik işlem biriminin gelişimi ve donanımdaki ilerlemeler ile birlikte bu alandaki geliştirmelere devam edilmiştir. Derin öğrenme

modellerinde her katman kendinden önceki katmanın çıktısını girdi olarak alır ve özellik çıkarmayı amaçlar.

Şeker ve diğerlerinin (2017) belirttiği gibi ilk denetimli ileri beslemeli çok katmanlı derin öğrenme modeli Ivakhnenko ve Lapa'nın (1966) geliştirdiği derin yapay ağ modelidir (Şekil 2.5). Bu çalışmada her katmandaki özellikler bir sonraki katmana iletilir ve geri yayılım işlemi uygulanmaz. Tekrarlayan sinir ağları, uzun kısa vadeli hafıza ağları, evrişimli sinir ağı gibi birçok derin öğrenme mimarisi mevcuttur ve hepsi farklı alanlarda etkili kullanıma sahiptir.

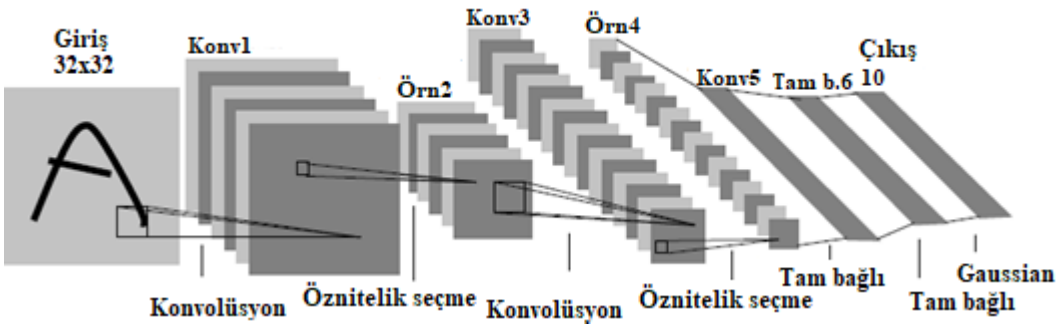


Şekil 2.5. Bilinen ilk denetimli ileri beslemeli çok katmanlı derin öğrenme modeli

2.3.2.Evrişimli Sinir Ağları

Şeker ve diğerlerinin (2017) ifade ettiği gibi evrişimli sinir ağı çok katmanlı bir yapay sinir ağı modelidir. Görüntü işleme, doğal dil işleme, ses işleme gibi pek çok alanda kullanımı mevcuttur. Özellikle görüntü ve video işleme alanında başarılı sonuçlar vermesi, bu alandaki çalışmaların çoğunun derin öğrenmeye yönelmesini sağlamıştır. Bu mimaride insan gözünün görme yetisinin modellenmesi amaçlanmıştır. LeCun ve diğerlerinin 1988'de geliştirdiği LeNet mimarisi ilk evrişimli sinir ağı modelidir. Bu mimari posta kodları, rakamlar ve benzer uygulamalar için kullanılmıştır.

LeNet mimarisi ReLU aktivasyonunu kullanan bir dizi konvolüsyon, ortalama havuzlama, tam bağlı katman ve 10 sınıflı bir yumuşatma katmanını içerir (Şekil 2.6). Giriş verisi ise 32x32 boyutlu görüntünün piksel değerlerini içeren bir matristir (Doğan ve Türkoğlu, 2019).



Şekil 2.6. LeNet mimarisi (LeCunn ve diğerleri, 1998)

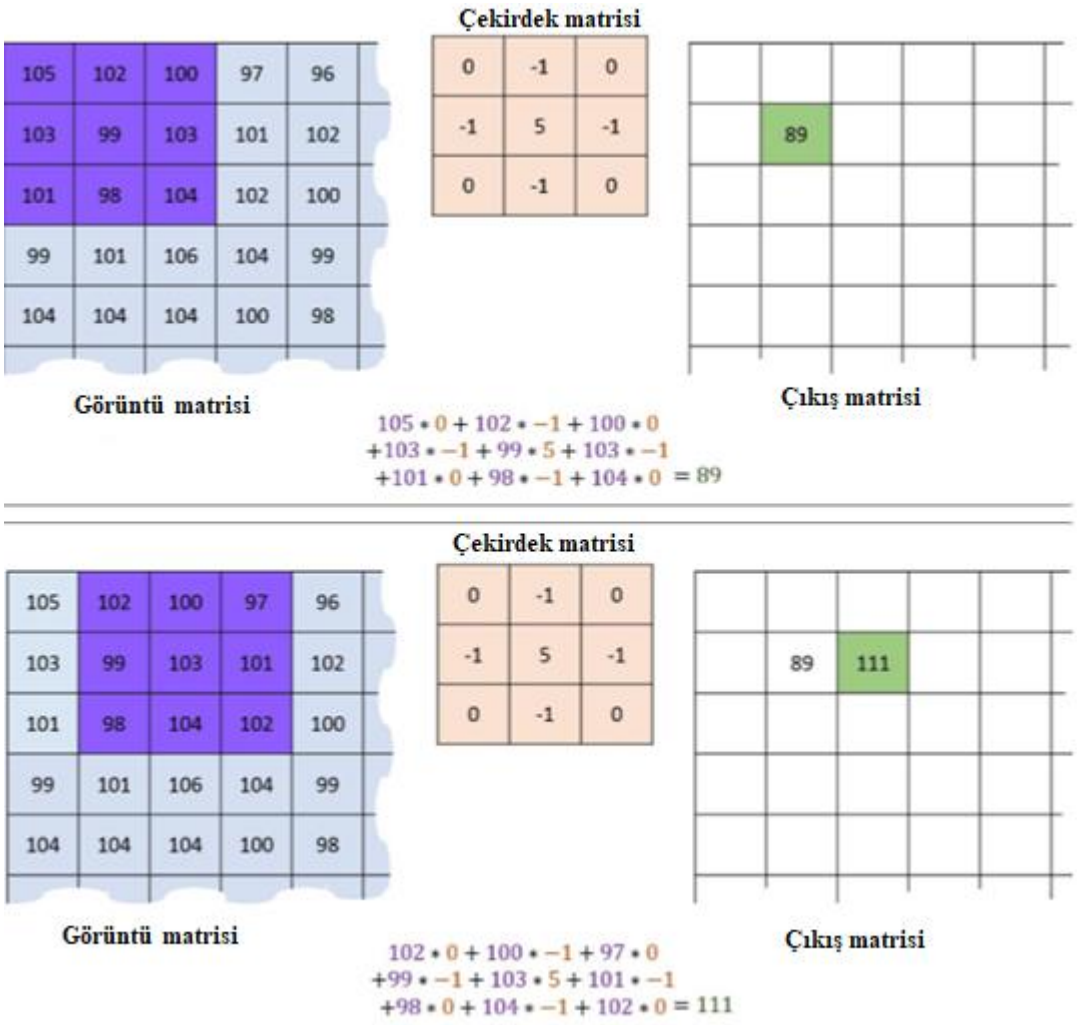
Evrişimli sinir ağı genel olarak giriş, evrişim ve havuzlama katmanları ile tam bağlı katmandan oluşur. Evrişim katmanı bu yapının temel elemanıdır ve farklı filtreler kullanarak düşükten yüksek seviyeye özellikleri çıkarmayı amaçlar. Özellik çıkarmak için öğrenilmiş filtreler görüntü pikselleri üzerinde gezdirilerek konvolüsyon işlemi uygulanır ve bu şekilde kenar, köşe, doğru gibi düşük seviyeli özellikler ya da resme ait daha yüksek seviyeli özellikler çıkartılmış olur.

Örnek bir konvolüsyon işleminde örnek görüntü üzerinde filtre sol yukarıdan sağ aşağıya doğru adım adım gezdirilir. Her adımda görüntü üzerindeki sayılar ile filtre üzerindeki örtüşen sayılar çarpılarak hepsi toplanır ve sonuç matrisine eklenir (Şekil 2.7). Bu şekilde hesaplama yapıldığı zaman görüntünün kenarlarındaki piksel değerleri hesaplama işlemine dâhil edilemez. Görüntü piksellerinin tüm kenarlarına 0 eklenerek bu eksiklik giderilir ve tüm görüntü piksellerinin hesaplamalarda kullanılması sağlanır. Bu işleme sıfır ekleme adı verilir. Filtre matrisinin görüntü matrisi üzerinde gezdirilmesi sırasında atlanan adım sayısına ise kaydırma (stride) adı verilir. İlk konvolüsyon katmanında bu işlem renkli görüntüler için üç kanallı bir görüntü üzerinde gerçekleştirilir ve filtrelerin üç boyutlu bir matris şeklinde oluşturulup görüntü boyunca kayarak işlem yapması sağlanır. 8 tane filtre kullanılması durumunda ve girdi görüntüsü

32x32x3 olması durumunda ilk konvolüsyon katmanı sonrası 32x32x8 boyutlu bir matris çıktısı oluşur. NxN boyutlu bir girdi matrisi ve FxF boyutlu filtre matrisi için P kadar sıfır eklendiği takdirde katman çıktı matrisi boyutu (2.1)'deki gibidir. Kaydırma değerinin birden farklı olması durumunda ise (2.1)'deki formül, (2.2)'deki formül ile güncellenir.

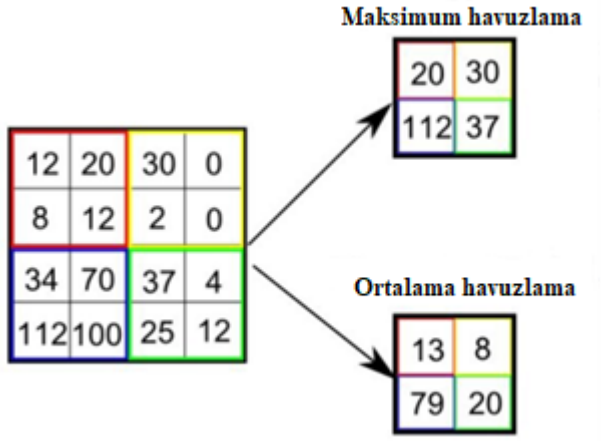
$$(\mathrm{N} + 2\mathrm{P} - \mathrm{F} + 1)\mathrm{x}(\mathrm{N} + 2\mathrm{P} - \mathrm{F} + 1) \tag{2.1}$$

$$((N + 2P - F)/S + 1)x((N + 2P - F)/S + 1) \tag{2.2}$$



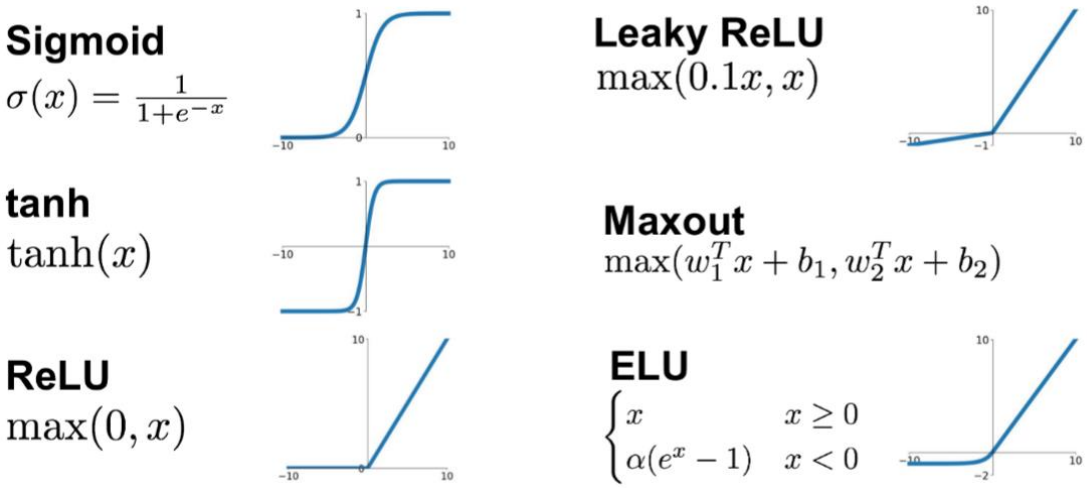
Şekil 2.7. Konvolüsyon işlemi (İleri, 2018)

Sonraki adımda ReLU aktivasyon fonksiyonu uygulanır ve bu adımda matris boyutu değişikliğe uğramaz. Daha sonra havuzlama katmanında boyut azaltma işlemi gerçekleştirilir. Bu amaçla maksimum havuzlama (maxpooling) ve ortalama havuzlama (average pooling) olmak üzere iki yöntem mevcuttur. Şekil 2.8'de sağ üst bölgede görülen maksimum havuzlama işleminde matris içerisindeki değerlerden en yüksek olanı seçilirken Şekil 2.8'de sağ alt bölgede görülen ortalama havuzlama işleminde ise matris içerisindeki değerlerin ortalaması seçilir. Genel bir havuzlama katmanı sonrası 32x32x8 boyutlu girdi matrisi 16x16x8 boyutlu bir matrise dönüştürülür.



Şekil 2.8. Havuzlama işlemi (Saha, 2018)

Tam bağlantılı katmandan önceki katman çıktıları sinir ağı için tek boyutlu bir vektör haline dönüştürülür. Ardından son katman olan çıkış katmanından önceki tam bağlı (fully connected - FC) katmana geçiş yapılır. Bu katmanda klasik yapay sinir ağı yöntemiyle öğrenme işlemi gerçekleştirilir ve çıkış verisi oluşturulur. Yapay sinir ağlarında kullanılan birçok aktivasyon fonksiyonu bulunmaktadır. Şekil 2.9'da bazı aktivasyonlar fonksiyonları verilmiştir. Sigmoid aktivasyon fonksiyonu kullanılırken bazı problemlere neden olmaktadır. Bunlar doymuş nöronların gradyanları öldürmesi, çıktılarının sıfır merkezli olmaması ve üstel işlemi hesaplama maliyetinin yüksek olmasıdır. Bu nedenle evrişimli sinir ağlarında kullanılması tercih edilmemektedir ("Training Neural Networks", 2021).



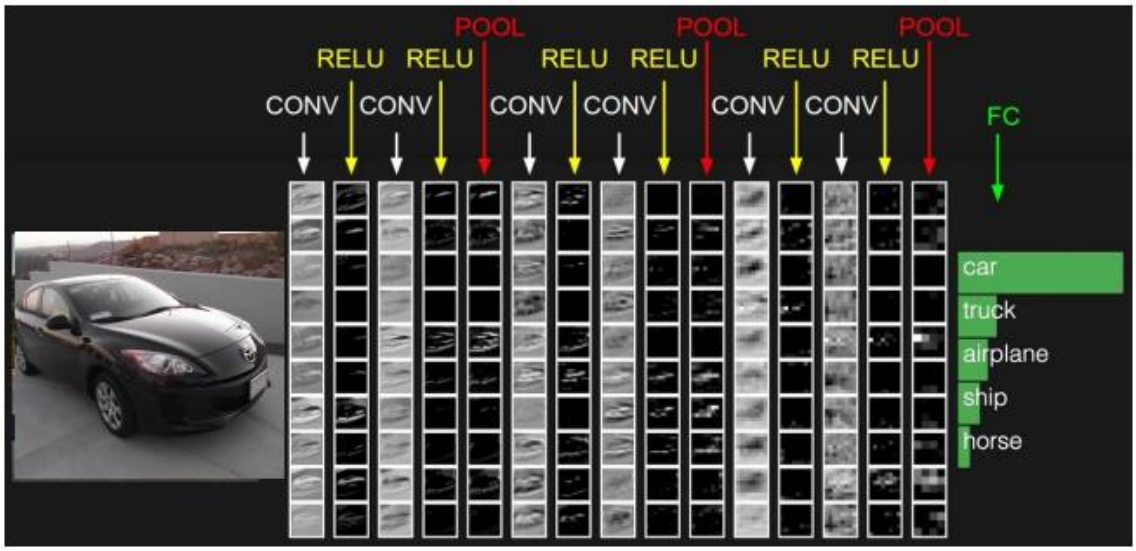
Şekil 2.9. Aktivasyon fonksiyonları (“Training Neural Networks”, 2021)

Tanh aktivasyon fonksiyonu sıfır merkezli olmasına rağmen doymuş nöronların gradyanları öldürmesi problemi mevcuttur. ReLU aktivasyon fonksiyonu pozitif bölgede doygunlaşmaya neden olmaz ve hesaplama açısından etkilidir. Pratikte ötekilere kıyasla daha hızlı yakınsar ve sigmoide kıyasla daha makul kabul edilir. Ancak nöron çıktısı sıfırdan düşük olduğu zaman sıfır gradyan problemi oluşabilir.

Leaky ReLU’nun hesaplama maliyeti düşüktür ve gradyan ölme durumu mevcut değildir. ELU ReLU’nun sahip olduğu tüm avantajlara sahiptir ancak üstel işlem gerektirdiğinden hesaplama maliyeti yüksektir.

Maxout aktivasyon fonksiyonu ReLU ve Leaky ReLU'yu genelleştirir, ancak nöron başına düşen parametre sayısının iki katına çıkmasına neden olur. Li ve diğerlerinin (2017) belirttiği gibi evrişimli sinir ağlarında genel olarak tercih edilen aktivasyon fonksiyonu ReLU fonksiyonudur.

Özetle Conv -> ReLU -> Conv -> ReLU -> Pool -> Conv -> ReLU -> Conv -> ReLU -> Pool -> Conv -> ReLU -> Conv -> ReLU -> Pool -> FC katmanlarına sahip bir evrişimli sinir ağı Şekil 2.10’da görselleştirilmiştir. Girdi olarak alınan görüntü tüm katmanlarda işlem gördükten sonra çıktı katmanına gelir ve çıktı değerlerindeki en yüksek olasılıklı etiket araba olduğu için sınıflandırma etiketi araba şeklinde belirlenir.



Şekil 2.10. Evrişimli sinir ağı katmanlarının görselleştirilmesi ("Training Neural Networks", 2021)

2.4. Nesne Takibi Yöntemleri

Video görüntülerindeki hareketli nesnelerin tespiti ve takibi için literatürde farklı yöntemler mevcuttur. Optik akış, Kalman filtresi, korelasyon filtresi, parçacık filtresi bu yöntemlerde yaygınlıkla kullanılan algoritmalardır. Klasik nesne takibi yaklaşımı alınan bir görüntü çerçevesinde basit bir hedef model oluşturmak ve daha sonraki çerçevelerde bu hedef modeli aramaktan ibarettir.

2.4.1.İlkel Nesne Takip Yöntemleri

Comanicu ve diğerlerinin (2000) çalışmalarında açıklanan ortalama kayma takipçisi yönteminde hedef model renk histogramlarıyla modellenir. Bhattacharyya uzaklık hesaplama yöntemi kullanılarak sonraki çerçevelerdeki aday bölgelerin renk histogramlarının dağılımı ile karşılaştırma yapılır ve hedef modelin konumu bulunur. Sonraki çerçevede en benzer konumun bulunması için ortalama kayma algoritması uygulanır.

Kalal ve diğerlerinin (2011) takip, öğrenme ve algılama yönteminde nesne takibi optik akış algoritmasına dayanır. Optik akış nesnelerin hareket bilgisini temsil eder. Optik

akış modeli temelde iki imge arasındaki uzay ve zaman farkının bulunması ile elde edilir. Hare ve diğerleri 2015'te Haar benzeri özellikler ve kernellerle yapılandırılmış çıktılı destek vektör makinasını kullanarak takip işlemini gerçekleştirmiştir.

Son yıllardaki çalışmalar takip yöntemine göre korelasyona, Bayes yöntemine ve veri ilişkilendirmeye dayalı yöntemler olarak gruplandırılabilir. Hedef model belirlemek için kullanılan özelliklere göre ise derin öğrenmeye dayalı ve daha düşük seviyeli özelliklere dayalı çalışmalar olarak sınıflandırılabilir. Korelasyon filtresi kullanılarak yapılan hedef nesne takibi çalışmaları Fourier uzayında işlem yapılması dolayısı ile oldukça hızlı çalışmaktadır ve bu nedenle gerçek zamanlı nesne takibi için tercih edilmiştir (Bolme ve diğerleri, 2010; Henriques ve diğerleri, 2014).

AdaBoost özellik çıkarma algoritmasının bir sürümü olan Boosting yöntemi çevrim içi eğitim işlemini gerçekleştirmektedir. Bu özelliği ile takip edilen nesnenin görünüm değişikliklerine karşı hızlı adapte olabilmektedir. Nesne görünüm modellerinin hızlı hesaplanabilir olması bu yöntemin gerçek zamanlı çalışmasına imkân sağlamaktadır. Gerçek zamanlı Çoklu Örnekle Öğrenme (Multiple Instance Learning - MIL) yöntemi nesneyi arka plandan ayırt etmek için çevrim içi ayırt edici bir sınıflandırıcı eğitmektedir. Eğitim için tek bir örnek yerine bir dizi görüntü yaması kullanmaktadır. MedianFlow çalışmasında izleme başarısızlık tespiti için ilerigeri hatasına dayanan yeni bir yöntem önerilmektedir. Geriye dönük izleme gerçekleştirilerek bir doğrulama yörüngesi oluşturulmakta ve söz konusu yörünge ile karşılaştırılmaktadır. Takip, öğrenme ve algılama (Tracking, Learning and Detection - TLD) yönteminin takip adımında nesnenin çerçeveden çerçeveye takibi, algılama adımında nesne görünüm modellerinin konumlandırılması ve takip adımının düzenlenmesi, öğrenme adımında ise algılama adımının hatalarını önlemek için güncelleme işlemleri gerçekleştirilmektedir. Ayrımcı korelasyon filtresi (Discriminative Correlation Filter - DCF) yöntemine dayanan Kanal ve Uzaysal Güvenilirlik İzleyicisi (The Channel and Spatial Reliability Tracker - CSRT) filtre boyutunu ayarlayabilmesi ile geleneksel DCF algoritmasını iyileştirmektedir (Brdjanin ve diğerleri, 2020).

2.4.2.Karesel Hata Toplamının Minimum Çıktısı

Bolme ve diğerleri (2010) karesel hata toplamının minimum çıktısı (MOSSE) çalışmasında nesne takibi için korelasyon filtresini kullanmıştır. Korelasyon filtresi işlem gücü hızlı olması dolayısıyla pek çok nesne takibi çalışmalarında tercih edilmektedir. Çalışmada takip edilecek nesnenin konumu başlangıçta belirlenerek nesnenin görünümü adaptif korelasyon filtreleri ile modellenir ve her bir hedef için Gauss cevabı oluşturulur, daha sonra nesne takibi ve filtre eğitimi birlikte çalışır. Sonraki görüntülerde belirlenen arama penceresindeki korelasyon çıktısına bakılarak en yüksek değerine göre nesnenin konumu belirlenir. Daha sonra çevrim içi güncelleme işlemi gerçekleştirilir. İşlem gücünün hızlandırılması için korelasyon işlemi Fourier uzayında gerçekleştirilir.

Korelasyon formülü, G, Gauss cevabının Fourier uzayına dönüştürülmüş formu; F, görüntünün Fourier uzayına dönüştürülmüş formu; H, filtrenin Fourier uzayına dönüştürülmüş formu olarak belirlendiğinde şu şekildedir:

$$G = F \odot H^* \tag{2.3}$$

$$H^* = (\sum G_i \odot F_i *)/(\sum F_i \odot F_i *) \tag{2.4}$$

Denklemdeki $\odot$ sembolü eleman bazında çarpımı, * ise karmaşık eşleniği temsil etmektedir. Filtre (H) konvolüsyonun gerçek çıktısı ve istenen çıktısı arasındaki hatanın karesinin toplamını en aza indiren bir optimizasyon problemi olarak çözülür. Bulunan filtre formül (2.4)'te gösterildiği şekildedir. Filtre çevrim içi güncellenerek görüntüdeki değişikliklere adapte olur. Saniyede 669 görüntü işleme kapasitesine sahiptir.

2.4.3.Çekirdek Korelasyon Filtresi

Henriques ve diğerleri (2014) çalışmasında yönelimli eğimlerin histogramı özelliği ile hedef nesnenin dairesel kaymalarını hesaplayarak korelasyon tabanlı hedef nesne takibi işlemini gerçekleştirir. Gauss kerneli kullanarak doğrusal korelasyonu çok kanallı

korelasyon filtresine çevirir ve böylece Fourier uzayında hızlı çalışacak korelasyon filtresi ile hedef nesne takibini gerçekleştirir.

Çekirdek korelasyon filtresinde de kullanılan yönelimli eğimlerin histogramı nesne algılama alanında oldukça sık kullanılan bir özellik tanımlayıcıdır. Görüntüdeki piksel eğimlerinin yönelim ve büyüklük değerlerini histogramlarla belirtir (Peker ve diğerleri, 2012). Yönelimli eğimlerin histogramı için yatay ve dikey Sobel filtreleri uygulanarak eğimlerin genliği (gradyan) ve yönelim açıları hesaplanır (Saygili ve Albayrak, 2018). G, gradyan; θ, yönelim eğimi; I, görüntü; S_y, dikey Sobel filtresi; S_x, yatay Sobel filtresi olarak belirlendiğinde gradyan ve yönelim açısı formülleri şu şekildedir.

$$I_x = I * S_y \tag{2.5}$$

$$I_y = I * S_d \tag{2.6}$$

$$|G| = \sqrt{{I_x}^2 + {I_y}^2} \tag{2.7}$$

$$\theta = arctan\frac{I_x}{I_y} \tag{2.8}$$

Görüntüdeki piksel eğimlerinin yönelim açıları 0-360 derece arasında eşit aralıklı kutulara gruplandığında baskın gradyanlar daha anlaşılır olabilmektedir (Peker ve diğerleri, 2012). Çalışmada ham pikseller ve yönelimli eğimlerin histogramı kullanılarak başarı ölçümleri yapılmış ve çekirdek korelasyon filtresinin yönelimli eğimlerin histogramı ile daha yüksek başarı sağladığı görülmüştür (Henriques ve diğerleri, 2014).

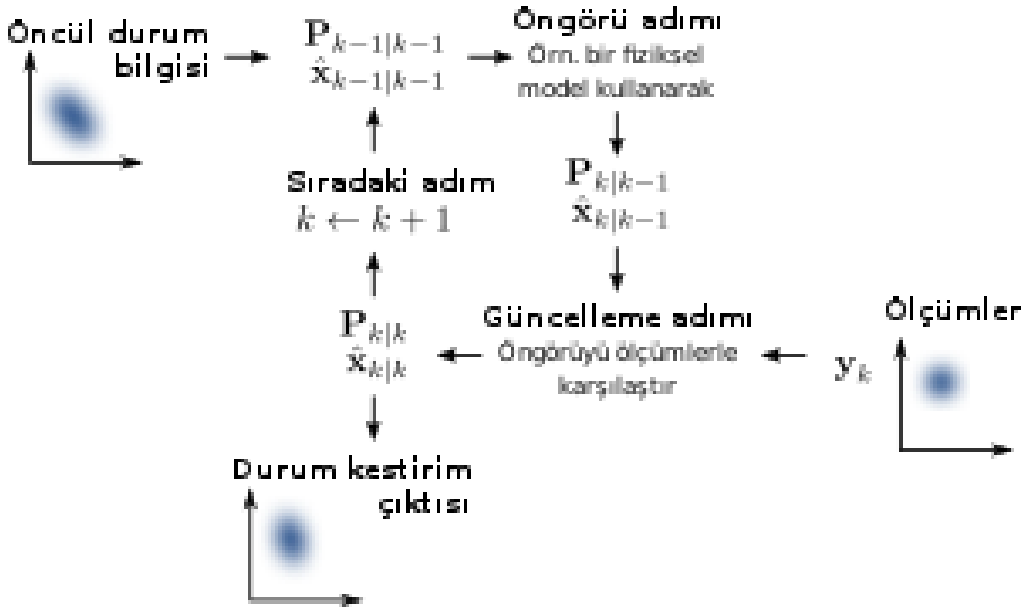
2.4.4.Bayes Yöntemi

Thomas Bayes tarafından isimlendirilen Bayes teoremi bir olayın olasılığını, olayla ilgili olabilecek koşulların önceki bilgisine dayanarak açıklar (Joyce, 2003). P(A|B), B olayı gerçekleştiğinde A olayının gerçekleşme olasılığı; P(B|A), A olayı

gerçekleştiğinde B olayının gerçekleşme olasılığı; P(A), marjinal olasılık olarak bilinen A olayının gözlemlenme olasılığı; P(B), marjinal olasılık olarak bilinen B olayının gözlemlenme olasılığı iken Bayes kuralı formülü şu şekildedir (Stuart ve Ord, 1994):

$$P(A|B) = P(B|A)P(A)/P(B) \tag{2.9}$$

Bayes yöntemine dayanan ve yaygınlıkla kullanılan takip yöntemleri Kalman filtreleme ve parçacık filtreleme tabanlı yöntemler olarak gruplandırılabilir. Kalman (1960) tarafından geliştirilen Kalman filtresi (KF) modelin önceki bilgilerinden sistemin durumunu tahmin eder ("Kalman Filtresi", 2021). Şekil 2.11'de gösterildiği gibi Kalman filtresi, sistemin tahmin durumunu takip eder ve tahminin varyansını veya belirsizliğini izler. Daha sonraki hesaplamalarda bir durum geçiş modeli kullanılarak tahmin güncellenir. $X_{k|k-1}$ y_k ölçümü hesaba katılmadan önce k adımındaki sistem durumunun tahminini ifade eder, $P_{k|k-1}$ ise bu tahminin belirsizliğini ifade eder.

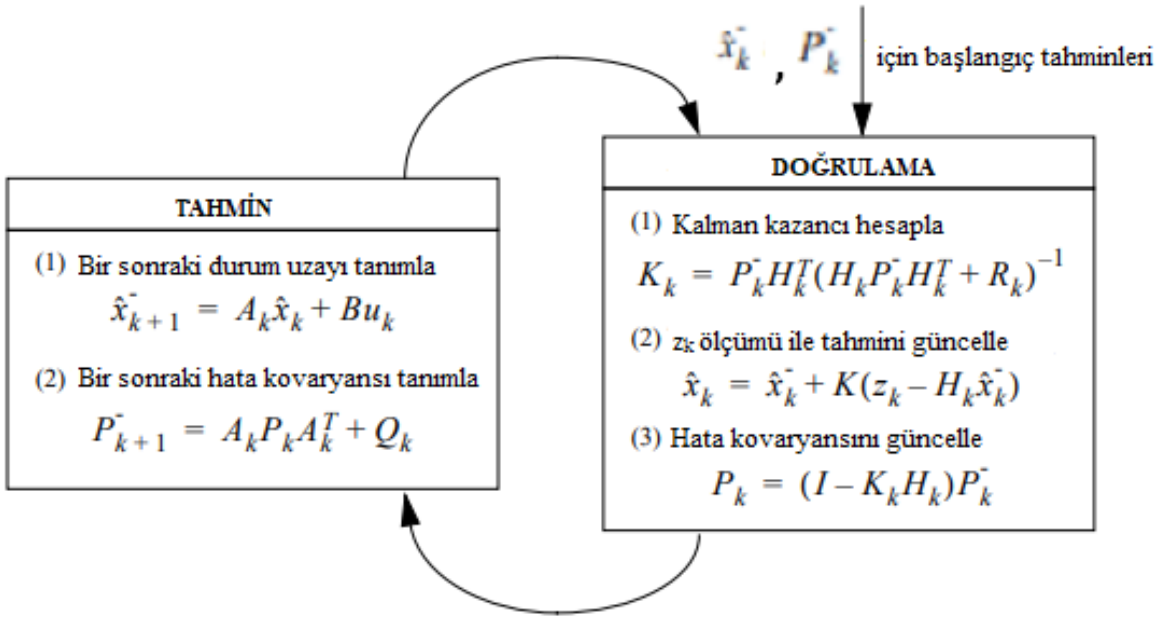


Şekil 2.11. Kalman filtresi ("Kalman Filtresi", 2021)

Başka bir ifade ile Kalman filtresi tahmin ve güncelleme olarak iki temel adımdan oluşur. $X_{k+1} = A_kX_k + Bu_k + w_k$ doğrusal stokastik fark denklemi tarafından yönetilen ayrık zaman kontrollü bir sürecin durumunu bir ölçüm $Z_k = H_kX_k + v_k$ ile tahmin eder.

Rasgele değer olan w_k değeri işlem gürültüsünü temsil eder ve beyaz, normal olasılık dağılımına $p(w) \approx N(0,Q)$ sahip olduğu varsayılır. Rasgele değer olan v_k değeri ise hesaplama gürültüsünü temsil eder ve beyaz, normal olasılık dağılımına $p(v) \approx N(0,R)$ sahip olduğu varsayılır. Gürültü değerleri w_k ile v_k'nın birbirinden bağımsız olduğu varsayılır. Pratikte işlem gürültüsü kovaryans matrisi Q ve hesaplama gürültüsü kovaryans matrisi R her adımda değişebilir, ancak uygulamada sabit kabul edilir. Boyutu n×n olan A matrisi durum geçiş modelini temsil ederken, boyutu n×1 olan B matrisi isteğe bağlı giriş kontrol değerini temsil eder. Boyutu m×n olan H matrisi ise mevcut durum bilgisini Z_k ölçümü ile ilişkilendirir (Welch ve Bishop, 1995).

Şekil 2.12'de Kalman filtresinin yinelemeli tahmin ve güncelleme (doğrulama) adımlarında kullanılan formüller gösterilmiştir. Tahmin adımında bulunan X_{k+1} ifadesi k+1 adımında ölçüm değerlerini görmeden önce tahmin edilen ortalama durumu, P_{k+1} ise tahmin edilen durumun kovaryans değerini temsil etmektedir.



Şekil 2.12. Kalman filtresi tahmin ve güncelleme adımı (Welch ve Bishop, 1995)

Sırasıyla X_k ve P_k ise k adımında ölçüm değerlerini gördükten sonra tahmini ortalama durumu ve bunun kovaryans değerlerini temsil etmektedir. Z_k ise k adımında ölçüm ortalama değeridir. Ölçüm adımında bulunan $Z_k - H_k X_k$ ifadesi k adımındaki yenilik;

H_k P_k H_k^T + R_k ölçüm tahmin kovaryansı; Kalman kazancı olarak bilinen K_k ise k zaman adımındaki tahminlerin ne kadar düzeltilmesi gerektiğini söyleyen filtre kazancıdır (Laaraiedh, 2012).

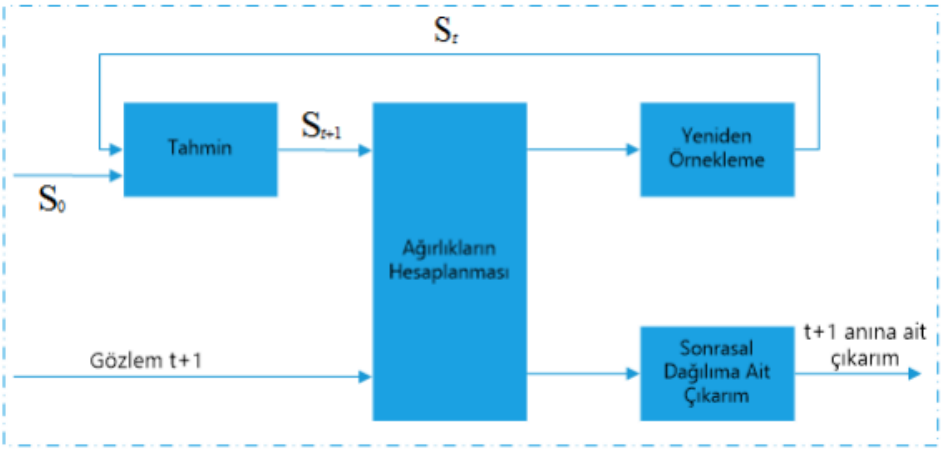
Li ve diğerleri (2010) yaptıkları çalışmada takip ettikleri nesneyi arama alanını azaltarak daha hızlı takip edebilmek için Kalman filtresini kullanır. Var olan nesnenin kayma bilgileri ile hareket modelini oluşturarak nesnenin sonraki görüntüdeki konumunu tahmin etmek için bu filtreden faydalanır.

Bewley ve diğerlerinin (2016) çalışmasında her hedefin durumu x = [u, v, s, r, u˙, v˙, s˙] şeklinde belirlenir ve u, v hedefin merkez noktasının yatay ve düşeydeki piksel konumunu temsil ederken s ölçeği ve r hedef kutusunun en boy oranını temsil eder. Takip edilen hedef tespit ile ilişkilendirildiği takdirde, tespit edilen nesnenin konum kutusu tespitin durumunu güncellemek için kullanılır. Hız bileşenleri Kalman filtresi kullanılarak çözülür. Wojke ve diğerleri (2017) çalışmalarında sabit hız hareketi ve doğrusal gözlem modeli ile standart Kalman filtresini kullanır.

Fan ve diğerleri (2016) Bayes yöntemine dayanan Kalman filtrelemenin nesne sayısı küçük kaldığında çoklu hedef izlemede etkili iken, nesnelerin sayısı arttığında, kimlik bilgisi sayısı daha sık hale geldiği durumda ve yöntemin özyinelemeli yapısı nedeniyle düzeltilmesinin zorlaştığını ifade eder.

Kalman filtresi ile nesne takibinin gerçek zamanlı uygulamalarda kullanımı, düşük işlem gücü gerektirerek hızlı çalışması bakımından uygundur (Yılmaz ve diğerleri, 2006). Ancak doğrusal yapıya sahip olduğundan durum değişkenlerinin Gauss dağılımına sahip olmadığı durumlarda doğrusal olmayan yapıya sahip parçacık filtresi tercih edilmektedir.

Parçacık filtresi doğrusal olmayan sistemlerde bilinen etkinliği ile nesne takibi yöntemlerine avantaj sağlamaktadır (Fan ve diğerleri, 2016). Parçacık filtresi genel akış diyagramı Şekil 2.13'te gösterildiği gibidir.



Şekil 2.13. Parçacık filtresi genel yapısı (Dilmen ve Talu, 2017)

Parçacık filtresi ile nesne takibinde durum tahmini $p(x_t|x_{t-1}, z_{0:t})$ olasılık yoğunluk fonksiyonu ile ifade edilir. Olasılık yoğunluk fonksiyonuyla durum tahmininde x_t durum bilgisini z_t ise gözlem verisini ifade eder. Durum geçiş ve ölçüm modeli için kullanılan fonksiyonlar sırasıyla $x_k = f_k(x_{k-1}, v_{k-1})$ ve $z_k = h_k(x_k, n_k)$ fonksiyonlarıdır. Yöntemlerde kullanılan x_k ifadesi sistemin k zamanındaki durum değerini, v_{k-1} gürültü değerini f_k ise doğrusal olmayan durumu geçiş fonksiyonunu ifade eder. Benzer şekilde n_k gürültü değerini z_k ölçüm modelini h_k ise ölçüm fonksiyonunu ifade eder. Soncul dağılım $p(x_k|z_{1:k})$ iki adımlı yinelemeli hesaplama yöntemiyle belirlenir. Tahmin adımı olan ilk adımda kullanılan formül şu şekildedir (Dilmen ve Talu, 2017):

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})d_{x_{k-1}} \quad (2.10)$$

Güncelleme adımı olan ikinci adımda ise tahmin adımındaki değer ile yeni ölçüm değerleri kullanılır ve soncul dağılım şu şekilde hesaplanır (Dilmen ve Talu, 2017):

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta\left(x_k - x_k^i\right) \quad (2.11)$$

Zhang ve diğerlerinin (2018) çalışmasında takip edilen her nesne birden fazla özelliğe sahip parçalar olarak belirlenir ve her parça korelasyon filtresi ile ilişkilendirilir. Geleneksel yöntemlere kıyasla daha az parçacık kullanılarak nesne takibi işlemi

gerçekleştirilir. Parçacık filtresi ve korelasyon filtresinin birlikte kullanıldığı bu yöntemde nesnenin hedef konumu parçacıkların ağırlıklı ortalaması kullanılarak tahmin edilir.

Diğer taraftan parçacık filtresinin durum parametre sayısı arttıkça hesaplama karmaşıklığı artar. Bu sebeple çok boyutlu durum uzayları için uygun bir tercih olmamaktadır (Vatavu, 2014).

2.4.5.Görünüm Modelleri ile Takip

Görünüm modeli nesne takibinde en önemli konulardan biridir. Poz değişiklikleri, ışık değişimi ile görünümde oluşan farklılıklar, hedef nesnenin yarı kapanma ya da tamamen örtünme gibi durumları nesne takibini zorlaştırmaktadır. Bu gibi durumlarda nesne görünümünün fazla değişmeyeceğini varsayan çalışmalar takip işleminde başarılı olamamaktadır. Görünüm modelinde kullanılacak nesne özniteliklerinin doğru seçimi çalışma başarısını büyük ölçüde etkilemektedir. Seçilen öznitelikler takip edilecek nesne için eşsiz olduğunda takip sırasında nesnelerin birbirleri ile yer değiştirmesi hatası azalacaktır. Gradyan temelli öznitelikler nesne takibi çalışmalarında tercih edilen öznitelikler arasındadır. Gradyan temelli öznitelikler kenar veya biçim bilgileri ile nesne görünüm modeli oluşturanlar ve gradyanların istatistiksel özetini kullanarak görünüm modeli oluşturanlar olarak gruplandırılabilir (Yang ve diğerleri, 2011). Yönelimli eğimlerin histogramı, ölçek değişmez öznitelik dönüşümü (Scale Invariant Feature Transform - SIFT), hızlandırılmış sağlam öznitelikler (Speeded Up Robust Features - SURF) gibi gradyan temelli öznitelikler pek çok nesne takibi çalışmasında tercih edilmiştir. Henriques ve diğerleri (2014) çalışmalarında yönelimli eğimlerin histogramı özniteliğini tercih ederek nesne takibini gerçekleştirmişlerdir. Nesnenin şekli çalışma anında değişiklik gösterdiğinden görünümün çevrim içi güncellenmesi gerekmektedir.

Renk bilgisi de tercih edilen öznitelikler arasındadır. Yaygın olarak kullanılan RGB renk modeli dışında HSV (Hue, Saturation, Value) ve HSI (Hue, Saturation, Intensity) gibi farklı renk uzayları da takip edilecek nesnenin görünümünü modellemek için kullanılmaktadır. Renk bilgisi nesneleri birbirinden ayırt etmede kullanılacak hızlı bir öznitelik olmasına rağmen ışık değişimi meydana geldiği zamanlarda piksel değerleri

değişeceği için takip başarılarını düşürmektedir. Bunlara ek olarak doku özelliği, uzay-zamansal özellikler, birden çok özelliğin birleştirilmesi ve derin öznitelikler de nesne takibi çalışmalarında tercih edilmiştir.

Derin özellikler derin öğrenme modelleri kullanılarak belirlenir. Derin öğrenme kullanılarak yapılan hedef nesne takibi çalışmaları düşük seviyeli özellikler yerine evrişimli sinir ağının katmanlarından elde edilen yüksek seviyeli özellikleri kullanarak hedef modeli oluşturmanın takip başarısını arttırdığını göstermektedir (Ma ve diğerleri, 2015; Qi ve diğerleri, 2016; Zhang ve diğerleri, 2018). Aynı zamanda Walia ve diğerleri (2016) düşük seviyeli tek bir özellik kullanılarak gerçekleştirilen nesne takibinin çeşitli zorlukları bulunduğunu göstermektedir. Bazı çalışmalarda çeşitli özelliklerin birlikte kullanılması ile nesne takibi yöntemleri geliştirilmiştir. Literatürde, bir noktada iyi olan ipuçlarının izleme işlemi sırasında başka bir durumda bozulabileceği iyi bir şekilde tartışılmıştır (Walia ve diğerleri, 2016).

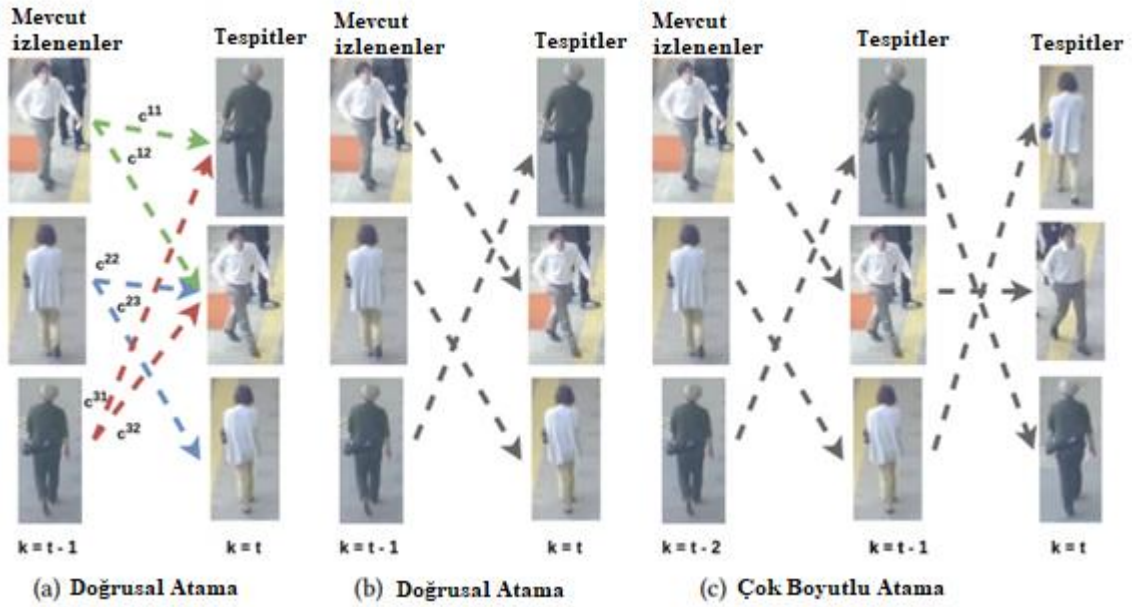
Zhang ve diğerleri (2018) çalışmasında VGGNet-19 evrişimli sinir ağı yapısını kullanarak parçacık filtresi ve korelasyon filtresini beraber uygulayabileceği bir nesne takibi algoritması geliştirmiştir. Ağırlıklandırılmış parçacıklar ile korelasyon işlemi yapılarak nesne takibi gerçekleştirilir. Ma ve diğerleri (2015) evrişimli sinir ağı katmanlarını kullanarak oluşturdukları hedef görünüm modelleri için korelasyon filtresi çalıştırarak hiyerarşik bir yapı elde etmiştir. Son katmandan ilk katmana doğru bu hiyerarşik yapı kullanılarak hedef nesnenin konumu tespit edilmeye çalışılmıştır. Qi ve diğerleri (2016) evrişimli sinir ağı özellikleri ile korelasyon filtresini kullanarak nesne takibi işlemini gerçekleştirmektedir. Evrişimli sinir ağı için üretilen korelasyon filtreleri çevrim içi sınırlama (hedged) algoritması kullanılarak daha güçlü tek bir korelasyon filtresine çevrilir.

2.5. Veri İlişkilendirme

Çoklu nesne takibi çalışmalarında takip edilen nesnelerin konumlarının güncellenmesi sırasında takip edilen nesnelerin tespit edilen nesnelerle ilişkilendirilmesi gerekir ve bu problem veri ilişkilendirme ile çözümlenir. Bir çerçevede tespit edilen nesneler ile önceki çerçevede tespit edilmiş ve takip işlemi uygulanmış nesnelerin ilişkilendirilmesi

çoklu nesne takibi çalışmalarında büyük bir öneme sahiptir. Nesne takibinde olduğu gibi veri ilişkilendirmede de her takip edilen nesne için belirlenen görünüm modelinin eşsiz olması veri ilişkilendirme başarısını arttırmaktadır.

Şekil 2.14'te çoklu nesne takibi çalışmalarında yeni nesneler ile var olan nesnelerin ilişkilendirilmesi gösterilmektedir. İlişkilendirme işlemi sadece son çerçevedeki ölçümleri kullanarak ya da önceki veya sonraki çerçevelerdeki birden fazla ölçümü kullanarak gerçekleştirilebilir. Bunların dışında gerçek zamanlı olmayan nesne takibi çalışmalarında tüm çerçevelerdeki ölçüm değerleri çevrim dışı kullanılarak da ilişkilendirme gerçekleştirilebilir. Takip edilen nesnelerin birbirine yakın olması ya da eksik tespitlerin olması durumlarında son ölçümlerden bir veya birkaçının kullanılması tercih edilmektedir. Ancak hesaplama yükü fazla olduğu için uygulanması da zordur (Emami ve diğerleri, 2020). Buna karşın doğruluk ve hassasiyet değerleri tek çerçevedeki ölçümlerin kullanıldığı yaklaşıma kıyasla yüksek, nesnelerin birbiri ile karıştırılması olasılığı ve kaçırılan hedef nesne sayıları daha düşüktür (Poore ve Gadaleta, 2006).



Şekil 2.14. Çoklu nesne takibinde veri ilişkilendirme (Emami ve diğerleri, 2020)

Literatürde çeşitli veri ilişkilendirme yöntemleri mevcuttur. Doğrusal atama (Linear assignment), küresel en yakın komşu (global nearest neighbor), çoklu hipotez izleme (multi-hypothesis tracking), minimum maliyetli akış çözümü (min-cost flow solution), kesişimin birleşime oranı (IOU) yöntemleri veri ilişkilendirme yöntemlerinden bazılarıdır.

2.5.1.Doğrusal Atama

Önceki çerçevede M kadar takip edilen nesne ve mevcut çerçevede N kadar tespit edilmiş nesne olduğunu varsayıldığı takdirde iki çerçevedeki nesneleri birbirleri ile ilişkilendirmek için MxN boyutlu C matrisi kullanılır. C matrisinin her bir elemanı, Şekil 2.14'te ifade edilen c_k^{ij}, k zamanındaki j ölçümünün i nesnesine atanmasındaki maliyet hesabını ifade eder. Amaç toplam maliyeti en aza indirecek şekilde ölçümlerin nesnelere atanmasını sağlamaktır. Hesaplamalarda bir nesneye atanan ölçümü temsil etmek için ikili karar değişkenleri $x^{ij} \in \{0,1\}$ kullanılarak 0-1 tamsayı programlamayla çözüm gerçekleştirilir. İkili atama matrisi olan X için j=1…n'ye kadar olan x^{ij} değerlerinin toplamı ve i=1…m'ye kadar olan x^{ij} değerlerinin toplamı 1 olacak şekilde şu formül kullanılır.

$$min \sum_{\mathrm{i}=1}^{\mathrm{m}} \sum_{j=1}^{n} c^{ij} x^{ij} \tag{2.12}$$

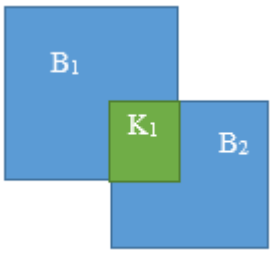
Bu şekilde bir formül ile tüm satır ve sütunlardaki değerlerin toplamının 1 olması sağlanır. Toplam atama sayısı min(m,n)! olmasına rağmen polinom zamanda çözüm sunan algoritmalar da mevcuttur. Macar algoritması (Hungarian Algorithm) $O(n^3)$ karmaşıklık zamanı ile bu probleme çözüm sunmaktadır. Fazla karmaşıklık, yanlış tespitler, yarı ya da tam örtünme, tespitlerin kaçırılması gibi durumlarda çalışma başarısı yüksek değildir (Emami ve diğerleri, 2020).

2.5.2.Kesişimin Birleşime Oranı

IOU hesabı iki çerçevenin kesişim kümelerinin birleşim kümesine olan oranıdır ve IOU = K1 / (K1+B1+B2) şeklinde ifade edilir. Bu metrik Şekil 2.15'te yeşil alanın mavi ve yeşil alanların bütününe oranı ile ifade edilmektedir.

İki çerçevenin birbiri ile tam örtüşmesi durumunda bu oran 1 iken çerçevelerin örtüşmesi azaldıkça bu oran da düşer. Çerçevelerin kesişmediği durumda ise bu oran 0 olur. Nesne takibi çalışmalarında çerçeveler arası geçişlerde nesnelerin konumunun fazla değişmeyeceği varsayıldığından literatürde IOU'nun ilişkilendirme amacıyla kullanıldığı çalışmalar mevcuttur. Hızlı çalışan bir yöntem olmasına karşın nesnelerin yarı ya da tam örtünmesi gibi durumlarda çalışma başarısı düşmektedir.

Çerçevelerin karşılaştırılması dışında nesnelerin merkez konumlarını dikkate alarak veri ilişkilendirme yapan nesne takibi çalışmaları da mevcuttur. Bu çalışmalarda amaçlanan ardışık iki çerçevedeki nesnelerin merkez konumları arasındaki farkları alıp en kısa mesafeye sahip nesnelerin ilişkilendirilmesidir. Bu yöntem de kesişimlerin birleşime oranı yöntemi gibi hızlı olmasına karşın aynı zorlu durumlarda düşük çalışma başarılarına sahiptir.



Şekil 2.15. Kesişimin birleşime oranı

2.5.3.Macar Algoritması

Optimizasyon algoritması olan Macar algoritması atama işlemini polinom zamanda çözer. Kuhn (1955) tarafından geliştirilen ve Kuhn-Munkres olarak da bilinen yöntem

farklı kişiler tarafından optimize edilmiş ve yöntemin çalışma karmaşıklığı $O(n^3)$'e kadar düşürülmüştür.

Algoritma dört adımdan oluşmaktadır. İlk iki adım tek sefer çalıştırılırken sonraki iki adım optimum atama sonuçlanana kadar yinelemeli şekilde çalıştırılır. En düşük satırları çıkarma adımı olan ilk adımda her satır için en düşük değer bulunur ve o satırdaki her öğeden bu en düşük değer çıkartılır. En düşük sütunu çıkarma adımı olan ikinci adımda her sütundaki en düşük değer bulunarak bu değer o sütundaki tüm değerlerden çıkartılır. Tüm sıfırların en az sayıda satırla kapatılması adımı olan üçüncü adımda minimum sayıda yatay ve dikey çizgi kullanılarak ortaya çıkan matristeki tüm sıfırlar örtülür. Eğer kullanılan minimum yatay ve dikey çizgi sayısı toplam sütun sayısı kadar ise atamalar optimize edilmiş anlamına gelir ve algoritma durur. Eğer gerekli sayı az ise dördüncü adıma geçilir. Ek sıfırlar oluşturma adımı olan dördüncü adımda önceki adımdaki çizgi ile kapanmamış en küçük değer bulunarak kaplanmamış tüm değerlerden bu sayı çıkartılır ve iki kere kaplanan tüm değerlere bu değer eklenir. Algoritma sonlanana kadar üçüncü ve dördüncü adımlar yinelemeli olarak çalıştırılır ("The Hungarian algorithm", 2021).

Örneğin bir atama probleminde 4 iş (J1, J2, J3, J4) ve 4 çalışan (W1, W2, W3, W4) bulunmakta ve her işin yalnız bir çalışana atanması gerekmektedir. Bu problemi Macar algoritmasının en az maliyetle çözmesi için dört temel adım takip edilir. Şekil 2.16'da her iş için atanan bir çalışanın maliyeti gösterilmektedir.

	J1	*J2*	*J3*	*J4*
W1	82	83	69	92
W2	77	37	49	92
W3	11	69	5	86
W4	8	9	98	23

Şekil 2.16. Macar algoritması örnek atama maliyetleri ("The Hungarian algorithm", 2021)

Şekil 2.17'de Macar algoritmasının ilk adımı gösterilmektedir. İlk satırdaki değerlerden o satırdaki en düşük değer olan 69 çıkarılmıştır. İkinci satırdaki değerlerden o satırdaki

en düşük değer olan 37 çıkarılmıştır. Üçüncü satırdaki değerlerden o satırdaki en düşük değer olan 5 çıkarılmıştır. Dördüncü satırdaki değerlerden o satırdaki en düşük değer olan 8 çıkarılmıştır.

	J1	*J2*	*J3*	*J4*	
W1	13	14	0	23	(-69)
W2	40	0	12	55	(-37)
W3	6	64	0	81	(-5)
W4	0	1	90	15	(-8)

Şekil 2.17. Macar algoritması birinci adım ("The Hungarian algorithm", 2021)

Şekil 2.18'de Macar algoritmasının ikinci adımı gösterilmektedir. İlk üç sütunda en küçük değer 0 olduğu için çıkarma işlemi uygulanmamıştır. Dördüncü sütunun en küçük değeri olan 15 o sütundaki tüm değerlerden çıkarılmıştır.

	J1	*J2*	*J3*	*J4*
W1	13	14	0	8
W2	40	0	12	40
W3	6	64	0	66
W4	0	1	90	0
				(-15)

Şekil 2.18. Macar algoritması ikinci adım ("The Hungarian algorithm", 2021)

Şekil 2.19'da Macar algoritmasının üçüncü adımı gösterilmektedir. Tüm sıfır değerlerini örten yatayda ve düşeyde minimum 3 çizgi çizilmiştir. Üç değeri matris boyutu olan 4'ten küçük olduğu için dördüncü adıma geçiş yapılmıştır.

	J1	*J2*	*J3*	*J4*	
W1	13	14	**0**	8	
W2	**40**	**0**	**12**	**40**	x
W3	6	64	**0**	66	
W4	**0**	**1**	**90**	**0**	x
			x		

Şekil 2.19. Macar algoritması üçüncü adım ("The Hungarian algorithm", 2021)

Şekil 2.20'de Macar algoritmasının dördüncü adımı gösterilmektedir. Örtülmeyen değerler arasında en küçük değer olan 6 yine örtülmeyen değerlerden çıkarılırken iki kere örtülen değerlerin hepsine eklenmiştir. Tekrar üçüncü adıma geçiş yapılmıştır.

	J1	*J2*	*J3*	*J4*
W1	7	8	0	2
W2	40	0	18	40
W3	0	58	0	60
W4	0	1	96	0

Şekil 2.20. Macar algoritması dördüncü adım ("The Hungarian algorithm", 2021)

Şekil 2.21'de Macar algoritmasının üçüncü adımı tekrar gösterilmektedir. Tüm sıfır değerlerini örten yatayda ve düşeyde minimum 4 çizgi çizilmiştir. Minimum gereken çizgi sayısı matris boyutuna eşit olduğu için algoritma durmuştur.

	J1	*J2*	*J3*	*J4*	
W1	**7**	**8**	**0**	**2**	x
W2	**40**	**0**	**18**	**40**	x
W3	**0**	**58**	**0**	**60**	x
W4	**0**	**1**	**96**	**0**	x

Şekil 2.21. Macar algoritması üçüncü adım tekrarı ("The Hungarian algorithm", 2021)

Şekil 2.22'de optimum iş ve çalışan atamaları gösterilmektedir. Sonuç matrisine göre W1 çalışanı J3 işine, W2 çalışanı J2 işine, W3 çalışanı J1 işine, W4 çalışanı i ise J4 işine atanmıştır.

	J1	*J2*	*J3*	*J4*
W1	7	8	**0**	2
W2	40	**0**	18	40
W3	**0**	58	0	60
W4	0	1	96	**0**

Şekil 2.22. Macar algoritması optimum atama ("The Hungarian algorithm", 2021)

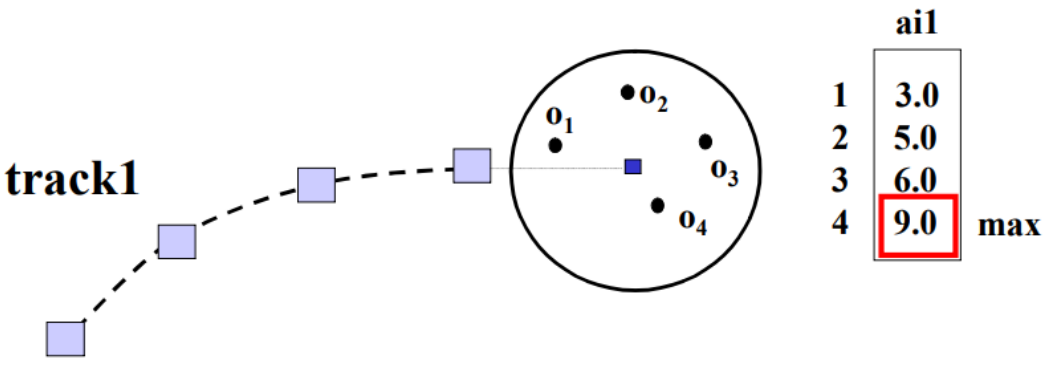
Şekil 2.23'te de optimum iş ve çalışan atamalarının maliyetleri gösterilmektedir. Sonuç matrisine göre W1'e J3 atama maliyeti 69, W2'ye J2 atama maliyeti 37, W3'e J1 atama maliyeti 11, W4'e J4 atama maliyeti 23 ve toplam maliyet ise 69 + 37 + 11 + 23 = 140 olmuştur.

	J1	J2	J3	J4
W1	82	83	**69**	92
W2	77	**37**	49	92
W3	**11**	69	5	86
W4	8	9	98	**23**

Şekil 2.23. Macar algoritması optimum atama maliyetleri ("The Hungarian algorithm", 2021)

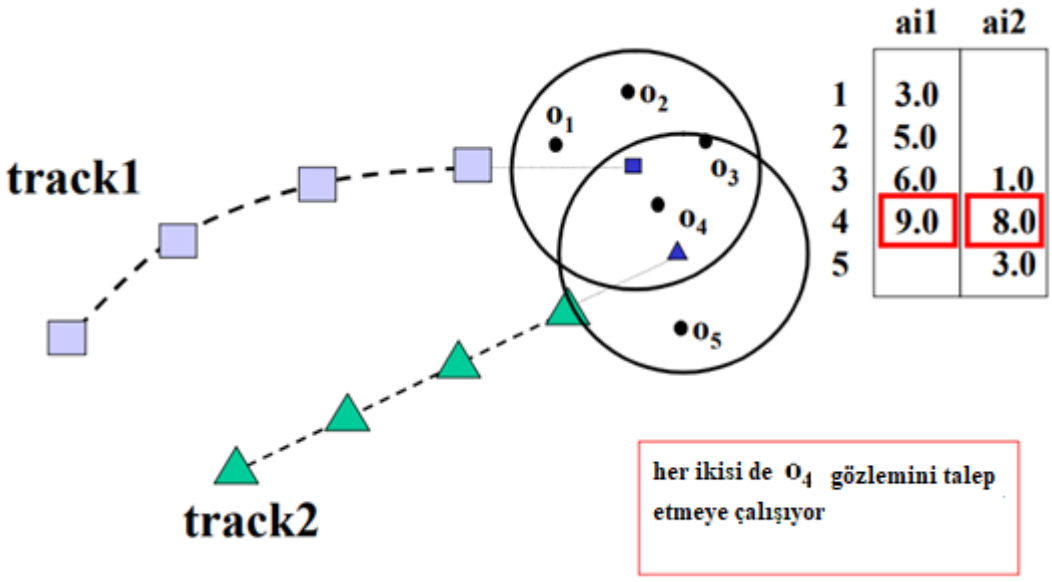
2.5.4.Küresel En Yakın Komşu

Küresel en yakın komşu (Global Nearest Neighbour - GNN) algoritması her adımda en olası tek hipotezi bulmaya çalışır. Takip edilen nesnelerin hepsi değerlendirilerek ilişkilendirme için en iyi olan seçilir. Şekil 2.24'te takip edilen track1 nesnesi için ilişkilendirmede tüm ihtimaller değerlendirilerek en yüksek benzerlik olan 9'a sahip 4 numaralı gözlem seçilmektedir. Ancak her nesne için atama işlemleri bağımsız şekilde gerçekleştirildiğinde aynı gözlem değerlerinin farklı nesnelere atanması problemi ortaya çıkabilmektedir (Collins, 2012).



Şekil 2.24. Küresel en yakın komşu örnek atama (Collins, 2012)

Şekil 2.25'te küresel en yakın komşu yöntemi ile veri ilişkilendirmede karşılaşılan aynı gözlem değerlerinin farklı nesnelere atanması problemi gösterilmektedir. Takip edilen nesneler track1 ve track2 için küresel en yakın komşu sırasıyla 9.0 ve 8.0 değerlerini veren 4 numaralı gözlemdir. Her iki nesne için aynı gözlem değeri seçildiğinde atama problemi oluşmaktadır. (Collins, 2012).



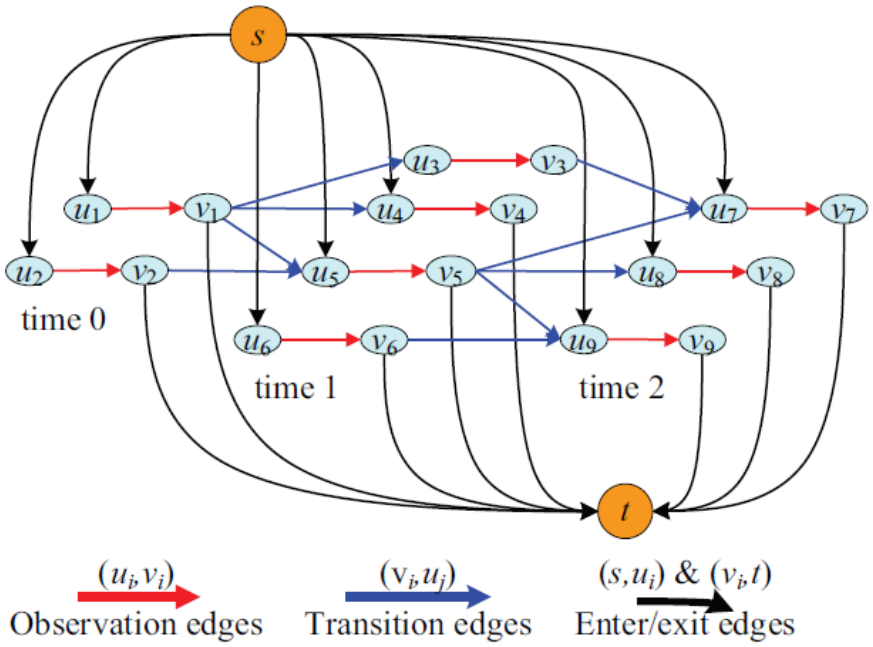
Şekil 2.25. Küresel en yakın komşu atama problemi (Collins, 2012)

2.5.5.Çizge Temelli Yaklaşımlar

Çizge yapılarını kullanarak ve çizgelerin düğüm noktalarını takip edilen nesnelerin konumlarını temsil etmek için kullanarak çoklu nesne takibine çözüm arayan çalışmalardır. Literatürde çizge temelli birçok yöntem mevcuttur. Ağ akışı ile genelleştirilmiş minimum klik problemi (Generalized Minimum Clique Problem - GMCP) ve küresel optimal genelleştirilmiş maksimum çoklu klik problemi (Generalized Maximum Multi Clique Problem - GMMCP) optimizasyon yöntemleri çizge temelli veri ilişkilendirme yöntemlerinden bazılarıdır.

Şekil 2.26'da 3 zaman adımı ve 9 gözlem içeren bir maliyet akışı ağı örneği gösterilmektedir. Çizgedeki düğümler algılama yanıtları veya izleme işaretleridir. Akışlar, iki düğümü birbirine bağlayan bir gösterge olarak modellenmektedir. Akış

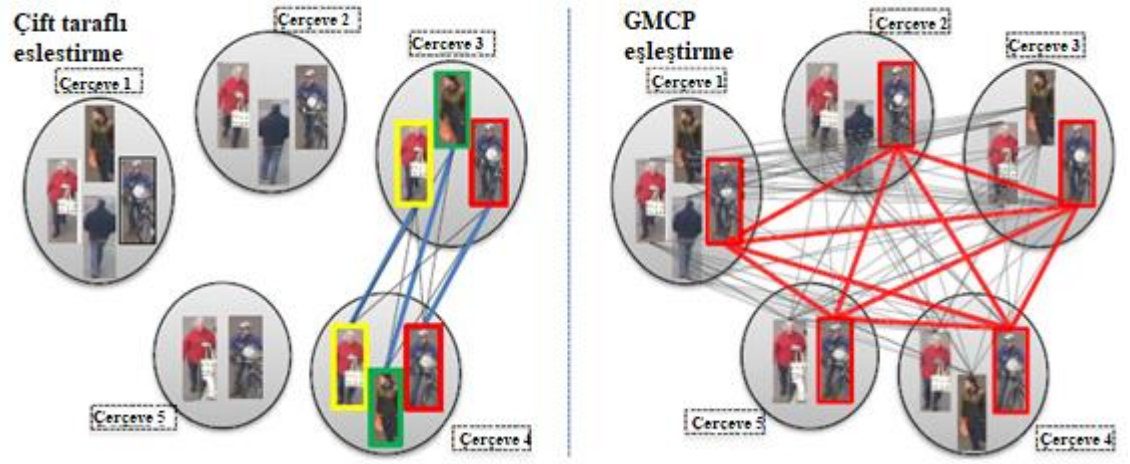
dengesi gereksinimini karşılamak için, grafiğe bir yörüngenin başlangıcına ve sonuna karşılık gelen bir kaynak düğüm (S) ve bir havuz düğümü (t) eklenmektedir. Grafikteki bir yörünge bir akış yoluna karşılık gelmektedir. Kaynak düğümden havuz düğümüne aktarılan toplam akış, yörünge sayısına eşittir ve geçiş maliyeti, tüm ilişkilendirme hipotezlerinin negatif log olasılığıdır. Global olarak en uygun çözüm çeşitli algoritmaların kullanımı ile polinom zamanda bulunabilir (Luo ve diğerleri, 2020). Literatürde çözüm için sunulmuş pek çok algoritma mevcuttur. Bunlardan bazıları k-en kısa yol, açgözlü algoritmalar, minimum akış algoritması, doğrusal programlama, gevşetilmiş Lagrange (Lagrangian relaxation) ve destekli yeniden etiketleme (push relabel) bunlardan bazılarıdır.



Şekil 2.26. 3 zaman adımı ve 9 gözlem içeren bir maliyet akışı ağı örneği (Zhang ve diğerleri, 2008)

Genelleştirilmiş minimum klik çizgeleri (Zamir ve diğerleri, 2012) çalışmasında veri ilişkilendirme için hareket ve görünüm modelini küresel şekilde birleştiren bir yöntem geliştirilmiştir. Tüm çerçeveler veri ilişkilendirmeye dâhil edilmiş ancak nesnelerin hepsi aynı anda ele alınmayarak her seferinde tek bir nesneye odaklanılmıştır. Şekil 2.27'de GMCP yönteminin çift taraflı eşleştirme (bipartite matching) yöntemi ile farkı

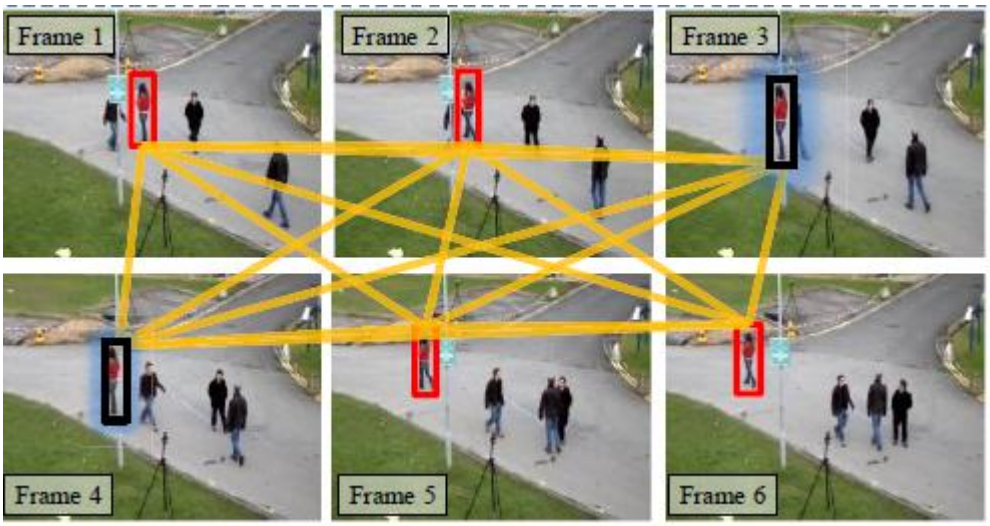
ifade edilmiştir. Gri kenarlar giriş çizgesini ve renkli kenarlar optimize edilmiş alt çizgeyi temsil etmektedir. Çift taraflı eşleştirme yönteminde sınırlı geçici pencerede tüm nesneler eşleştirilirken, GMCP yönteminde odak nesne haricindeki nesnelerin tümü örtük olarak sürece dâhil edilir ve tüm çerçeveler boyunca her seferinde bir nesne için eşleme yapılır.



Şekil 2.27. Çift taraflı eşleştirme ve GMCP karşılaştırması (Zamir ve diğerleri, 2012)

Giriş videosu ilk adımda çerçevelere ayrıştırılıp her çerçeve içindeki nesneler parça tabanlı nesne tespiti ile bulunmaktadır. Daha sonra giriş videosu her biri f kadar çerçeve içeren s tane parçaya ayrılıp her parça için GMCP yöntemi ile nesnelerin yörüngesi belirlenir. Veri ilişkilendirme adımının girdisi düğüm sayısı V, kenar sayısı E ve kenarların ağırlığı w iken çizge G (V, E, w) ile ifade edilir. Çalışmada örtünmelerin üstesinden gelmek için minimum maliyetli varsayım düğümleri eklenmiştir. Şekil 2.28'de 3 numaralı ve 4 numaralı çerçevelerde (Frame 3, Frame 4) yaya etrafında çizilen siyah kutu varsayım düğümü ekleme işlemini temsil etmektedir.

Takip edilen nesnenin uzun bir zaman dilimindeki hareketini modellemek zordur ancak kısa bir zaman diliminde sabit hızlı ya da ivmeli bir hareketi modellenebilir. Çalışmada hareket modelinin küresel veri ilişkilendirmeye dâhil edilmesi hiyerarşik bir yapıda gerçekleştirilmektedir. Çizgedeki ağırlıklar nesneler arasındaki hareket ve görünüm modelinin benzerlik oranına göre belirlenmektedir.



Şekil 2.28. GMCP varsayım düğümü ekleme (Zamir ve diğerleri, 2012)

Dehghan ve diğerleri (2015) çalışmalarında veri ilişkilendirme yöntemi kullanarak gerçek dünyadaki izleme senaryosuna daha yakın bir model düşünmüştür. Önerilen GMMCP yöntemi ardışık gelen çerçeveler üzerindeki hareketli nesneler arasında ilişki kurarak hedef nesne için en uygun yol haritasını çıkartmaya çalışmıştır. Ardışık olarak alınan tüm çerçevelerdeki tüm hedef nesneler ile bir ağ yapısı oluşturulmuştur. Nesne takibi sırasında sıklıkla görülen hedef örtünme veya kaybolma durumları için belirli bir süre boyunca kaybolan hedefler yerine sahte hedefler koyularak hatalar en aza indirilmeye çalışılmıştır. Çalışmanın temeli GMCP yöntemine dayanmaktadır.

2.5.6.Derin Özellikler ile Basit Çevrim İçi ve Gerçek Zamanlı Takip

Bewley ve diğerleri (2016) basit çevrim içi ve gerçek zamanlı izleme (SORT) yöntemi geliştirerek çoklu yaya takibini gerçekleştirmiştir. Görüntüdeki yayaların tespiti için Faster Region CNN (Ren ve diğerleri, 2017) kullanılmıştır. Her nesnenin görüntüler arasındaki yer değişimini modellemek için diğer nesnelerden ve kamera hareketinden bağımsız doğrusal sabit bir hız modeli kullanılmıştır. Her nesnenin durumu nesnenin görüntüdeki merkez piksel konumunun yatay konumu u, nesnenin görüntüdeki merkez piksel konumunun dikey konumu v, ölçek s, hedef çevreleyici dikdörtgenin en boy

oranı r ve u, v, s değerlerinin hızları u˙, v’, s˙ iken yer değişim modeli x = [u, v, s, r, u˙, v˙, s˙]^T şeklinde ifade edilir. En boy oranının (r) sabit olduğu varsayılmaktadır. Takip edilen nesne ile tespit edilen nesne ilişkilendirildiği zaman hedef durumun güncellenmesi Kalman filtresi ile gerçekleştirilir. Takip edilen hedef ile tespitlerin eşleşmediği durumda ise hedefin durumu doğrusal hız modeli ile güncellenir.

Aynı veri ilişkilendirme çalışmasında her takip edilen hedef ve tespit çiftine ilişkin bir maliyet matrisi oluşturulur ve her bir çift için kesişimlerin birleşime oranı ile benzerlik hesaplaması gerçekleştirilir. Daha sonra en iyi atamaların hesaplanması için Macar algoritması kullanılır. Ayrıca çalışmada iki farklı nesne tespiti yönteminin karşılaştırmasını yaparak nesne tespitinin nesne takibindeki başarıyı etkilediğini göstermişlerdir. Çizelge 2.1’de ACF (Dollar ve diğerleri, 2014), FrRCNN (ZF) (Zeiler ve Fergus, 2014), FrRCNN(VGG16) (Simonyan ve Zisserman, 2014) nesne tespit algoritmalarının başarı oranları ve nesne takibi üzerindeki etkisi gösterilmektedir. Buna göre nesne tespiti başarı oranı en yüksek yöntem olan FrRCNN (VGG16) nesne takibinde de en yüksek başarı oranına sahiptir (Bewley ve diğerleri, 2016).

Çizelge 2.1. Nesne tespiti yönteminin nesne takibi üzerindeki etkisi

Takip Algoritması	**Tespit Algoritması**	**Nesne Tespiti**		**Nesne Takibi**	
		Recall	**Precision**	**ID Sw**	**MOTA**
SORT	ACF	33.6	65.7	224	15.1
	FrRCNN(ZF)	41.3	72.4	347	24.0
	FrRCNN(VGG16)	49.5	77.5	274	34.0

Çizelge 2.2 Zheng ve diğerlerinin (2016) çalışmasında 1261 yayanın 1.100.000’den fazla görüntüsünü içeren veri seti ile eğitilmiş evrişimli sinir ağı modelinin katmanlarını göstermektedir. Nesnelerin derin özelliğini temsil eden 128x1 boyutlu vektör Dense 10

katmanında hesaplanmaktadır. Sonraki katmanda kosinüs metriği ile uyumlu olması için normalleştirme yapılmaktadır.

Çizelge 2.2. DeepSORT evrişimli sinir ağı modeli

Adı	Parça (Patch) Boyutu / Kaydırma (Stride)	Çıktı Boyutu
Conv 1	3 x 3 / 1	32 x 128 x 64
Conv 2	3 x 3 / 1	32 x 128 x 64
MaxPool 3	3 x 3 / 2	32 x 64 x 32
Residual 4	3 x 3 / 1	32 x 64 x 32
Residual 5	3 x 3 / 1	32 x 64 x 32
Residual 6	3 x 3 / 2	64 x 32 x 16
Residual 7	3 x 3 / 1	64 x 32 x 16
Residual 8	3 x 3 / 2	128 x 16 x 8
Residual 9	3 x 3 / 1	128 x 16 x 8
Dense 10		128
Batch ve L2 normalleştirme		128

Wojke ve diğerleri (2017) SORT algoritmasına görünüm modelini entegre ederek bu algoritmayı geliştirmişlerdir. Bu entegrasyon ile nesnelerin uzun süreli takibi gerçekleştirilmiş ve örtünme durumlarındaki başarı oranı arttırılmıştır. Takip senaryosunda her nesne, nesnenin merkez konumunun yataydaki ve düşeydeki koordinatları (u, v), en boy oranı (y), yükseklik (h) ve bunların çerçeve koordinatlarındaki hızlarını ifade eden 8 boyutlu durum vektörü ile ifade edilmiştir. Standart Kalman filtresi ile doğrusal ve sabit hız modeli kullanılmaktadır. Sınır koordinatları nesne durumunun doğrudan ölçümleri olarak alınmaktadır. Hareket bilgilerini dahil etmek için tahmin edilen Kalman durumu ile yeni gelen ölçümler arasındaki Mahalanobis mesafesi kullanılmaktadır. Ancak kamera hareketinin görüntüde hızlı yer değiştirmelere yol açtığı durumlarda ve örtünme durumlarıyla başetmede Mahalanobis mesafesi yetersiz kalmaktadır. Bu nedenle atama işlemi için bir ek metrik tanımlanmıştır. Ek metrik hedefler ve tespitler arasındaki en kısa kosinüs mesafesini hesaplar. Mahalanobis mesafesi kısa periyotlardaki ölçümlerde olası nesne konumları arasında harekete dayalı bilgi sağlamaktadır. Kosinüs mesafesi ise görünüm bilgilerini dikkate alarak nesnelerin uzun süreli örtünmesi durumlarında hareketin ayırt

edici olamamasının eksikliğini kapatmaktadır. Çalışmada iki mesafe de belirli bir ağırlık atanmış şekilde birlikte kullanılmaktadır.

2.6. Çoklu Nesne Takibi Kıyaslama

Literatürdeki çoklu nesne takibi çalışmalarının performans ve başarı karşılaştırması için ortak bir kıyaslama gerekmektedir. Başarı ölçümleri ve karşılaştırmalar için yaygın olarak kullanılan kıyaslamalardan biri olan çoklu nesne takibi (MOT) kıyaslama çeşitli ortamlarda sabit ve hareketli kameralardan elde edilmiş veri setleri bulundurmaktadır. MOT eğitim ve test veri setlerindeki videolarda bulunan çerçeve sayıları, toplamda takip edilen nesne sayıları ve toplamda tespit edilen kişi sayıları Çizelge 2.3'te gösterilmektedir.

Çizelge 2.3. MOT veri setindeki videolara ilişkin bilgiler

Veri Seti Adı	Çerçeve Sayısı	Takip Edilen Nesne Sayısı	Tespit Edilen Nesne Sayısı
2D MOT 15 – Eğitim Seti	5500	500	39905
2D MOT 15– Test Seti	5783	721	61440
MOT 16 – Eğitim Seti	5316	517	110407
MOT 16 – Test Seti	5919	759	182326
MOT 17 – Eğitim Seti	15948	1638	336891
MOT 17 – Test Seti	17757	2355	564228
MOT 20 – Eğitim Seti	8931	2332	1336920
MOT 20 – Test Seti	4479	1501	765465

MOT için kullanılan değerlendirme ölçütleri ve bunların açıklamaları Çizelge 2.4'te gösterilmektedir. Not sütununda bulunan ↑ sembolü ilgili ölçütün değeri büyük ise performansın daha başarılı, ↓ sembolü ise ilgili ölçütün değeri küçük ise performansın daha başarılı olduğunu ifade etmektedir (Luo ve diğerleri, 2020).

Çizelge 2.4. MOT değerlendirme ölçütlerine genel bakış

Değerlendirme Ölçütü	Açıklama	Beklenti
Recall	Doğru eşleştirilmiş tespitlerin gerçekte eşleşmesi gereken tespitlere oranı	↑
Precision	Doğru eşleştirilmiş tespitlerin toplam tespitlere oranı	↑
FAF/FPPI	Bir sekans üzerinden ortalaması alınan çerçeve başına yanlış pozitif sayısı	↓
MODA	Kaçırılan tespitleri ve FAF'ı birleştirir	↑
MODP	Doğru pozitifler ve taban gerçeklik arasındaki ortalama örtüşme	↑
MOTA	Yanlış negatifleri, yanlış pozitifleri ve uyumsuzluk oranını birleştirir	↑
IDs	İzlenen bir yörüngenin eşleşen kimlik kesinliğini kaybetme sayısı	↓
MOTP	Doğru pozitif yörüngeler üzerinden ortalaması alınan taban gerçeklikle örtüşme oranı	↑
TDE	Taban gerçeklik ve tespit sonucu arasındaki mesafe	↓
OSPA	Taban gerçeklik ve tespit sonuçları arasındaki uzamsal mesafe ve önem değeri	↓
MT	Tespit çıktısının uzunluğunun taban gerçeklik yörüngeyi %80'den daha fazla kaplaması oranı	↑
ML	Tespit çıktısının uzunluğunun taban gerçeklik yörüngeyi %20'den daha az kaplaması oranı	↓
PT	1.0 – (MT + ML)	-
FM	İzleme sonucunda bir doğru pozitif yörüngenin kesintiye uğrama sayısı	↓
RS	Kısa süreli örtünmeden doğru bir şekilde kurtulan yörüngelerin oranı	↑
RL	Uzun süreli örtünmeden doğru bir şekilde kurtulan yörüngelerin oranı	↑

Hata matrisi (confusion matrix) olarak da bilinen karmaşıklık matrisi sınıflandırma çalışmalarında doğru ve yanlış sonuçların hesaplanması için kullanılan M×M boyutlu bir matristir. Karmaşıklık matrisi Çizelge 2.5'te belirtilen şu ölçütler üzerinden oluşturulur.

- Doğru pozitif (TP) ölçütü pozitif tahmin değeri ve pozitif gerçek değere sahip tahminler için doğru tahminleri ifade eder.

- Yanlış pozitif (FP) ölçütü pozitif tahmin değeri ve negatif gerçek değere sahip tahminler için yanlış tahminleri ifade eder.
- Yanlış negatif (FN) ölçütü negatif tahmin değeri ve pozitif gerçek değere sahip tahminler için yanlış tahminleri ifade eder.
- Doğru negatif (TN) ölçütü negatif tahmin değeri ve negatif gerçek değere sahip tahminler için doğru tahminleri ifade eder.

Çizelge 2.5. Karmaşıklık matrisi

Gerçek Değer	Tahmin Değeri		
		Pozitif (1)	Negatif (0)
	Pozitif (1)	Doğru Pozitif (TP)	Yanlış Negatif (FN)
	Negatif (0)	Yanlış Pozitif (FP)	Doğru Negatif (TN)

TP, FP, FN, TN ölçütleri başarı ölçümü için farklı ölçütlerde kullanılmaktadır. Bu ölçütlerden bazıları doğruluk (accuracy), duyarlılık (recall), kesinlik (precision) ve F1 puanı (F1 score) olup yüksek başarılar için tümünün yüksek değerler olarak hesaplanması beklenmektedir. Denklem (2.13)'te verilen doğruluk ölçütü tahmin değerlerinden doğru olarak belirlenenlerin tüm veri kümesine oranı ile hesaplanmaktadır. Çalışmanın başarı değeri hesaplanırkan doğruluk ifadesinin tek başına kullanılması yetersiz kalacaktır. (2.14)'teki duyarlılık ölçütü gerçek değerleri doğru olan örneklerden kaçının doğru olarak tahmin edildiğini hesaplamaktadır. (2.15)'te verilen kesinlik ölçütü tahmin sırasında doğru olarak belirlenen verilerin kaçının gerçekten doğru olduğunu hesaplamaktadır. F1 puanı duyarlılık ve kesinlik ölçütlerinde bir denge oluşturmak için kullanılmaktadır. Verinin eşit dağılmadığı durumlarda F1 puanı önemli bir başarı ölçütü olmaktadır. F1 puanına ait formül (2.16)'daki gibidir.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

$$F1 = 2\,\frac{Precision * Recall}{Precision + Recall} \tag{2.16}$$

Çoklu nesne takibi çalışmalarındaki başarı ölçümlerini ortak kıyaslamada kullanılan çoklu nesne takip doğruluğu (MOTA) ve çoklu nesne takip kesinliği (MOTP) ölçütlerine ait formüller sırasıyla (2.17) ve (2.18)'de verilmektedir (Dendorfer ve diğerleri, 2020; Leal-Taixé ve diğerleri, 2015; Milan ve diğerleri, 2016). Denklemlerde t çerçeve indeksini, GT taban gerçeklik sayısını, $d_{t,i}$ çerçevedeki hedef nesnenin çevreleyici kutusunun taban gerçeklik ile örtüşme oranını ve c_t ise çerçevedeki eşleşme sayısını ifade etmektedir.

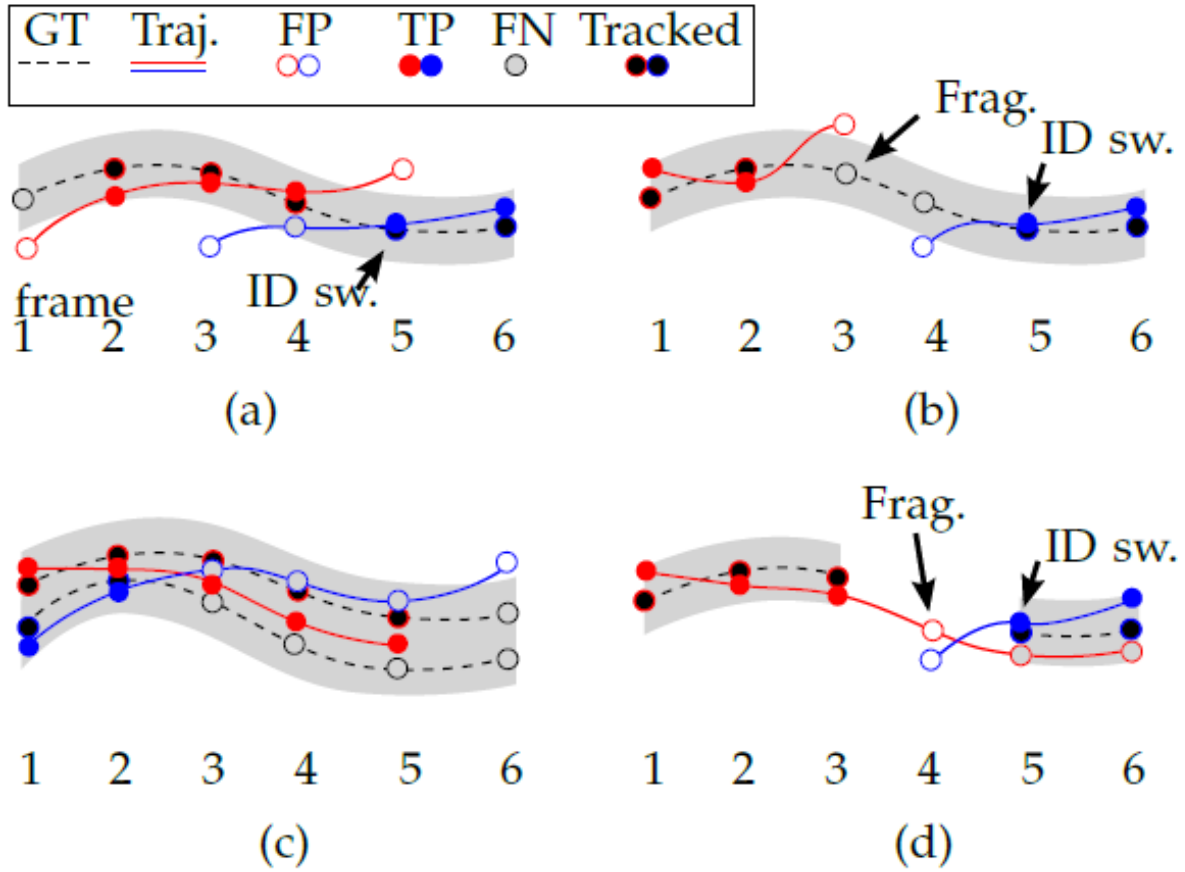
$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{2.17}$$

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{2.18}$$

Üç hata kaynağını birleştiren MOTA kıyaslama ölçütü çalışmalarda performans hesaplama için sıklıkla kullanılmaktadır. Bir konumlandırma kesinliği ölçütü olan MOTP ise tespit sonucu elde edilen tüm doğru pozitifler ile bunlara karşılık gelen taban gerçeklik hedefler arasındaki ortalama örtüşme oranını ifade eder.

Şekil 2.29'da tespit ve hedef atamalarını gösteren 4 durum verilmektedir. Şekil 2.29 (a)'da kimlik belirleyicilerin yer değiştirmesi gösterilmektedir. Önceden atanan kırmızı olan yörüngeden mavi olana geçildiğinde önceki kimlik numarası (ID) değişerek farklı bir kimlik numarası oluşmaktadır. Takip edilen nesneler arasında örtünme, yarı örtünme ve kapanma gibi durumlar meydana geldiğinde yüksek olasılıkla bu türden hatalar oluşmaktadır. Kapanma durumu ortadan kalktığı zaman nesneler eski kimlik bilgilerini kaybederek birbirlerinin kimlik bilgilerini alabilmektedir. IDs ile ifade edilen kıyaslama

ölçütü bu gibi durumlar için hesaplanmaktadır. Bu ölçütün düşük olması çalışmanın başarı oranının yüksek olduğu anlamına gelmektedir. Şekil 2.29 (b)'de parçalanma (fragmentation) başarı ölçütü anlatılmaktadır. Hedef 1'den 2'ye kadar olan çerçeveler boyunca doğru takip edilmiş ancak 3. çerçevedeki yanlış takip sonrasında yörünge kesintiye uğramıştır. Daha sonra 4. çerçevede yeniden başlayan takip için oluşan kimlik numarası ilk çerçevedeki kimlik numarası ile aynı değildir. Mavi ile gösterilen yörüngede kimlik değişimi (IDs) oluşmuştur. Şekil 2.29 (c)'de 1 numaralı çerçevede başlayan yörüngeler son çerçevede 5 kaçırılmış hedef (FN) ve 4 yanlış pozitif (FP) ile sonuçlanmıştır. Şekil 2.29 (d)'de bir yörüngenin 4. çerçevedeki örtünme dolayısıyla devamlılığının doğru şekilde sağlanamadığı gösterilmektedir.



Şekil 2.29. Tespit ve hata atamaları (Leal-Taixé ve diğerleri, 2015)

3. MATERYAL VE YÖNTEM

Bu bölümde, evrişimli sinir ağlarının katmanlarından gelen derin özelliklereve veri ilişkilendirmeye dayalı çoklu nesne takibi çalışması için kullanılan materyal ve yöntemler anlatılmaktadır. Nesne tespiti çoklu nesne takibi çalışmalarında büyük öneme sahiptir. Nesne tespitinin başarısız olması nesne takibinin de başarısız olmasına neden olmaktadır. Bu nedenle nesne tespiti için seçilecek yöntem oldukça önemlidir ve nesne takibi başarısını doğrudan etkilemektedir.

Mevcut çalışmada öncelikle farklı nesne takibi yöntemleri ile tekli nesne takibi gerçekleştirilmiştir. Bu yöntemlerin takip performansı ve çalışma zamanı karşılaştırılması yapılmıştır. Ayrıca video görüntülerindeki hareketli nesnelerin tespiti için evrişimli sinir ağı kullanılmış ve her katmandan elde edilen özelliklerden nesne takibi için gerekli görünüm modellerini oluşturmada faydalanılmıştır. Literatürde var olan yöntemlerle de başarı ve hız karşılaştırması yapılmıştır. Nesne takibi için farklı görünüm modelleri ile korelasyon filtresi oluşturulmuştur. Bu bağlamda en etkin görünüm modeli seçilmiş ve gerçek zamanlı çoklu yaya takibi sürecine uyarlanmıştır. Bunların yanı sıra nesnelerin kaybolması, üst üste binmesi veya kısmi örtünme durumlarında nesnelerin doğru kimlik bilgisi ile takip edilmesini sağlayan bir veri ilişkilendirme algoritması geliştirilmiştir. Sonuçlar literatürde kabul görmüş çoklu nesne takibi başarı kıyaslama yöntemleri ile hesaplanmış ve mevcut çoklu nesne takibi çalışmaları ile de karşılaştırma yapılmıştır.

3.1. Veri Seti

Literatürde tekli ve çoklu nesne takibi çalışmalarında kullanılan çeşitli veri setleri bulunmaktadır. Bu çalışmada çoklu nesne takibi için 2D MOT 15 (Leal-Taixé ve diğerleri, 2015) tekli nesne takibi için ise OTB-100 (Wu ve diğerleri, 2015) veri setleri kullanılmıştır.

2D MOT 15 veri seti 11 eğitim ve 11 test video görüntüsü içermektedir. Farklı ortam ve şartlarda elde edilen videoların sırasıyla saniyedeki çerçeve, çözünürlük, uzunluk, takip,

kutu, yoğunluk, 3 boyut, kamera hareket bilgisi, kamera bakış açısı ve gölgeler bilgisi Çizelge 3.1'de gösterilmektedir.

Çizelge 3.1. 2D MOT 15 eğitim ve test veri seti (Leal-Taixé ve diğerleri, 2015)

Training sequences										
Name	FPS	Resolution	Length	Tracks	Boxes	Density	3D	Camera	Viewpoint	Shadows
TUD-Stadtmitte	25	640x480	179 (00:07)	10	1156	6.5	yes	static	medium	cloudy
TUD-Campus	25	640x480	71 (00:03)	8	359	5.1	no	static	medium	cloudy
PETS09-S2L1	7	768x576	795 (01:54)	19	4476	5.6	yes	static	high	cloudy
ETH-Bahnhof	14	640x480	1000 (01:11)	171	5415	5.4	yes	moving	low	cloudy
ETH-Sunnyday	14	640x480	354 (00:25)	30	1858	5.2	yes	moving	low	sunny
ETH-Pedcross2	14	640x480	840 (01:00)	133	6263	7.5	no	moving	low	sunny
ADL-Rundle-6	30	1920x1080	525 (00:18)	24	5009	9.5	no	static	low	cloudy
ADL-Rundle-8	30	1920x1080	654 (00:22)	28	6783	10.4	no	moving	medium	night
KITTI-13	10	1242x375	340 (00:34)	42	762	2.2	no	moving	medium	sunny
KITTI-17	10	1242x370	145 (00:15)	9	683	4.7	no	static	medium	sunny
Venice-2	30	1920x1080	600 (00:20)	26	7141	11.9	no	static	medium	sunny
Total training			**5503 (06:29)**	**500**	**39905**	**7.3**				

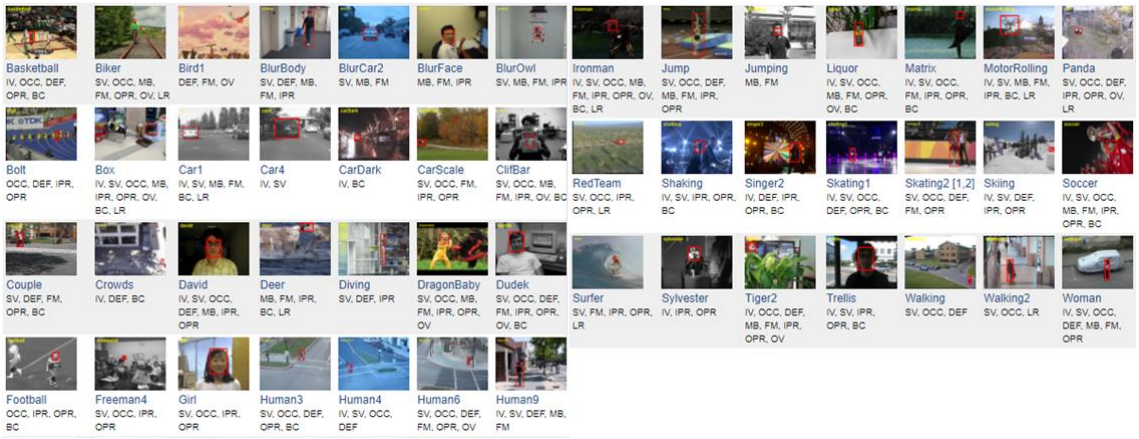
Testing sequences										
Name	FPS	Resolution	Length	Tracks	Boxes	Density	3D	Camera	Viewpoint	Weather
TUD-Crossing	25	640x480	201 (00:08)	13	1102	5.5	no	static	medium	cloudy
PETS09-S2L2	7	768x576	436 (01:02)	42	9641	22.1	yes	static	high	cloudy
ETH-Jelmoli	14	640x480	440 (00:31)	45	2537	5.8	yes	moving	low	sunny
ETH-Linthescher	14	640x480	1194 (01:25)	197	8930	7.5	yes	moving	low	sunny
ETH-Crossing	14	640x480	219 (00:16)	26	1003	4.6	no	moving	low	cloudy
AVG-TownCentre	2.5	1920x1080	450 (03:45)	226	7148	15.9	yes	static	high	cloudy
ADL-Rundle-1	30	1920x1080	500 (00:17)	32	9306	18.6	no	moving	medium	sunny
ADL-Rundle-3	30	1920x1080	625 (00:21)	44	10166	16.3	no	static	medium	sunny
KITTI-16	10	1242x370	209 (00:21)	17	1701	8.1	no	static	medium	sunny
KITTI-19	10	1242x374	1059 (01:46)	62	5343	5.0	no	moving	medium	sunny
Venice-1	30	1920x1080	450 (00:15)	17	4563	10.1	no	static	medium	sunny
Total testing			**5783 (10:07)**	**721**	**61440**	**10.6**				

2D MOT 15 veri seti için iki boyutlu tespit dosyasının örnek verisi Çizelge 3.2'de gösterilmektedir. İlk sütun yayanın hangi çerçevede görüldüğünü, ikinci sütun yayanın sahip olduğu eşsiz tanımlayıcı numarayı, üçüncü sütun yayanın çevreleyici kutusunun sol üst yataydaki piksel konumunu, dördüncü sütun yayanın çevreleyici kutusunun sol üst düşeydeki piksel konumunu, beşinci sütun yayanın çevreleyici kutusunun genişliğini, altıncı sütun yayanın çevreleyici kutusunun yüksekliğini ifade etmektedir. Daha sonraki yedinci sütun yaya tespitinin güven değerini ifade ederken son üç sütun yayanın gerçek dünya koordinatlarındaki 3 boyutlu konumunu ifade eder. Nesnenin 2 boyutlu olduğu durumlarda bu 3 sütunun değeri kullanılmaz ve değeri -1 olarak verilir. Yaya tespiti güven değerinin 0 olduğu durumlar ise hesaplamaya dâhil edilmemektedir. Çizelge 3.2'de gösterilen örnek veride 1 numaralı çerçevede 1, 2 ve 3 numaralı yayalar varken 2 numaralı çerçevede 1 numaralı yayanın hala göründüğü ifade edilir.

Çizelge 3.2. İki boyutlu tespit dosyası örnek verisi

1	2	3	4	5	6	7	8	9	10
1	1	794,2	47,5	71,2	174,8	1	-1	-1	-1
1	2	164,1	19,6	66,5	163,2	1	-1	-1	-1
1	3	875,4	39,9	25,3	145,0	0	-1	-1	-1
2	1	781,7	25,1	69,2	170,2	1	-1	-1	-1

OTB-100 veri seti farklı ortamlardan alınan video görüntüleri üzerinde farklı nesnelerin takip bilgilerini içermektedir. Bu çalışmada gerçekleştirilen tekli nesne takibi işlemlerinde Şekil 3.1'de gösterilen OTB-100 videoları kullanılmıştır. Her video için 'groundtruth_rect.txt' dosyası içerisinde takip edilen nesnenin her çerçevedeki konum bilgisi sırasıyla yer almaktadır. Her satır nesneyi çevreleyen kutunun sol üst köşe yatay konumu, sol üst köşe dikey konumu, genişliği ve yüksekliği olmak üzere 4 bilgiyi içermektedir.



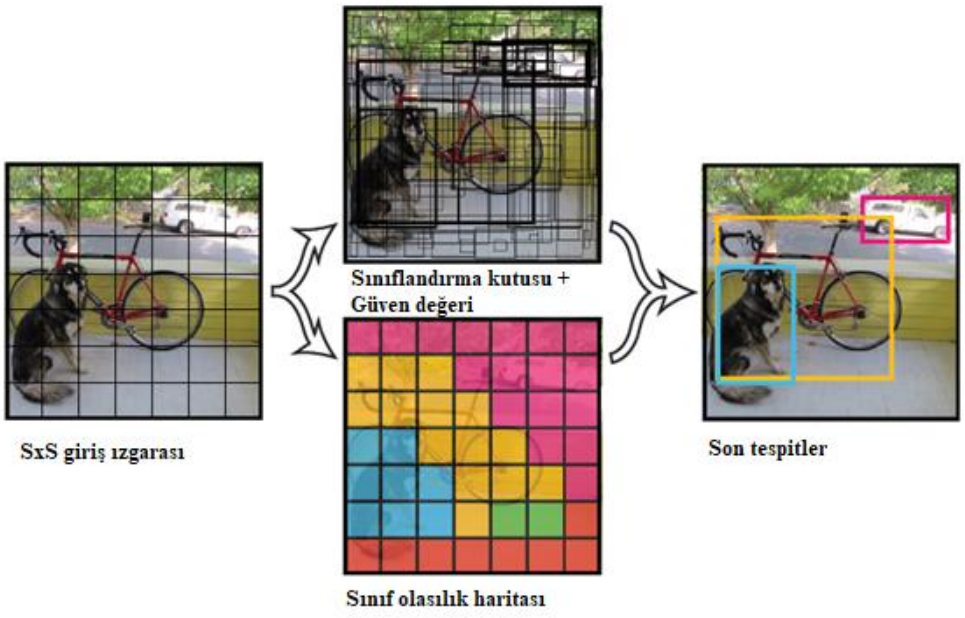
Şekil 3.1. OTB-100 veri seti ("Visual Tracker Benchmark", 2021)

3.2. Yazılım ve Donanım

Gerçekleştirilen tekli ve çoklu nesne takip sistemleri standart kapasitede bir dizüstü bilgisayarda çalıştırılmıştır. Donanım olarak bilgisayar 32 GB RAM, Intel i7-9700K merkezi işlem birimi ve GeForce RTX 2070 grafik işlemci birimi barındırmaktadır. Python programlama dili ile geliştirilen bu çalışmada kullanılan editör PyCharm editörü, kullanılan kütüphaneler ise Keras, Tensorflow, OpenCV, Numpy, Matplotlib kütüphaneleridir.

3.3. Yolo

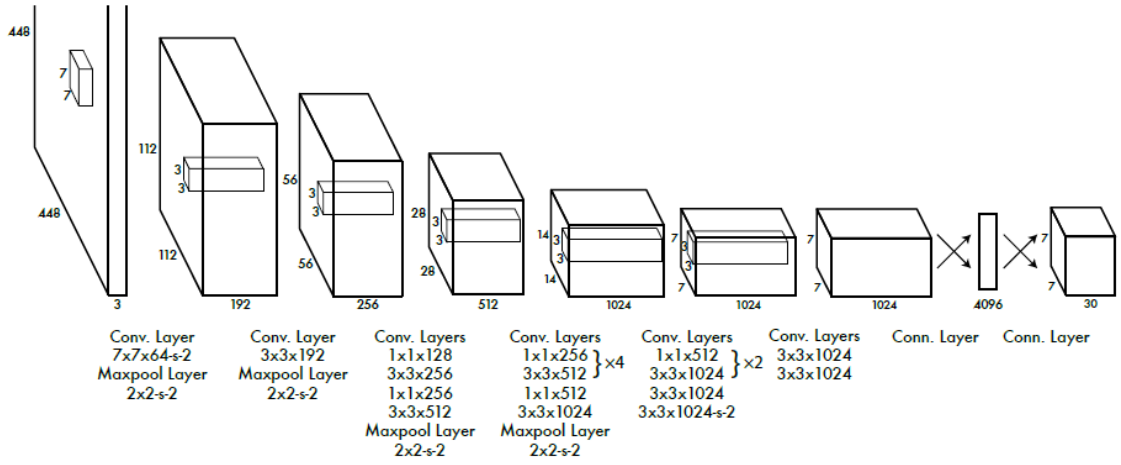
Literatürde çeşitli çalışmalar yüksek başarı oranları ile nesne tespitini gerçekleştirmektedirler. Donanımdaki gelişmeler sonrası derin öğrenme çalışmalarına yönelim artmış, bu artış da nesne tespiti başarı oranlarını arttırmıştır. R-CNN, Faster R-CNN, Single Shot Detector (SSD), You Only Look Once (YOLO) derin öğrenme tabanlı nesne tespiti yöntemlerinden bazılarıdır. Yolo hem performans hem de başarı oranının yüksek olması nedeniyle gerçek zamanlı nesne tespiti çalışmalarında tercih edilmektedir. İlk versiyonu Redmon ve diğerleri (2016) tarafından geliştirilen gerçek zamanlı nesne tespiti ağı Yolov1 daha sonraki yıllarda farklı versiyonlar ile güncellenmiştir. Redmon ve diğerleri (2016) Yolov1 ile nesne tespiti için yeni bir yaklaşım sunmuşlardır. Nesne algılamayı uzamsal olarak ayrılmış çevreleyici kutu ve ilişkili sınıf olasılıkları ile regresyon problemi olarak ele almışlardır. Tahmin işlemi için görüntü tek bir yapay sinir ağından geçirilir ve saniyede 45 kare ile gerçek zamanlı işlem sağlanır. Girdi görüntüsü sistem tarafından 448x448 olarak yeniden boyutlandırılır. Şekil 3.2'de gösterildiği gibi yeniden boyutlandırılmış görüntü SxS boyutlu ızgaralara ayrılmaktadır. Eğer nesnenin merkez noktası bir ızgaranın içinde yer alıyorsa o ızgara nesnenin algılanmasında belirleyici bir rol oynar.



Şekil 3.2. Yolo modeli (Redmon ve diğerleri, 2016)

Izgaraların her biri B kadar çevreleyici kutu tanımlanır ve o kutuların güven değerleri tahmin edilir. Bu güven değeri ağın o ızgara içerisinde nesne olduğundan ne kadar emin olduğu ve kutunun tahmininin ne kadar doğru olduğu bilgisini tutar. Güven değeri $Pr(Object)*IOU^{truth\ pred}$ olarak ifade edilir. Her çevreleyici kutu için x, y, w, h, s olmak üzere 5 tahmin yapılır. Ayrıca her ızgara için $Pr(Class_i|Object)$ şeklinde ifade edilen C koşullu sınıf olasılığı tahmin edilir. Her ızgara için kutuların sayısından bağımsız olarak sadece bir sınıf olasılık dağılımı bilgisi tutulur.

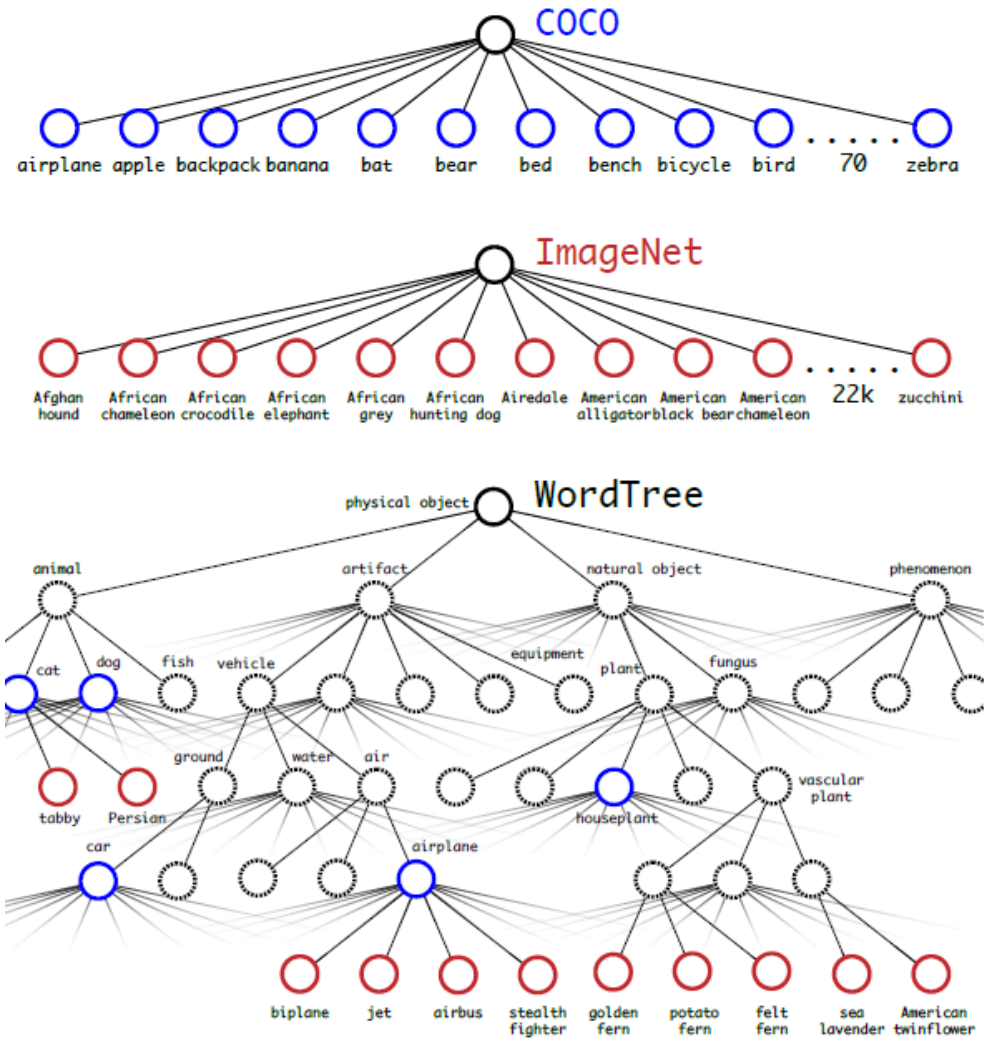
Yolov1 evrişimli sinir ağı PASCAL VOC nesne tanıma veri seti üzerinde uygulanmıştır. İlk katman nesne özelliklerini çıkartırken FC katmanı ızgara olasılıkları ve kutu konumlarını tahmin eder. Yolov1 ağ mimarisi GoogleNet ağ modeline dayanmaktadır. Şekil 3.3'te Yolov1 ağ mimarisi gösterilmektedir. Ağda 24 evrişim katmanı ve 2 FC katmanı bulunmaktadır.



Şekil 3.3. Yolov1 ağ mimarisi (Redmon ve diğerleri, 2016)

Yolov1 her ızgara hücresi yalnız 2 kutuyu tahmin edip 1 sınıfa sahip olabildiği için çevreleyici kutu tahminlerine güçlü uzamsal kısıtlar getirir ve modelin tahmin edeceği yakın nesnelerin sayısı sınırlanır. Özellikle grup halinde olan küçük nesnelerin algılanmasında başarı oranı düşmektedir. Çalışmadaki ana hata kaynağı nesnelerin yanlış konumlandırılmasıdır.

Redmon ve Farhadi (2016) 9000'den fazla nesne kategorisini gerçek zamanlı olarak algılayabilen Yolo9000 isimli bir ağ geliştirmişlerdir. Yolov1 çalışmasının devamı olan bu çalışmada sınıflandırma başarısı sürdürülürken Yolov1'deki konumlandırma eksiklikleri üzerine odaklanılmıştır. Özetle Yolo9000 gerçek zamanlı bir konvolüsyonel sinir ağı olarak algılama ve sınıflandırma problemlerini birlikte optimize etmiş ve 9000'den fazla nesne kategorisini algılamayı başarmıştır. ImageNet ve Coco veri setlerini birlikte kullanarak ağı eğitmek için Şekil 3.4'te gösterilen WordTree isimli bir hiyerarşik yapı kullanılmıştır.



Şekil 3.4. Yolo9000 eğitiminde WordTree hiyerarşik yapısı ile veri seti birleştirme (Redmon ve Farhadi, 2016)

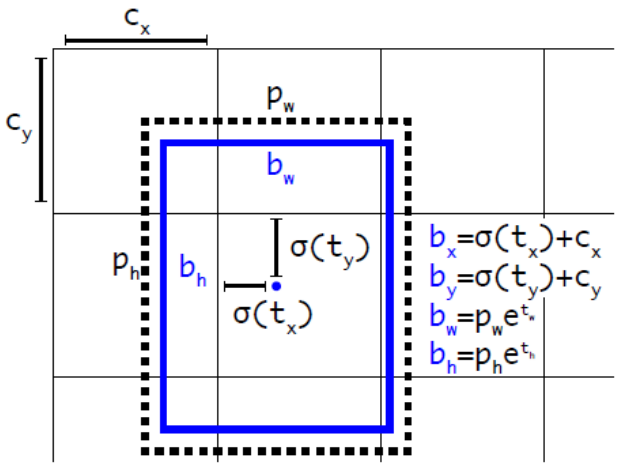
Yolov2 ile eklenen geliştirmeler ve başarı oranları adım adım Şekil 3.5'te gösterilmektedir. FC katmanları kaldırılmış yerine kutuları tahmin etmek için destek (anchor) kutuları kullanılmıştır. Ayrıca 448 olan giriş görüntüsü boyutu 416 olarak güncellenmiş ve evrişim katmanının çıktısının daha yüksek çözünürlüklü hale gelmesi için bir havuzlama katmanı kaldırılmıştır. Mevcut ağda görüntü başına 1000'den fazla kutu kullanımı öngörülmektedir.

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Şekil 3.5. Yolov1'den Yolov2'ye yol haritası (Redmon ve Farhadi, 2016)

Destek kutuları kullanılmadan önce Yolov1 her bir ızgarada sadece bir nesne tahmin edebiliyorken bu kutuların kullanımı ile artık her bir ızgarada daha fazla nesne tahmin edilmesi mümkün hale gelmiştir. Destek kutuları ağın eğitim işlemi öncesinde belirlenmiş belirli yükseklik ve genişliğe sahip kutulardır. Bir takım parametreler ile oluşturulan bu kutular çok küçük veya büyük nesnelerin tahmin edilebilmesini etkilemektedir.

Yolov3 Redmon ve Farhadi tarafından 2018 yılında geliştirilmiştir. Yolov3 çevreleyici kutu tahminlerini Yolo9000'deki gibi boyut kümelerini kullanarak gerçekleştirir. Her çevreleyici kutu için ağ tarafından t_x, t_y, t_w, t_h şeklinde bir koordinat ve boyut bilgisi tahmin edilir. Nesnelerin çevreleyici kutuları Şekil 3.6'da gösterildiği şekilde hesaplanmaktadır.



Şekil 3.6. Çevreleyici kutunun boyut öncelikleri ve konum tahmini ile bulunması (Redmon ve Farhadi, 2018)

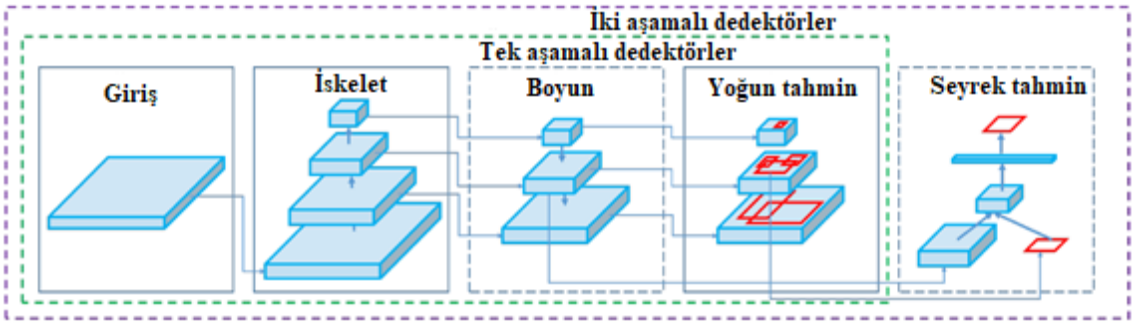
Yolov3'te entropi (softmax) yerine lojistik regresyon kullanılmaktadır. Bir görüntüde tek bir nesne yerine birden fazla nesne tespit edilebilmektedir. Kutuları 3 farklı ölçekte tahmin etmektedir. Çevreleyici kutu önceliklerini belirlemek için k-ortalama yöntemi kullanılmaktadır. Özellik çıkarmak için Yolov2 ağ yapısı ve Darknet-19 ağ yapısı birleştirilip hibrit bir mimari kullanılmıştır. Darknet-53 olarak isimlendirilen ve Şekil 3.7'de gösterilen bu mimari toplamda 53 evrişim katmanına sahiptir. Darknet-53 Darknet-19'dan çok daha güçlü ve ResNet-101 veya ResNet-152'den daha verimlidir (Redmon ve Farhadi, 2018).

Nesne tespiti algoritmaları tek aşamalı ve iki aşamalı olarak gruplandırılabilir. R-CNN ailesinin dahil olduğu iki aşamalı nesne tespiti yöntemlerinde ilk olarak arama bölgeleri belirlenirken ikinci aşamada sınıflandırma yaklaşımıyla bölge adayları belirlenir. Yolo mimarilerinin dahil olduğu tek aşamalılarda ise tespit olası konumların örneklemeleri üzerinden çalıştırılır. Tek ve çift aşamalı nesne tespit ağlarının genel yapısı Şekil 3.8'de gösterilmiştir. İlk katman görüntünün ağa verildiği girdi (input) katmanıdır. Daha sonra özellik çıkarımında kullanılan iskelet (backbone) katmanı bulunmaktadır. VGG16, ResNet-50, CSPDarknet53 gibi ağ mimarileri bu katmanda seçilir. Boyun (neck) ara katmanı daha fazla bilgi elde etmek amacıyla kullanılmaktadır. Yolov3'te boyun ara katmanında Özellik Piramit Ağı (Feature Pyramid Network) seçilmiştir. Baş (head)

katmanında tek aşamalı tespit ağları için yoğun tahmin (dense prediction), çift aşamalı tespit ağları için ise seyrek tahmin (sparse prediction) kullanılır.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Şekil 3.7. Darknet-53 (Redmon ve Farhadi, 2018)



Şekil 3.8. Tek ve çift aşamalı nesne tespit ağları (Bochkovskiy ve diğerleri, 2020)

Yolov4 ile Yolov3'te sağlanan ortalama hassasiyet %10, fps ise %12 arttırılmıştır (Bochkovskiy ve diğerleri, 2020). Yolov4 ile gelen yenilikler nesne tespiti eğitimi için

ekler çantası (bag of freebies - BoF) ve ayrıcalıklar çantası (bag of specials - BoS) yöntemlerinin kullanılması ve tek bir GPU kullanımı ile eğitim işleminin etkin hale getirilmiş olmasıdır. Bunlara ek olarak tek GPU üzerinde eğitim için mozaik ve öz muhalif eğitim (self-adversarial training - SAT) yeni veri arttırma metotları geliştirilmiş; genetik algoritmaları uygularken optimum hiperparametreler seçilmiş; SAM (Woo ve diğerleri, 2018), PAN ve çapraz mini yığın normalleştirmesi (cross mini-batch normalization - CmBN) üzerinde değişiklikler gibi yenilikler eklenmiştir.

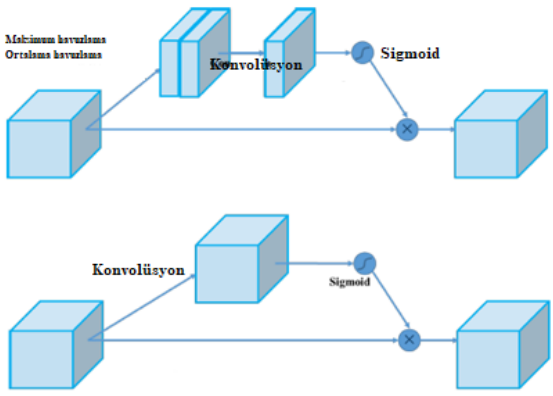
Nesne tespit yöntemleri tarafından benimsenen ve BoF olarak bilinen teknik genellikle veri arttırmadır. Veri arttırımının amacı girdi görüntüsünü farklı fotometrik ve geometrik bozulmalara uğratarak nesne tespit modelinin farklı ortamlardan elde edilen görüntülere karşı daha sağlam çalışmasını sağlamak ve böylece başarı oranını yükseltmektir. Yolov4 fotometrik bozulma için girdi görüntüsünün parlaklığını, kontrastını, tonunu, doygunluğunu ve gürültüsünü ayarlamaktadır, geometrik bozulma için ise rastgele ölçekleme, kırpma, çevirme ve döndürme eklemektedir. Bunlara ek olarak stil aktarımı (Generative Adversarial Networks - GAN) da veri arttırma için kullanılmaktadır ve bu yöntemle evrişimli sinir ağı tarafından öğrenilen doku önyargısı da etkin bir şekilde azaltılabilmektedir.

BoS tespit maliyetini az miktarda arttırırken tespit başarısını önemli ölçüde arttırabilen eklenti yöntemlere verilen isimdir. Yolov4'te iki farklı gerçek zamanlı evrişimli sinir ağı sunulmaktadır. İlk ağda GPU için CSPResNeXt50 ya da CSPDarknet53 ile 1 ve 8 arası gruplanmış evrişim kullanımı, ikinci ağda görsel işlemci birimi (Visual processing unit - VPU) için gruplanmış evrişim kullanımı gerçekleştirilmiştir. Yolov4' te kullanılan yöntemler aşağıdaki gibi özetlenebilir.

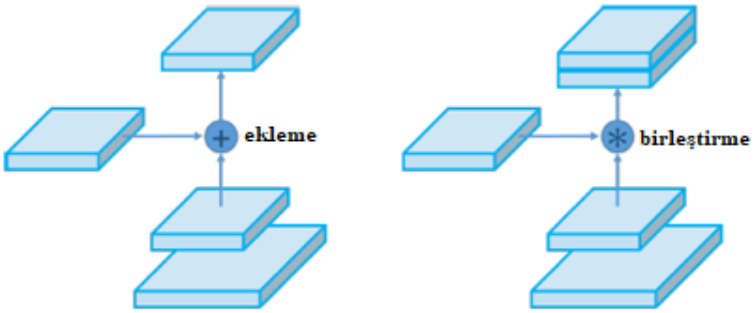
- İskelet: CSPDarknet53 (Wang ve diğerleri, 2020)
- Boyun: SPP, PAN (Liu ve diğerleri, 2018)
- Baş: Yolov3
- İskelet için BoF: CutMix ve mozaik veri artırımı, DropBlock düzenleme, sınıf etiketi yumuşatma

- İskelet için BoS: görev aktivasyonu (Mish activation), Çapraz evre kısmi bağlantılar (Cross-stage partial connections - CSP), çok girdili ağırlıklandırılmış artık bağlantılar (Multi-input weighted residual connections - MiWRC)
- Nesne tespiti için BoF: CIoU-loss, CmBN, DropBlock düzenlileştirme, mozaik veri artırımı, SAT, ızgara hassasiyetini giderme, tek bir doğru değer için çoklu destek (anchor) kullanımı, kosinüs tavlama zamanlayıcı, optimal hiperparametreler, rastgele eğitim şekilleri
- Nesne tespiti için BoS: görev aktivasyonu, SPP-bloğu, SAM-bloğu, PAN yol birleştirme bloğu, DIoU-NMS

Mozaik dört farklı eğitim görüntüsünü birleştirerek yeni bir veri arttırma yaklaşımı getirir. Böylelikle nesnelerin varsayılan bağlamları dışında da tespitine imkan sağlar. Yüksek mini yığın boyutu ihtiyacı önemli oranda azaltılır. SAT ileri ve geri olarak adlandırılan iki aşamalı yeni bir veri arttırma tekniğidir. İlk aşamada yapay sinir ağı ağın ağırlıkları yerine orijinal görüntüyü değiştirir. Böylelikle yapay sinir ağında görüntüde istenen bir nesnenin olmadığı yanılgısını oluşturmak için ağa karşı bir çeşit saldırı gerçekleştirilir. Sonraki aşamada yapay sinir ağı bu değiştirilmiş görüntüde nesne tespit etmek için eğitilir. CmBN çağraz yığın normalleştirmesinin değiştirilmiş bir sürümüdür. SAM uzamsal dikkatten noktasal dikkat için değiştirilmiştir. PAN'in kısa yol bağlantısı toplama yerine birleştirme ile değiştirilmiştir. Şekil 3.9'da üst kısımda SAM alt kısımda ise Yolov4 ile değiştirilmiş SAM bulunmaktadır. Şekil 3.10'da ise sol kısımda PAN sağ kısımda Yolov4 ile değiştirilmiş PAN bulunmaktadır.

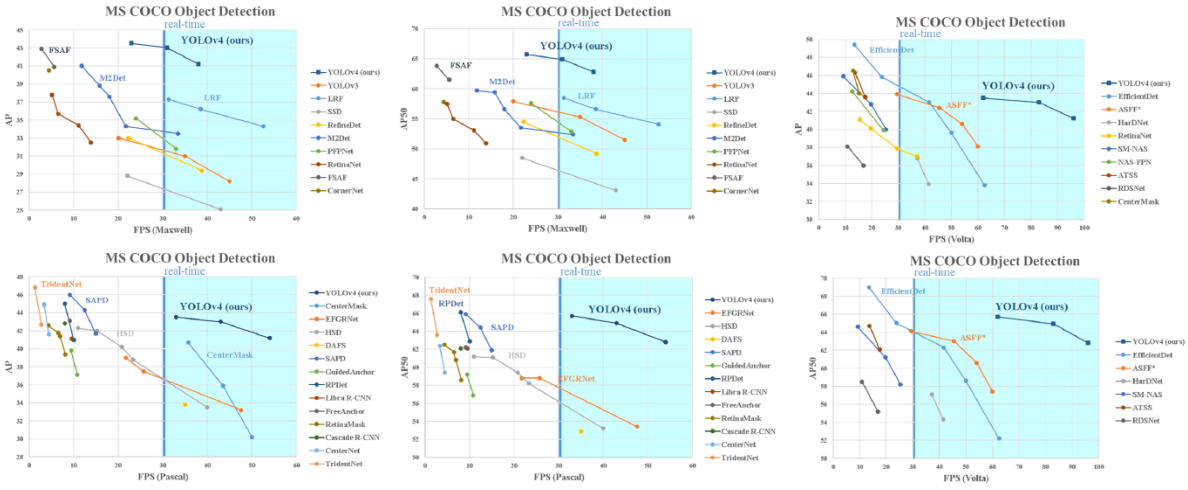


Şekil 3.9. SAM (Bochkovskiy ve diğerleri, 2020)



Şekil 3.10. PAN (Bochkovskiy ve diğerleri, 2020)

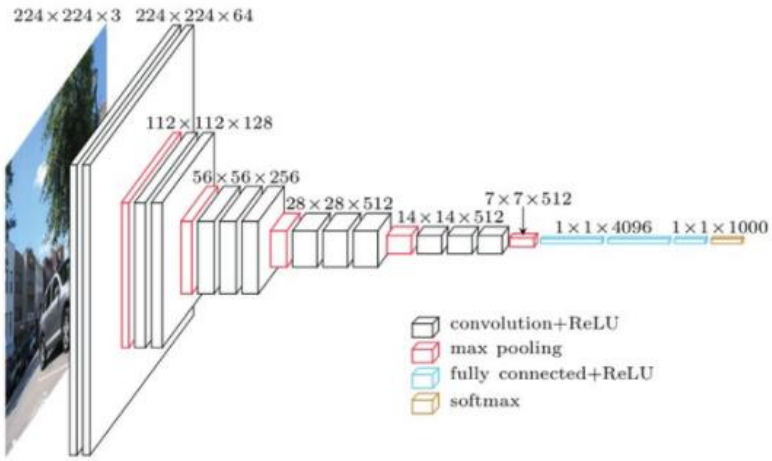
Değiştirilmiş PAN ile amaçlanan bilgileri birleştirerek doğruluk oranını arttırmaktır. Yolov4 fps ve başarı karşılaştırmaları Şekil 3.11'de gösterilmektedir. Bazı çalışmaların hızı Maxwell, Pascal, Volta gibi GPU modellerinin yalnız biri için verilmiştir. Farklı nesne tespiti algoritmalarının hız ve doğruluk karşılaştırmalarına bakıldığında Yolov4 hız ve doğruluk için oldukça başarılı sonuçlar göstermiştir. Dolayısıyla bu çalışmada gerçek zamanlı çoklu nesne takibi için yayaları tespit etmede ve bunların derin özelliklerini belirlemede Yolov4 tercih edilmiştir.



Şekil 3.11. Farklı nesne tespiti yöntemlerinin hız ve doğruluğunun karşılaştırması (Bochkovskiy ve diğerleri, 2020)

3.4. Visual Geometry Group-19

Visual Geometry Group-19 (VGG-19) Simonyan ve Zisserman tarafından geliştirilen 16 evrişim, 3 tam bağlı, 5 maksimum havuzlama ve 1 entropi (softmax) olmak üzere toplam 19 katman içeren bir derin öğrenme ağıdır (Simonyan ve Zisserman, 2014). Şekil 3.12'de VGG-19 ağ yapısı gösterilmektedir. Şekil 3.13'te VGG-19 ağ yapısında kullanılan evrişim, maksimum havuzlama, tam bağlı ve entropi katmanları kırmızı kutularla çerçevelenmiş girdilerde ifade edilmektedir.



Şekil 3.12. VGG-19 derin öğrenme ağı (Simonyan ve Zisserman, 2014)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Şekil 3.13. VGG-19 katmanları (Simonyan ve Zisserman, 2014)

Bu çalışmanın tekli nesne takibi kısmında VGG-19 derin öğrenme ağı kullanılarak farklı evrişim katmanlarından elde edilen özellikler ile nesne takibi deneyleri yapılmıştır. Ağın 5 farklı bloğunun son evrişim katmanlarından elde edilen özelliklerin nesne takip başarısına etkisi gözlemlenmiştir. Farklı katmanların birlikte ve yalnız kullanımlarının takip başarısına etkisi analiz edilmiştir.

3.5. Çekirdek Korelasyon Filtresi

Korelasyon filtresi uzamsal alanda doğrusal ayırıcı olarak görülebilir. Korelasyon filtresine dayalı nesne takip yöntemlerinde hedef nesne modellenerek görüntü arka planından ayırt edilmeye çalışılır. Bunun için eğitim görüntüleri kullanılır ve hedef görünüm modeli oluşturulur ya da hedef görünüm belirlendikten sonra hedef nesnenin dairesel kaymaları kullanılarak hedef görünüm modeli oluşturulur. Model eğitme işlemi ve model güncelleme işlemi eş zamanlı olarak gerçekleştirilir. Nesneyi takip etmek için hedef görünüm modeli kullanılarak oluşturulan korelasyon filtresi sonraki çerçevelerde bir arama penceresi ile ilişkilendirilir.

Korelasyon işleminin çıktısındaki değerler arasından en yüksek değere sahip olan arama penceresi hedef nesnenin takip edilmiş konum bilgisini verir. Model güncelleme işlemi bu yeni konum bilgisi kullanılarak çevrim içi gerçekleşir. Korelasyon işlemi Fourier boyutunda gerçekleştirildiğinde hızlı çalışmakta, bu nedenle gerçek zamanlı nesne takibi uygulamalarında tercih edilmektedir.

Korelasyon işlemi girdi görüntü matrisi ve çekirdek olarak da adlandırılan filtre matrisini alarak filtreyi görüntü matrisi üzerinde gezdirir. Bu gezdirme sürecinde filtre değerleri ile görüntüde karşılık gelen piksel komşuluk değerlerinin nokta çarpımını hesaplar ve çıktı matrisini üretir. Evrişim (convolution) işlemi ile benzer tarafları bulunmaktadır. İki işlem arasındaki temel fark evrişimde nokta çarpımını hesaplamadan önce filtrenin yatay ve dikey eksenlerde çevrilmesidir. Korelasyon ve evrişim işlemlerinin farkları Çizelge 3.3'te gösterilmektedir.

Çizelge 3.3. Korelasyon ve evrişim farkları

	Korelasyon	Evrişim
Çekirdek Döndürme	Çekirdek döndürülmez.	Çekirdek döndürülür.
Formül	$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$	$F * I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x-i, y-j)$

Çekirdek korelasyon filtresi (KCF) takip problemini tepe regresyon problemi olarak ele alır. Tepe regresyon basit bir kapalı form çözümü kabul eder ve daha karmaşık yöntemlere yakın bir performans elde edebilir. Eğitimde amaçlanan (3.1)'de gösterildiği gibi x_i örnekleri ve bunların regresyon hedefleri y_i üzerindeki kare hatayı en aza indiren bir $f(z) = w^T z$ fonksiyonu bulmaktır. Düzenlileştirme parametresi olan λ parametresi aşırı öğrenmeyi kontrol etmek için kullanılır.

$$\min_{w} \sum_{j=1}^{n} (f(x_i) - y_i))^2 + \lambda ||w||^2 \tag{3.1}$$

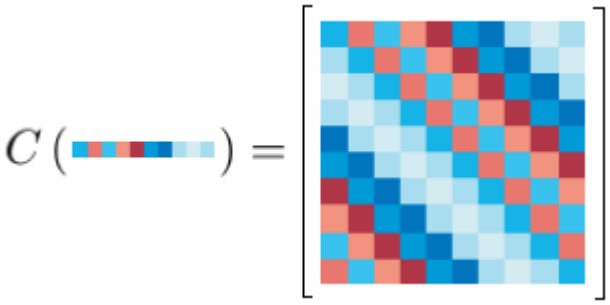
Hatayı en aza indiren çözüm (3.2)'deki gibi kapalı bir formda bulunabilir (Rifkin ve diğerleri, 2003). Bu kapalı formülde X, her satırı için bir x_i örneğini barındıran bir veri matrisini ifade ederken, I birim matrisini ve y ise regresyon hedefini temsil etmektedir. Aynı formülün Fourier uzayındaki karmaşık versiyonu ise (3.3)'te verilmektedir.

$$w = (X^T X + \lambda I)^{-1} X^T y \tag{3.2}$$

$$w = (X^H X + \lambda I)^{-1} X^H y \tag{3.3}$$

Döngüsel kayma hedef görüntü verisinin yatay ve dikeydeki olası tüm kaymalarını ifade etmektedir. Örneğin n×1 boyutlu bir x hedef verisi ele alındığında amaç hedef veriyi pozitif, olası tüm kaymalarla elde edilen diğer verileri ise negatif örnekler olarak kullanarak bir sınıflandırıcı eğitmektir. Örnek bir kayma görüntüsü Şekil 3.14'te gösterilmektedir. En üst satır n×1 boyutlu x hedef verisini gösterirken, ikinci satır hedef

verinin 1 kaymasını, üçüncü satır hedef verinin 2 kaymasını, son satır ise hedef verinin n-1 kaymasını göstermektedir.



Şekil 3.14. Kayma matrisi gösterimi (Henriques ve diğerleri, 2014)

Kayma matrisi üzerinden tanımlanan X veri matrisi (3.4)'teki denklemle ifade edilir. Burada F ayrık Fourier sabiti olarak bilinmektedir. (3.2) ile (3.4) beraber ele alındığında oluşan yeni formül (3.5)'te gösterilmektedir.

$$X = F\ diag(x)\mathrm{F}^H \tag{3.4}$$

$$X^H X = F\ diag(x^*)\mathrm{F}^H F diag(x) F^H \tag{3.5}$$

Korelasyon filtresi w'nun belirlenebilmesi için yapılan hesaplamalar sırasıyla (3.6)'dan (3.9)'a kadar olan formüllerde ifade edilmiştir. Korelasyon filtresi için kullanılan Gauss çekirdek denklemi ise (3.10)'da verilmektedir. $\mathrm{F}^H F = I$ faktörü kaldırıldığında (3.5) denklemi (3.6)'ya dönüşür. Köşegen matrislerdeki işlemler eleman bazındai olduğu için denklem (3.6) eleman bazında çarpım ile (3.7)'deki gibi ifade edilebilir. Bu denklemler doğrusal regresyon için denklem (3.3)'e yinelemeli olarak uygulandığında denklem (3.8)'deki gibi nicelikler köşegenin (diagonal) içine koyulabilir. Denklem (3.9) ile hedef nesnenin tespit edilmesi için oluşturulan filtre (3.8)'in iyileştirilmiş şekliyle ifade edilir.

$$X^H X = F\ diag(x^*) diag(x) F^H \tag{3.6}$$

$$X^H X = F\ diag(x^* \odot x) F^H \tag{3.7}$$

$$w = diag\left(\frac{x^*}{x^* \odot x + \lambda}\right) y \tag{3.8}$$

$$w = \frac{x^* \odot y}{x^* \odot x + \lambda} \tag{3.9}$$

$$k^{x\,x\prime} = exp\left(-\frac{1}{\sigma^2}\left(\left|\left|x\right|\right|^2 + \left|\left|x'\right|\right|^2 - 2F^{-1}\left(x^* \odot x'\right)\right)\right) \tag{3.10}$$

3.6. Veri İlişkilendirme

Çoklu nesne takibi çalışmalarında kullanılan veri ilişkilendirme n. çerçevede kimliklendirilmiş ve (n+1). çerçevede takip edilen nesnelerin yine (n+1). çerçevede tespit edilen nesneler ile ilişkilendirilmesidir. Veri ilişkilendirme çoklu nesne takibi başarısı için büyük bir öneme sahiptir ve başarısı nesnelerin birbirinden ne kadar fazla oranda ayırt edildiği ile doğru orantılıdır. Eğer nesneler birbirinden oldukça farklı ise ilişkilendirme başarılı iken nesneler birbirine çok benzer ise bu başarı düşebilmektedir. Bu durumda ilişkilendirme için kullanılan özellik seçimi başarı oranını doğrudan etkilemektedir. Bu çalışmada özellik seçiminin veri ilişkilendirme başarısına etkisini ölçmek amacıyla farklı özellikler kullanılmıştır.

İlişkilendirme için kullanılan özelliklerden birincisi çevreleyici kutuların bilgisi, ikincisi renk histogramı bilgisi, diğeri ise Yolov4 ağından elde edilen 128x1 boyutlu derin özelliklerdir. Ayrıca çalışmada kullanılan mesafe ölçütleri kosinüs mesafesi ve kesişimin birleşime oranı ölçütleridir. Çevreleyici kutular üzerinden hesaplanan kesişimin birleşime oranı ve renk histogramı ile derin özellikler üzerinden hesaplanan kosinüs mesafesi ölçümleriyle Macar algoritması çalıştırılmış ve buna göre sonuç çoklu nesne takip başarıları değerlendirilmiştir. Takip edilen bir yayanın yeni tespit edilen yayalar ile arasındaki benzerlik bu ölçütlerle hesaplanır. Öklid mesafesi arttıkça iki yayanın birbirinden uzaklaştığı, azaldıkça birbirine yakınlaştığı yani eşleştiği değerlendirilir. 0 ile 1 aralığında olan kosinüs benzerliği için ise yayaların özellikleri birebir aynı olduğu takdirde en yüksek benzerlik değeri olan 1 elde edilir. Kosinüs benzerliği ölçütü (3.11)'de ifade edilmektedir. İlişkilendirme sürecinde eşleşme

maliyetini ifade etmesi için kosinüs benzerliğinin ve kesişimin birleşime oranı metriğinin kullanılmasıyla elde edilen ölçümler 1'den çıkartılarak kullanılır. Birden çıkartılmış kosinüs benzerliği değeri kosinüs mesafesi olarak adlandırılır.

$$\cos(\beta) = -\frac{v.w}{||v||||w||} \tag{3.11}$$

3.7. Çoklu Yaya Takibi Yöntemlerinin Birleştirilmesi

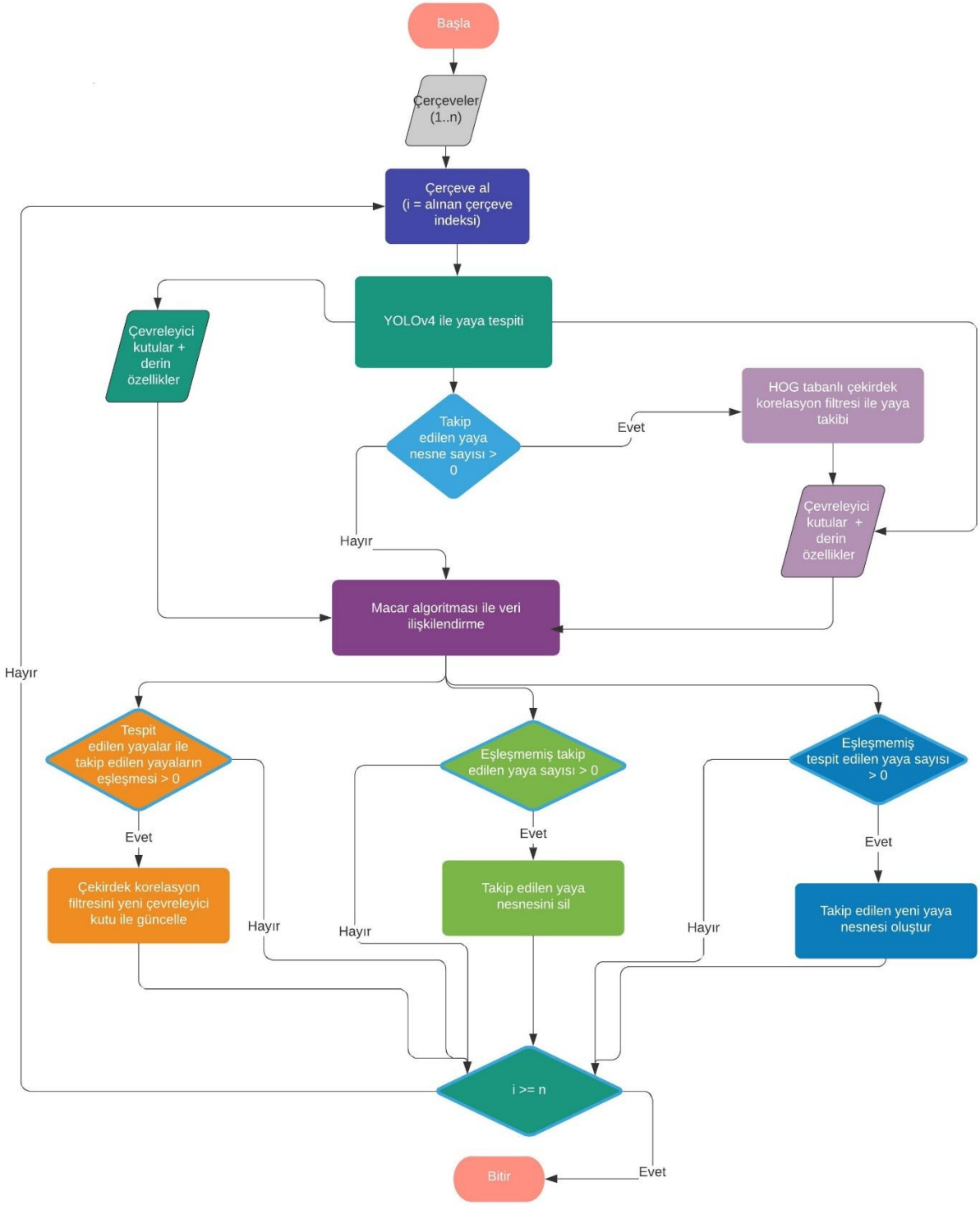
Bu çalışmanın son kısmında farklı ortam şartlarında çoklu yaya tespiti ve takibi gerçekleştirilmiştir. İlk çerçevede yayaların tespit edilmesi için Yolov4 nesne tespiti yöntemi kullanılmaktadır. Çerçevede tespit edilen yayaların sonraki çerçevedeki konumunun tahmini için gerçek zamanlı ve etkin çalışması dolayısıyla HOG tabanlı çekirdek korelasyon filtresi kullanılarak takip işlemi gerçekleştirilmiştir. Sonraki çerçeve için tekrar Yolov4 nesne tespiti yöntemi çalıştırılarak yeni yayalar tespit edilmiştir. Mevcut çerçevede takip edilen yayalar ile tespit edilen yayaların ilişkilendirilmesi için Macar algoritması kullanılmıştır. Çoklu nesne takibi süreci yinelemeli bir şekilde nesne tespiti, nesne takibi ve yayaların ilişkilendirilmesi suretiyle devam etmektedir.

Çalışmada çoklu yaya takibinde kullanılan veri ilişkilendirme için üç farklı yöntem denenmiştir. Veri ilişkilendirme için Bölüm 3.6'da bahsedilen konum tabanlı veya görünüm tabanlı özellikler kullanılır. İlk yöntem olan derin özellikler ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi sürecinin akış diyagramı Şekil 3.15'te gösterilmektedir. Algoritmanın akışı ilk çerçeve elde edildikten sonra Yolov4 ile çerçevedeki yayaların tespit edilmesi ile başlar. Daha sonra eğer takip edilen yaya nesne sayısı 0'dan büyük ise yayalar çekirdek korelasyon filtresi ile takip edilir. Daha sonra Yolov4 ile tespit edilen yayalar ile takip edilen yayalar Macar algoritması ile ilişkilendirilir. Bu ilişkilendirmede her yaya nesnesi için 128x1 boyutlu derin özellikler ve kosinüs metriği kullanılır. İlişkilendirme sırasında yayaların derin özellikleri için kosinüs mesafesi hesaplanır ve bir maliyet matrisi oluşturulur. Oluşturulan maliyet matrisi Macar algoritması ile optimize edilir ve en uygun atamalar gerçekleştirilir. Veri

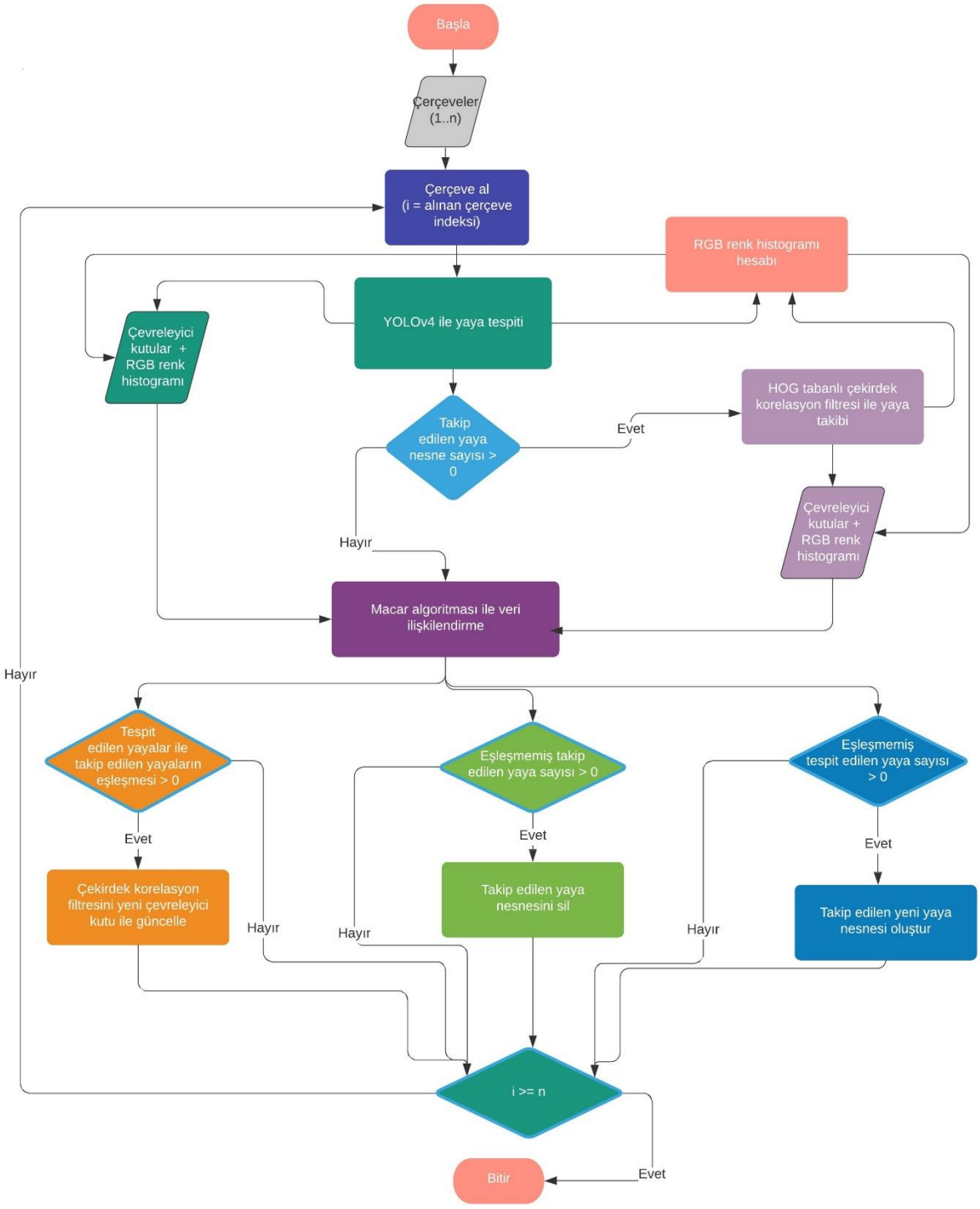
ilişkilendirme sonrası takip edilen yayalar ile tespit edilen yayalar eşleştiği durumda takip edilen yayalar için çekirdek korelasyon filtresinde yeni çevreleyici kutular ile güncelleme yapılır. Takip edilen yayaların derin özellikleri Yolov4 ile tespit edilen ilişkilendirilmiş yayaların derin özellikleri ile güncellenir. Takip edilen yayalar içerisinden tespit ediler yayalar ile eşleşmeyen yayaların olması durumunda yayalar için oluşturulmuş nesneler silinme durumuna geçer. Kısmi tıkanma, kısa süreli yayaların gözden kaybolması gibi durumlar göz önünde bulundurularak yayalar hemen silinmemektedir. Art arda 5 görüntüde ilgili yayanın tespit edilip ilişkilendirilmediği durumda yaya nesnesi silinerek sistemden tamamen kaldırılır. Tespit edilen yayalar içerisinden takip edilen yayalar ile eşleşmeyen yayaların olması durumunda eşleşmeyen tespit edilmiş yayalar için yeni takip yaya nesneleri oluşturulur. Daha sonra takip eden çerçeve alınarak aynı işlemler yinelemeli bir şekilde tekrar çalıştırılır. Son çerçeve için de tespit, takip ve ilişkilendirme adımları tamamlandıktan sonra çoklu nesne takip sürecinin çalışması sonlanır.

İkinci yöntem olan RGB renk histogramı ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi sürecinin akış diyagramı Şekil 3.16'da gösterilmektedir. Algoritmanın akışı ilk yönteme benzer şekilde işlemektedir. Ancak veri ilişkilendirme adımı için derin özellikler yerine RGB renk histogramı kullanılmaktadır. İlişkilendirme sırasında yayaların RGB renk histogramı özellikleri için kosinüs mesafesi hesaplanır ve bir maliyet matrisi oluşturulur. Oluşturulan maliyet matrisi Macar algoritması ile optimize edilir ve en uygun atamalar gerçekleştirilir.

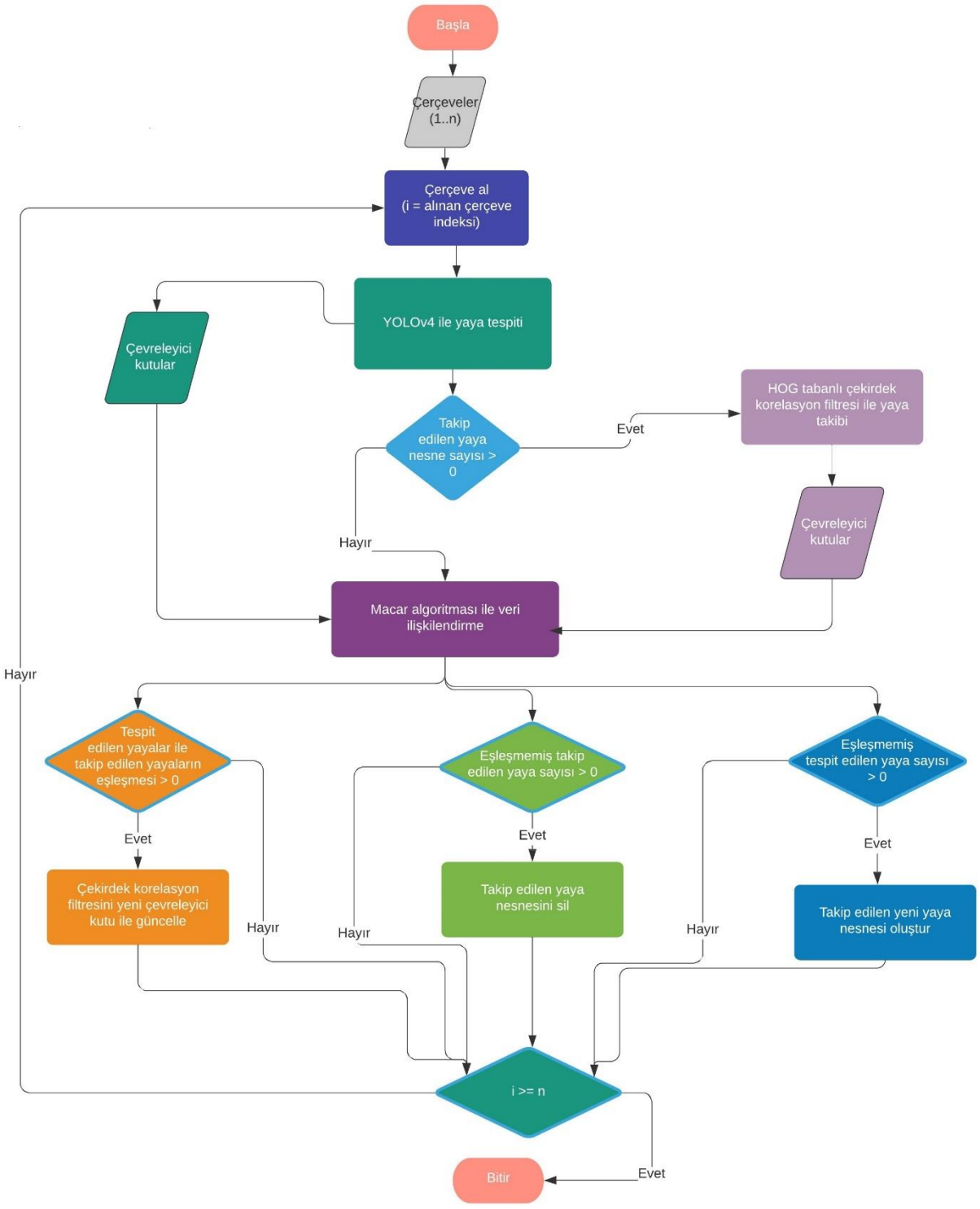
Üçüncü yöntem olan çevreleyici kutular ve IOU metriği ile yayaların ilişkilendirildiği çoklu yaya takibi sürecinin akış diyagramı Şekil 3.16'da gösterilmektedir. Algoritmanın akışı ilk iki yönteme benzer şekilde işlemektedir. Ancak veri ilişkilendirme adımı için derin özellikler ya da RGB renk histogramı yerine yalnızca çevreleyici kutular kullanılmaktadır. İlişkilendirme sırasında yayaların çevreleyici kutuları için IOU hesaplanır ve bir maliyet matrisi oluşturulur. Oluşturulan maliyet matrisi Macar algoritması ile optimize edilir ve en uygun atamalar gerçekleştirilir. Son çerçeve de tamamlanana kadar tespit, takip ve ilişkilendirme yinelemeli şekilde çalıştırılır ve son çerçeveden sonra sistem çalışması sona erer.



Şekil 3.15. Derin özellikler ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı



Şekil 3.16. RGB renk histogramı ve kosinüs metriği ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı



Şekil 3.17. IOU ile yayaların ilişkilendirildiği çoklu yaya takibi akış diyagramı

3.8. Başarı Ölçüm Kriterleri ve MOT Kıyaslama

Bu çalışmada başarı, kesinlik ve saniye başına çerçeve (frame per second- fps) metrikleri tekli nesne takibinde performans ölçümü için kullanılmıştır. Çoklu nesne takibi çalışmasının performans değerlendirmesi ise MOT'ta bulunan metriklerin kullanımıyla sağlanmıştır. Kesinlik ölçütü, literatürdeki nesne takibi çalışmalarında yaygın olarak kullanılan bir değerlendirme ölçütüdür. Kesinlik ölçütü için hedef nesnenin tahmin konumu merkez noktası ile gerçek konum merkez noktası arasındaki fark Öklid yöntemi ile hesaplanmaktadır. Kesinlik performans ölçütü 20 piksel eşik değeri için hesaplanan mesafe ölçümleri içerisinden 20 sayısına eşit ya da küçük olanların sayısının toplam görüntü sayısına bölünmesi ile hesaplanır. İdeal nesne takibi yöntemlerinde istenen kesinlik değerinin yüksek olmasıdır. Ancak, takipçi sistem hedefi kaybettiğinde, çıktı konumu rasgele olabilir ve ortalama hata değeri izleme performansını saptırır.

Başarı metriği, tahmin edilen hedefin çevreleyici kutusu ve taban gerçeklik çevreleyici kutunun kesişiminin birleşimine oranı ile hesaplanmaktadır. Başarı performans ölçütü hesaplanan eğrinin altındaki alan (area under curve - AUC) ile hesaplanmaktadır. OTB-100 veri setinde yer alan 100 görüntü dizisi için hesaplanan kesinlik ve başarı değerlerinin ortalaması performans ölçütündeki kesinlik ve başarı değerini vermektedir. Çoklu nesne izleme doğruluğu ölçütü yanlış pozitifler, kaybolan hedefler ve kimlik değişimleri hata ölçümlerini birleştirir.

Tek geçiş değerlendirmesi, ilk alınan görüntü üzerinde hedef konumun taban gerçeklik verisi kullanılarak belirlenmesi ve algoritmanın bu konum ile başlatılıp ardışık görüntüler üzerinde çalıştırılması ile hesaplanır. Zamanda sağlamlık değerlendirmesinde, farklı bölümlere ayrılmış video çerçeveleri kullanılarak her bölüm için başarı oranı hesaplanır. Daha sonra bütün bölümler için toplam istatistiklerin ölçümü ile zamanda sağlamlık değerlendirmesi yapılır.

Uzayda sağlamlık değerlendirmesinde, başlangıç hedef çerçevesi kaydırılarak ya da ölçeklenerek nesne takibi gerçekleştirilir. Daha sonra zamanda sağlamlık değerlendirmesi yapılarak uzayda sağlamlık değerlendirilir.

4. BULGULAR VE TARTIŞMA

Bu çalışmada zorlu ortam koşullarında çoklu yaya takibi amaçlanmaktadır. Bu amaçla yapılan çalışma üç ana bölümden oluşmaktadır. İlk bölümde OTB-100 veri seti üzerinde farklı takip algoritmaları kullanılarak tekli nesne takibi çalışmaları gerçekleştirilmektedir. Böylelikle farklı algoritmaların aynı veri setleri üzerindeki başarı oranları karşılaştırılmaktadır. İkinci bölümde OTB-100 veri seti üzerinde çekirdek korelasyon filtresi kullanılarak farklı görünüm modellerinin takip başarısına etkisi analiz edilmektedir. Üçüncü bölümde 2D MOT 15 çoklu yaya takibi veri seti kullanılarak seçilen nesne tespiti, nesne takibi ve veri ilişkilendirme yöntemlerinin birleştirilmiş ve performans ölçümü gerçekleştirilmiştir.

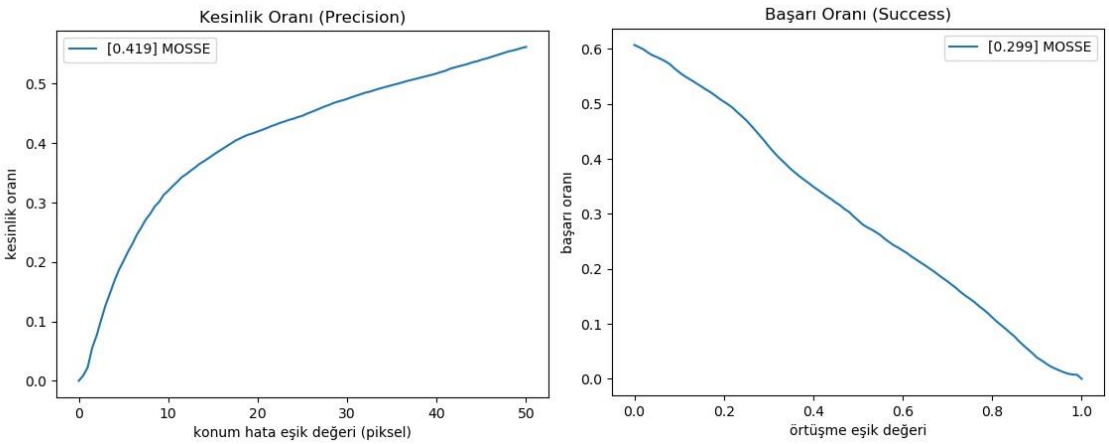
4.1. Farklı Takip Algoritmaları Kıyaslama

Bu bölümde OTB-100 içerisinde bulunan 100 farklı görüntü dizisi ile MOSSE, çekirdek korelasyon filtresi, Kalman filtresi ve hiyerarşik çekirdek korelasyon filtresi kullanılarak tekli nesne takibi gerçekleştirilmiştir. Bu yöntemlere ek olarak aynı veri seti üzerinde OpenCV kütüphanesinde bulunan Boosting, MIL, MedianFlow, TLD ve CSRT takip yöntemleri ile tekli nesne takibi gerçekleştirilmiştir. Sonuçlar hassasiyet ve başarı grafikleri ile her veri seti ve takip yöntemi için gösterilmektedir. Ayrıca tüm sonuçlar tek bir tablo üzerinde karşılaştırılmaktadır.

4.1.1.Mosse ile Tekli Nesne Takibi Sonuçları

MOSSE ile tekli nesne takibi çalışmasının OTB-100 veri seti için ortalama başarı ve kesinlik sonuçları sırasıyla Şekil 4.1'de gösterilmektedir. Kesinlik sonucu algoritmanın tahmin ettiği değerler ve gerçek değerler karşılaştırılarak merkez noktaları arasındaki uzaklık farkı ile hesaplanmaktadır. İki değer arasındaki farkın 0 olması durumunda tahmin edilen nesnenin çevreleyici kutusu ile gerçek değer tam örtüşür. Bu değer arttıkça tahminin gerçek değerden uzaklaştığı ve başarısının düştüğü anlamına gelmektedir. Başarı sonucu tahmin edilen çevreleyici kutu değerleri ve gerçek çevreleyici kutu değerleri arasındaki kesişimlerin birleşime oranı (IOU) yöntemi ile

hesaplanmaktadır. Hesaplanan değerin 1 olması durumunda tahmin edilen nesnenin çevreleyici kutusu ile gerçek değer tam örtüşür. Bu değer azaldıkça tahminin gerçek değerden uzaklaştığı ve başarısının düştüğü anlamına gelmektedir.



Şekil 4.1. MOSSE OTB-100 kesinlik ve başarı sonucu

Çizelge 4.1'de MOSSE yöntemi ile tekli nesne takibi sonuçları OTB-100 veri seti içerisinde bulunan 100 farklı görüntü dizisinin her biri için ayrı ayrı sayısal ifadelerle gösterilmektedir. Çizelgede P ile ifade edilen sütun kesinlik (precision) ölçümünü temsil etmektedir ve sonuçları 20 piksel eşik değeri için gösterilmektedir. Çizelgede S ile ifade edilen sütun başarı (success) ölçümünü temsil etmektedir ve sonuçları eğrinin altında kalan toplam alanı (area under curve - AUC) için gösterilmektedir. Ayrıca yöntemin hızını belirtmek için fps değerleri her veri seti için ifade edilmektedir. Ortalama kesinlik, başarı ve fps değerleri çizelgenin sonunda koyu ifadeler ile belirtilmektedir.

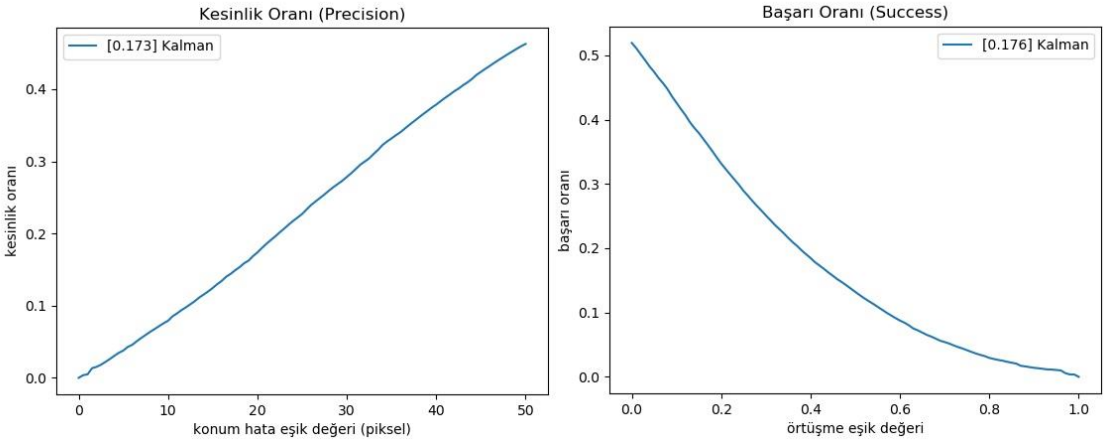
MOSSE yöntemi OTB-100 veri seti için saniyede ortalama 699 görüntü işlemektedir. Ortalama kesinlik ve başarısı ise sırasıyla 0,419 ve 0,299'dur. Kesinlik ve başarı sonuçları her veri seti için ayrı ayrı değerlendirildiğinde car2, car4, car24 ve carDark gibi Çizelge 4.1 P sütununda koyu renk ile ifade edilen görüntü dizilerinde MOSSE yönteminin başarılı olduğu görülmektedir. Ancak blurBody, human3 görüntü dizileriiçin ise en düşük başarı ve kesinlik sonucunu vermektedir. MOSSE yöntemi hızlı olmasına karşın nesnenin ani konum değişiklikleri ya da zor ortam şartlarında yöntemin başarı oranı düşmektedir.

Çizelge 4.1. MOSSE yöntemi OTB-100 sayısal sonuçlar

Dataset	P	S	Fps	Dataset	P	S	Fps	Dataset	P	S	Fps
basketball	0,025	0,071	826	diving	0,372	0,227	957	man	**1,0**	0,594	1400
biker	0,444	0,228	1119	dog	**1,0**	0,348	1142	matrix	0,12	0,097	614
bird1	0,324	0,193	616	dog1	0,685	0,44	1124	mhyang	0,999	0,786	823
bird2	0,485	0,498	362	doll	0,209	0,214	775	motorRolling	0,116	0,223	310
blurBody	0,009	0,02	181	dragonBaby	0,062	0,123	818	mountainBike	**1,0**	0,753	447
blurCar1	0,012	0,029	309	dudek	0,776	0,695	273	panda	0,616	0,229	1354
blurCar2	0,135	0,188	235	faceOcc1	0,131	0,298	345	redTeam	**1,0**	0,497	1291
blurCar3	0,076	0,1	526	faceOcc2	0,888	0,703	403	rubik	0,153	0,291	321
blurCar4	0,039	0,074	162	fish	0,769	0,754	619	shaking	0,033	0,144	422
blurFace	0,164	0,173	281	fleetFace	0,42	0,476	181	singer1	**1,0**	0,355	219
blurOwl	0,035	0,033	403	football	0,804	0,496	926	singer2	0,902	0,697	225
board	0,656	0,668	95	football1	0,162	0,123	1177	skater	0,588	0,525	622
bolt	0,017	0,02	796	freeman1	0,997	0,384	1500	skater2	0,131	0,198	415
bolt2	0,038	0,053	1075	freeman3	0,213	0,123	1440	skating1	0,765	0,447	833
box	0,119	0,115	359	freeman4	0,092	0,021	1553	skating21	0,055	0,214	305
boy	0,148	0,132	625	girl	0,632	0,242	1245	skating22	0,093	0,286	125
car1	0,741	0,139	1064	girl2	0,075	0,299	416	skiing	0,099	0,055	947
car2	**1,0**	0,689	998	gym	0,39	0,226	851	soccer	0,092	0,081	377
car4	**1,0**	0,432	1382	human2	0,02	0,059	184	subway	0,246	0,18	1176
car24	**1,0**	0,494	386	human3	0,004	0,008	467	surfer	0,053	0,028	1151
carDark	**1,0**	0,822	1188	human4	0,199	0,102	520	suv	0,533	0,524	758
carScale	0,651	0,417	1044	human5	0,238	0,142	592	sylvester	0,744	0,568	745
clifBar	0,167	0,165	1279	human6	0,279	0,198	607	tiger1	0,056	0,137	431
coke	0,144	0,123	525	human7	0,392	0,283	630	tiger2	0,107	0,09	346
couple	0,093	0,068	1156	human8	**1,0**	0,512	589	toy	0,601	0,387	682
coupon	**1,0**	**0,94**	637	human9	0,016	0,033	529	trans	0,371	0,551	104
crossing	0,508	0,246	1227	ironman	0,108	0,073	811	trellis	**1,0**	0,629	371
crowds	0,115	0,088	599	jogging1	0,221	0,432	925	twinnings	0,561	0,369	843
dancer	0,529	0,407	594	jogging2	0,163	0,123	669	vase	0,734	0,315	856
dancer2	**1,0**	0,765	678	jump	0,402	0,221	512	walking	0,738	0,32	413
david	**1,0**	0,52	789	jumping	0,15	0,085	1162	walking2	0,432	0,285	739
david2	**1,0**	0,56	1264	kiteSurf	0,488	0,302	1037	woman	0,101	0,079	1103
david3	0,405	0,282	392	lemming	0,167	0,128	294	**Ortalama**	**0,419**	**0,299**	**699**
deer	0,07	0,089	447	liquor	0,262	0,246	195				

4.1.2.Kalman Filtresi ile Tekli Nesne Takibi Sonuçları

Kalman filtresi ile tekli nesne takibi çalışmasının OTB-100 veri seti için ortalama başarı ve kesinlik sonuçları sırasıyla Şekil 4.2'de gösterilmektedir. Çizelge 4.2'de Kalman filtresi ile tekli nesne takibi sonuçları OTB-100 veri seti içerisinde bulunan 100 farklı görüntü dizisinin her biri için ayrı ayrı sayısal ifadelerle gösterilmektedir.



Şekil 4.2. Kalman OTB-100 kesinlik ve başarı sonucu

Çizelge 4.2. Kalman filtresi OTB-100 sayısal sonuçlar

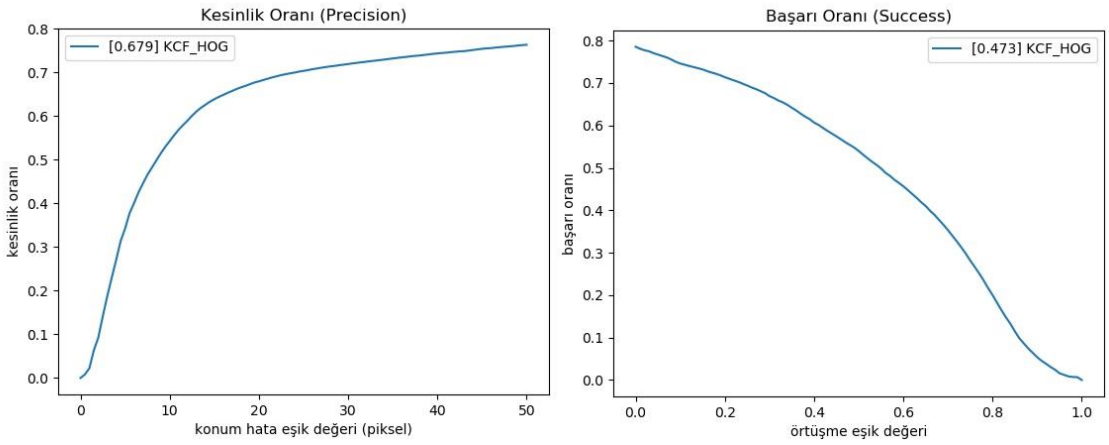
Dataset	P	S	Fps	Dataset	P	S	Fps	Dataset	P	S	Fps
basketball	0,076	0,074	8073	diving	0,544	0,345	8328	man	0,381	0,246	8792
biker	0,092	0,045	11723	dog	0,37	0,196	5737	matrix	0,16	0,156	6570
bird1	0,01	0,006	12339	dog1	0,176	0,151	10859	mhyang	0,299	0,293	11320
bird2	0,071	0,116	10957	doll	0,151	0,14	9048	motorRolling	0,049	0,155	9674
blurBody	0,015	0,086	9810	dragonBaby	0,239	0,256	16118	mountainBike	0,048	0,069	10800
blurCar1	0,065	0,272	9357	dudek	0,079	0,204	9511	panda	0,084	0,023	10342
blurCar2	0,089	0,332	9757	faceOcc1	0,232	0,585	11298	redTeam	0,357	0,142	9743
blurCar3	0,118	0,228	9105	faceOcc2	**0,596**	**0,587**	10000	rubik	0,153	0,331	7564
blurCar4	0,137	0,455	9992	fish	0,185	0,357	10431	shaking	0,036	0,116	10170
blurFace	0,037	0,21	8338	fleetFace	0,008	0,026	9895	singer1	0,017	0,032	9756
blurOwl	0,068	0,141	6844	football	0,061	0,078	11469	singer2	0,055	0,201	15327
board	0,021	0,099	10200	football1	0,054	0,046	10473	skater	0,425	0,429	12273
bolt	0,314	0,153	7423	freeman1	0,064	0,034	9000	skater2	0,103	0,222	10111
bolt2	0,143	0,101	9492	freeman3	0,143	0,054	12990	skating1	0,212	0,151	10280
box	0,047	0,091	8785	freeman4	0,173	0,029	7554	skating21	0,273	0,328	8410
boy	0,196	0,124	8839	girl	0,906	0,505	11692	skating22	0,08	0,208	9445
car1	0,082	0,089	12494	girl2	0,091	0,1	10053	skiing	0,049	0,023	11501
car2	0,541	0,372	12405	gym	0,167	0,113	10001	soccer	0,179	0,171	9534
car4	0,162	0,09	11318	human2	0,03	0,264	9600	subway	0,074	0,031	9677
car24	0,237	0,209	11249	human3	0,058	0,061	10817	surfer	0,04	0,019	9667
carDark	0,137	0,063	7568	human4	0,019	0,01	9797	suv	0,101	0,189	11179
carScale	0,052	0,04	7618	human5	0,049	0,019	8678	sylvester	0,248	0,243	11231
clifBar	0,54	0,314	8421	human6	0,04	0,02	9538	tiger1	0,124	0,231	11762
coke	0,052	0,176	6898	human7	0,304	0,227	12463	tiger2	0,008	0,022	8034
couple	0,093	0,041	9232	human8	0,141	0,074	9879	toy	0,17	0,144	10381
coupon	0,391	0,417	12483	human9	0,128	0,142	10845	trans	0,153	0,349	15369
crossing	0,117	0,045	15074	ironman	0,127	0,101	6379	trellis	0,371	0,323	10078
crowds	0,017	0,01	9381	jogging1	0,446	0,295	9010	twinnings	0,203	0,197	8937
dancer	0,133	0,309	11755	jogging2	0,547	0,465	9303	vase	0,373	0,253	13454
dancer2	0,513	0,439	10619	jump	0,041	0,107	17553	walking	0,017	0,009	5768
david	0,219	0,275	7935	jumping	0,316	0,178	7369	walking2	0,062	0,098	8019
david2	0,283	0,127	9977	kiteSurf	0,107	0,077	8274	woman	0,3	0,185	11106
david3	0,024	0,019	8103	lemming	0,022	0,056	9264				
deer	0,028	0,078	11732	liquor	0,427	0,543	8329	**Ortalama**	**0,173**	**0,176**	**10031**

Çizelgede P ile ifade edilen sütun kesinlik (precision) ölçümünü temsil etmektedir ve sonuçları 20 piksel eşik değeri için gösterilmektedir. Çizelgede S ile ifade edilen sütun başarı (success) ölçümünü temsil etmektedir ve sonuçları eğrinin altında kalan toplam alanı için gösterilmektedir. Ayrıca yöntemin hızını belirtmek için fps değerleri her veri seti için ifade edilmektedir. Ortalama kesinlik, başarı ve fps değerleri çizelgenin sonunda koyu ifadeler ile belirtilmektedir.

Kalman filtresinin en başarılı sonucu girl verisi için, en başarısız sonucu fleetFace verisi içindir. Kalman filtresi hızlı olmasına karşın nesnenin ani konum değişiklikleri, hızının aniden artması gibi durumlarda başarı oranı düşmektedir. Ayrıca yöntemin başarısı ortam ve kamera değişikliklerinde de düşmektedir.

4.1.3.Çekirdek Korelasyon Filtresi ile Tekli Nesne Takibi Sonuçları

Çekirdek korelasyon filtresi (KCF) ile tekli nesne takibi çalışmasının OTB-100 veri seti için ortalama kesinlik ve başarı sonuçları Şekil 4.3'te gösterilmektedir. Çekirdek korelasyon filtresi için kullanılan öznitelik yönelimli eğimlerin histogramı (HOG), takip fonksiyonu ise Gauss çekirdek yöntemidir.



Şekil 4.3. Çekirdek korelasyon filtresi OTB-100 kesinlik ve başarı sonucu

Çizelge 4.3'te çekirdek korelasyon filtresi ile tekli nesne takibi sonuçları OTB-100 veri seti içerisinde yer alan 100 farklı görüntü dizisinin her biri için için ayrı ayrı sayısal

ifadelerle gösterilmektedir. Çizelgede P ile ifade edilen sütun kesinlik (precision) ölçümünü temsil etmektedir ve sonuçları 20 piksel eşik değeri için gösterilmektedir. Çizelgede S ile ifade edilen sütun başarı (success) ölçümünü temsil etmektedir ve sonuçları eğrinin altında kalan toplam alanı için gösterilmektedir. Ayrıca yöntemin hızını belirtmek için fps değerleri her veri seti için ifade edilmektedir. Ortalama kesinlik, başarı ve fps değerleri çizelgenin sonunda koyu ifadeler ile belirtilmektedir.

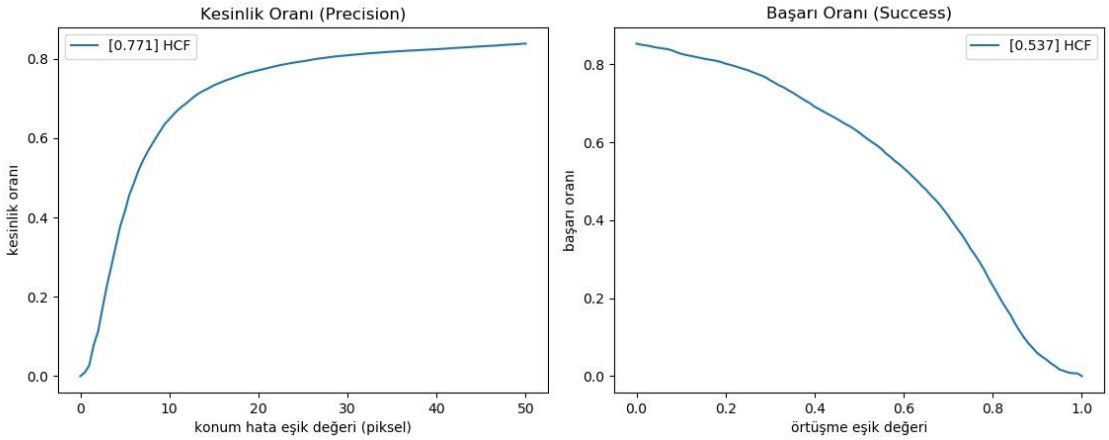
Çizelge 4.3. Çekirdek korelasyon filtresi OTB-100 sayısal sonuçlar

Dataset	P	S	Fps	Dataset	P	S	Fps	Dataset	P	S	Fps
basketball	0,931	0,645	171	diving	0,353	0,243	130	man	**1,0**	0,837	325
biker	0,451	0,247	660	dog	0,984	0,344	180	matrix	0,18	0,148	70
bird1	0,113	0,071	87	dog1	**1,0**	0,55	158	mhyang	**1,0**	0,797	61
bird2	0,465	0,562	55	doll	0,975	0,567	200	motorRolling	0,043	0,092	87
blurBody	0,557	0,408	48	dragonBaby	0,292	0,214	144	mountainBike	**1,0**	0,714	81
blurCar1	0,995	0,8	173	dudek	0,876	0,728	42	panda	0,509	0,27	478
blurCar2	0,998	0,76	128	faceOcc1	0,82	0,768	120	redTeam	**1,0**	0,501	173
blurCar3	0,992	0,814	92	faceOcc2	0,972	0,751	24	rubik	0,192	0,325	71
blurCar4	0,997	0,816	32	fish	**1,0**	0,839	57	shaking	0,033	0,043	95
blurFace	**1,0**	0,799	53	fleetFace	0,453	0,58	72	singer1	0,783	0,355	54
blurOwl	0,474	0,398	51	football	0,796	0,559	146	singer2	0,948	0,758	28
board	0,809	0,728	19	football1	0,986	0,758	265	skater	0,931	0,612	77
bolt	0,017	0,011	201	freeman1	0,387	0,214	474	skater2	0,692	0,568	32
bolt2	0,017	0,011	209	freeman3	0,911	0,327	1134	skating1	**1,0**	0,492	147
box	0,412	0,301	35	freeman4	0,53	0,168	939	skating21	0,366	0,356	63
boy	**1,0**	0,767	223	girl	0,872	0,592	207	skating22	0,469	0,384	39
car1	0,739	0,139	73	girl2	0,088	0,153	34	skiing	0,074	0,051	512
car2	**1,0**	0,684	158	gym	0,802	0,427	71	soccer	0,276	0,203	54
car4	**1,0**	0,425	622	human2	0,178	0,187	26	subway	**1,0**	0,782	249
car24	0,944	0,483	22	human3	0,034	0,025	40	surfer	0,976	0,489	630
carDark	**1,0**	0,605	568	human4	0,52	0,346	175	suv	0,979	0,884	146
carScale	0,806	0,42	262	human5	0,244	0,179	484	sylvester	0,843	0,65	90
clifBar	0,445	0,26	250	human6	0,29	0,213	264	tiger1	0,887	0,719	22
coke	0,931	0,574	132	human7	0,472	0,285	29	tiger2	0,849	0,666	96
couple	0,657	0,413	211	human8	**1,0**	0,512	178	toy	0,989	0,474	109
coupon	**1,0**	**0,936**	99	human9	0,849	0,393	117	trans	0,306	0,512	37
crossing	**1,0**	0,717	343	ironman	0,187	0,133	175	trellis	**1,0**	0,632	58
crowds	**1,0**	0,788	188	jogging1	0,231	0,183	185	twinnings	0,905	0,565	83
dancer	**1,0**	0,644	30	jogging2	0,163	0,12	22	vase	0,801	0,316	189
dancer2	**1,0**	0,774	21	jump	0,074	0,096	28	walking	**1,0**	0,53	200
david	**1,0**	0,527	103	jumping	0,339	0,273	310	walking2	0,648	0,479	55
david2	**1,0**	0,839	352	kiteSurf	0,357	0,257	568	woman	0,938	0,684	101
david3	**1,0**	0,774	119	lemming	0,275	0,228	50				
deer	0,873	0,675	36	liquor	0,403	0,407	112	**Ortalama**	**0,679**	**0,473**	**175**

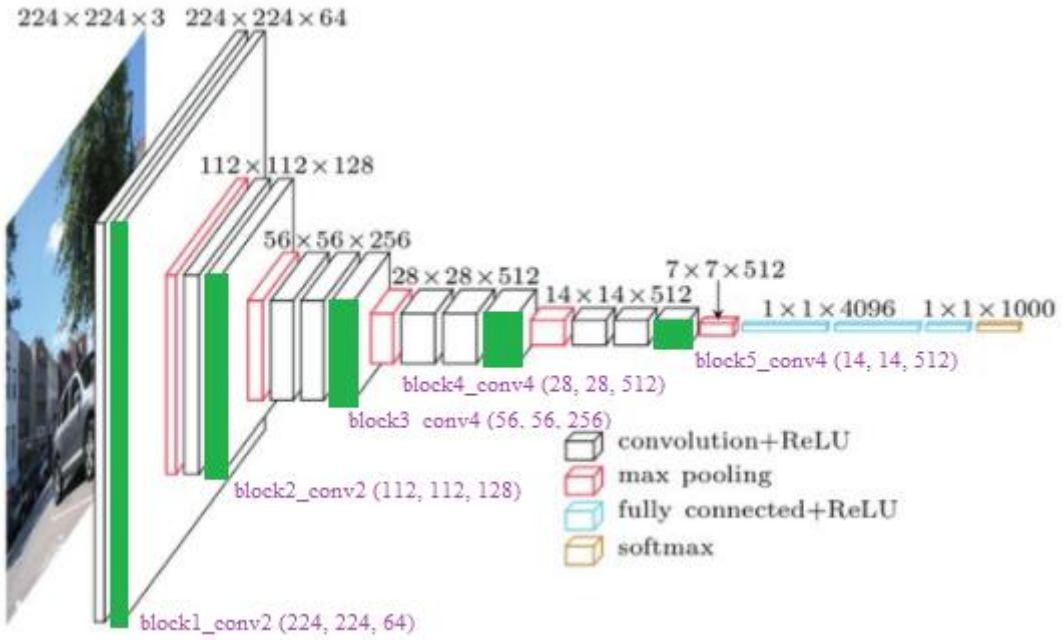
Kesinlik ve başarı sonuçları her veri seti için ayrı ayrı değerlendirildiğinde david, human8, dog1 ve walking gibi Çizelge 4.3 P sütununda koyu renk ile ifade edilen görüntü dizilerinde KCF yönteminin başarılı olduğu görülmektedir. KCF yöntemi saniyede ortalama 175 görüntü işleme gücü ile hızlı olmasına karşın nesnenin ani konum değişiklikleri ya da zor ortam şartlarında başarı oranı düşmektedir. Yöntemin en başarısız olduğu görüntü dizileri bolt ve bolt2 olarak görülmektedir. HOG özniteliği nesneyi ayırt etmek amacıyla sınıflandırma çalışmalarında sıkça kullanılmasına karşın görüntü üzerinde benzer nesnelerin bulunması durumunda başarısı düşmektedir.

4.1.4.Hiyerarşik Çekirdek Korelasyon Filtresi ile Tekli Nesne Takibi Sonuçları

Hiyerarşik çekirdek korelasyon filtresi ile tekli nesne takibi çalışmasının OTB-100 veri seti için ortalama kesinlik ve başarı sonuçları Şekil 4.4'te gösterilmektedir. Çekirdek korelasyon filtresi ile kullanılan öznitelikler VGG-19 ağının evrişim katmanlarından gelen özelliklerdir. VGG-19 ImageNet ile eğitilmiş ve özellik çıkarma amacıyla kullanılan bir derin öğrenme modelidir. Hiyerarşik çekirdek korelasyon filtresi ile nesne takibi yönteminde çekirdek korelasyon filtresi ile Şekil 4.5'te belirtilen block1_conv2, block2_conv2, block3_conv4, block4_conv4, block5_conv4 evrişim katmanlarından son üç bloğun evrişim katman çıktıları birlikte kullanılmaktadır. Bu çıktılar sırasıyla block3_conv4, block4_conv4, block5_conv4 çıktılarıdır ve hiyerarşik yapıda kullanılmaktadır. Evrişim katmanlarının tercih edilme nedeni uzamsal çözünürlüğün korunmak istenmesi ve nesne takibi için faydalı öznitelik oluşturmasıdır. Katman çıktı boyutları şekilde ifade edildiği gibi 3. katmanın son evrişim çıktısı için 56x56x256, 4. katmanın son evrişim çıktısı için 28x28x512 ve 5. katmanın son evrişim çıktısı için 14x14x512 boyutundadır. Nesne takibinde 3 katmanın her biri için korelasyon çıktısı oluşturulmakta ve nesnenin konumunun tahmini için üç çıktı hiyerarşik yapıda kullanılmaktadır. Sırasıyla kosinüs çarpımı ile sınır belirsizliği ortadan kaldırılan ve normalize edilen block3_conv4, block4_conv4, block5_conv4 katman çıktıları 0,02, 1, 0,5 katsayıları ile çarpılarak toplandıktan sonra takip edilen nesnenin sonraki adımdaki konumu tespit edilmektedir. Bu şekilde ağırlık değerleri ile çarpılıp daha sonra toplanmalarının nedeni hedef konuma en başarılı korelasyon çıktısının etki etmesinin sağlanmasıdır.



Şekil 4.4. Hiyerarşik çekirdek korelasyon filtresi OTB-100 kesinlik ve başarı sonucu



Şekil 4.5. VGG-19 katmanları

Çizelge 4.4'te sonuçlar OTB-100 veri setinde bulunan 100 farklı görüntü dizisinin her biri için ayrı ayrı sayısal ifadelerle gösterilmektedir. Çizelgede P ile ifade edilen sütun kesinlik (precision) ölçümünü temsil etmektedir ve sonuçları 20 piksel eşik değeri için gösterilmektedir. Çizelgede S ile ifade edilen sütun başarı (success) ölçümünü temsil etmektedir ve sonuçları eğrinin altında kalan toplam alanı için gösterilmektedir. Ayrıca

yöntemin hızını belirtmek için fps değerleri her veri seti için ifade edilmektedir. Ortalama kesinlik, başarı ve fps değerleri çizelgenin sonunda koyu ifadeler ile belirtilmektedir.

Çizelge 4.4. Hiyerarşik çekirdek korelasyon filtresi OTB-100 sayısal sonuçlar

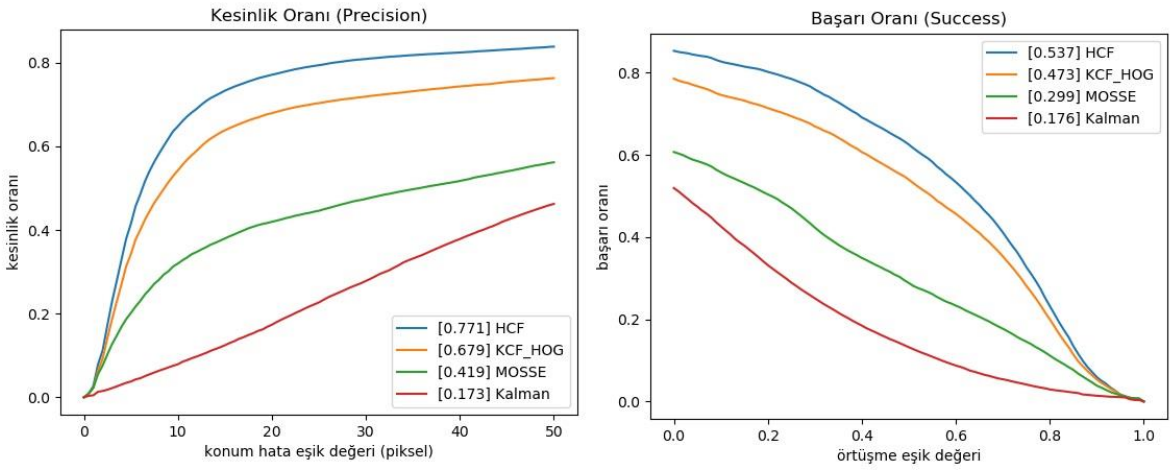
Dataset	P	S	Fps	Dataset	P	S	Fps	Dataset	P	S	Fps
basketball	**1,0**	0,821	2	diving	0,251	0,221	2	man	**1,0**	0,837	6
biker	0,444	0,244	9	dog	**1,0**	0,364	3	matrix	0,35	0,254	3
bird1	0,591	0,36	4	dog1	0,997	0,546	3	mhyang	**1,0**	0,81	1
bird2	0,98	0,832	1	doll	0,967	0,568	3	motorRolling	0,872	0,537	1
blurBody	0,97	0,712	1	dragonBaby	0,717	0,536	2	mountainBike	**1,0**	0,703	1
blurCar1	0,996	0,804	3	dudek	0,901	0,736	1	panda	0,449	0,219	7
blurCar2	0,968	0,764	2	faceOcc1	0,659	0,74	2	redTeam	**1,0**	0,467	6
blurCar3	**1,0**	0,817	1	faceOcc2	0,991	0,755	0	rubik	0,964	0,612	1
blurCar4	**1,0**	0,823	0	fish	**1,0**	0,833	1	shaking	0,033	0,032	1
blurFace	**1,0**	0,793	2	fleetFace	0,58	0,596	1	singer1	0,994	0,354	1
blurOwl	0,945	0,773	1	football	**1,0**	0,697	3	singer2	0,036	0,039	0
board	0,761	0,675	0	football1	**1,0**	0,806	5	skater	**1,0**	0,625	1
bolt	**1,0**	0,808	4	freeman1	0,399	0,208	7	skater2	0,768	0,601	0
bolt2	0,017	0,011	3	freeman3	0,837	0,306	13	skating1	**1,0**	0,514	2
box	0,395	0,29	0	freeman4	0,307	0,157	12	skating21	0,52	0,434	1
boy	**1,0**	0,783	4	girl	**1,0**	0,713	4	skating22	0,395	0,364	0
car1	0,741	0,14	1	girl2	0,073	0,064	0	skiing	0,198	0,115	7
car2	**1,0**	0,685	2	gym	0,235	0,111	1	soccer	0,878	0,425	1
car4	**1,0**	0,426	8	human2	0,655	0,675	0	subway	**1,0**	0,811	5
car24	0,998	0,493	1	human3	0,035	0,025	1	surfer	**1,0**	0,498	8
carDark	**1,0**	0,664	8	human4	0,793	0,497	3	suv	0,979	0,834	2
carScale	0,722	0,42	5	human5	0,245	0,184	7	sylvester	0,854	0,672	1
clifBar	0,515	0,347	4	human6	0,304	0,209	5	tiger1	0,825	0,683	0
coke	0,852	0,546	2	human7	**1,0**	0,486	1	tiger2	0,575	0,577	1
couple	0,814	0,521	4	human8	**1,0**	0,506	3	toy	0,889	0,474	2
coupon	**1,0**	**0,935**	1	human9	**1,0**	0,395	2	trans	0,331	0,52	0
crossing	**1,0**	0,714	5	ironman	0,645	0,443	3	trellis	**1,0**	0,616	1
crowds	**1,0**	0,796	4	jogging1	0,99	0,723	3	twinnings	0,998	0,603	1
dancer	**1,0**	0,64	1	jogging2	**1,0**	0,773	0	vase	0,598	0,315	3
dancer2	**1,0**	0,792	1	jump	0,066	0,14	0	walking	**1,0**	0,553	4
david	**1,0**	0,543	1	jumping	0,99	0,693	6	walking2	0,616	0,491	1
david2	**1,0**	0,814	6	kiteSurf	0,464	0,349	8	woman	0,938	0,688	2
david3	**1,0**	0,779	2	lemming	0,275	0,226	1	**Ortalama**	**0,771**	**0,537**	**3**
deer	**1,0**	0,751	0	liquor	0,99	0,857	2				

Hiyerarşik çekirdek korelasyon filtresi ortalama 0,763 kesinlik ve 0,532 başarı değeri ile deneylerde kullanılan 4 yöntem arasında en başarılı sonucu veren nesne takibi yöntemidir. Ancak derin öğrenme modeli kullanarak hedef nesneyi derin özellik ile

modellemesi saniyede işlenen görüntü sayısını oldukça düşürmektedir. OTB-100 veri seti için ortalama saniyede 3 görüntü işlemesi nesne takibi çalışmalarında yüksek işlem kapasiteli bilgisayarlara ihtiyaç duyduğunu göstermektedir.

4.1.5.Tekli Nesne Takibi Sonuçlarını Karşılaştırma

Bu bölümde deneyleri gerçekleştirilen MOSSE, Kalman, çekirdek korelasyon filtresi ve hiyerarşik çekirdek korelasyon filtresi yöntemlerinin OTB-100 veri seti için ortalama kesinlik ve başarı sonuçları karşılaştırılmaktadır. Şekil 4.6'da kesinlik oranı grafiği ve başarı oranı grafiğinde sonuçlar 4 yöntem için birlikte gösterilmektedir. Şekilde ifade edilen HCF bölüm 4.1.4'te bahsedilen VGG19 ağı ile hiyerarşik çekirdek korelasyon filtresi yöntemini temsil etmektedir. Şekilde ifade edilen KCF_HOG bölüm 4.1.3'te bahsedilen çekirdek korelasyon filtresinin OG özniteliği ile kullanılması yöntemini temsil etmektedir. Benzer şekilde MOSSE bölüm 4.1.1'de bahsedilen MOSSE nesne takibi yöntemini ve Kalman bölüm 4.1.2'de bahsedilen Kalman filtresi ile nesne takibi yöntemini temsil etmektedir.



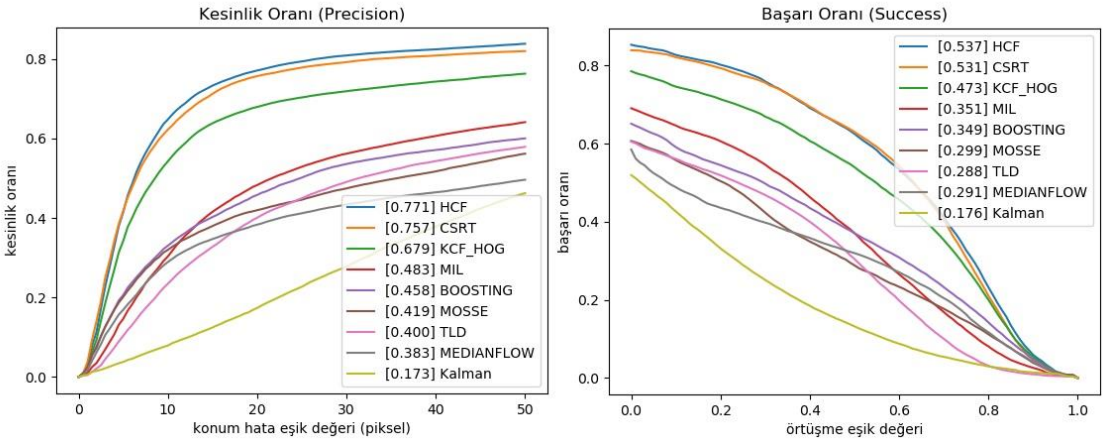
Şekil 4.6. Tekli nesne takibi karşılaştırma

Sonuçlar değerlendirildiğinde HCF ve KCF_HOG yöntemlerinin daha başarılı olduğu görülmektedir. Nesne takip yönteminin başarısı takip edilen nesne için seçilen öznitelik ve ortam şartlarına bağlı olarak değişiklik göstermektedir. Takip edilen nesne için

seçilen özniteliğin nesneyi ortamdan ayırt edebilmesi başarı oranını arttırmaktadır. Karşılaştırmalar değerlendirildiğinde OTB-100 veri seti için en başarılı sonucu 0,771 başarı kesinlik ve 0,537 başarı oranı ile hiyerarşik çekirdek korelasyon filtresinin verdiği görülmektedir. Ancak ortalama 3 fps değeri ile bu yöntemin gerçek zamanlı nesne takibi uygulamalarında kullanılması tercih edilmemektedir. Başarı sonuçları ve fps sonucu optimum olan HOG özniteliği ile çekirdek korelasyon filtresi ise gerçek zamanlı çalışmalarda tercih edilmektedir.

4.1.6.OpenCV Takip Yöntemleri ile Karşılaştırma

Bu bölümde OpenCV'de bulunan takip yöntemleri ile bu çalışmada tekli nesne takibinde değerlendirilen MOSSE, Kalman, çekirdek korelasyon filtresi ve hiyerarşik çekirdek korelasyon filtresi yöntemleri karşılaştırılmaktadır. Şekil 4.7'de kesinlik oranı ve başarı oranı grafiğinde sonuçlar OpenCV kütüphanesinde bulunan CSRT, MIL, BOOSTING, TLD, MedianFlow yöntemleri ile MOSSE, Kalman, çekirdek korelasyon filtresi ve hiyerarşik çekirdek korelasyon filtresi yöntemleri için birlikte gösterilmektedir. Yöntemlerin fps değerleri en hızlı olan yöntemden en yavaş olana doğru sırasıyla Çizelge 4.5'te gösterilmektedir.



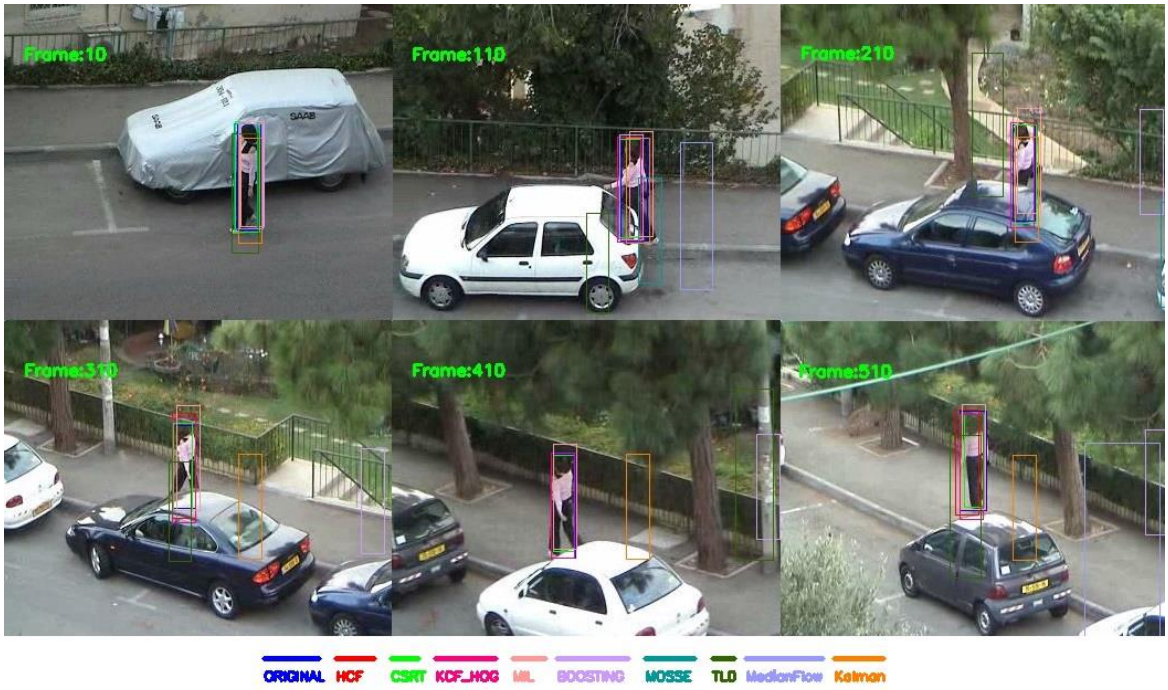
Şekil 4.7. OpenCV yöntemleri ile tekli nesne takibi karşılaştırma

Çizelge 4.5. Tekli nesne takibi fps karşılaştırma

Yöntem	fps	Yöntem	fps	Yöntem	fps
Kalman	10031	KCF_HOG	175	TLD	50
MedianFlow	1395	BOOSTING	68	MIL	21
MOSSE	699	CSRT	62	HCF	3

Sonuçlar değerlendirildiğinde en başarılı yöntem HCF iken en başarısız yöntem Kalman filtresidir. Kalman filtresi 10031 fps ile en yüksek saniye başına çerçeve değerine sahip olmasına rağmen yüksek başarı oranı gerektiren çalışmalarda tercih edilmemektedir. HCF için evrişimli sinir ağının hiyerarşik katmanlarından gelen semantik ve uzaysal özelliklerin bir arada kullanılması ile nesne takibinde başarı oranının arttığı görülmektedir. Ancak VGG-19 ağının üç evrişim katmanı ile üç korelasyon filtresini çalıştırması fps değerini düşürdüğünden HCF yüksek işlem gücüne sahip GPU'lu bilgisayarın mevcut olması durumunda gerçek zamanlı uygulamalarda tercih edilebilmektedir. OpenCV kütüphanesindeki yöntemlerden en başarılısı olan CSRT ortalama 0.757 kesinlik ve 0.531 başarı değerleri ile 2. en başarılı sonucu vermektedir ve 62 fps değeri ile gerçek zamanlı işlem gücü kapasitesine sahiptir. KCF_HOG yöntemi ise 175 fps ile CSRT yönteminden yaklaşık üç kat hızlı çalışmaktadır. MIL yöntemi ise düşük kesinlikleri ve 21 fps hızı ile yüksek başarı ve gerçek zamanlı çalışma gerektiren nesne takibi çalışmalarında tercih edilmemektedir. BOOSTING yöntemi, MOSSE'ye kıyasla daha başarılı sonuç vermektedir, ancak MOSSE yaklaşık 10 kat daha hızlı çalışmaktadır. TLD yönteminin başarı sonucu MOSSE'ye yakın iken yaklaşık 14 kat daha yavaş çalışmaktadır. MEDIANFLOW yöntemi ise yüksek 1395 fps işleyebilmesine rağmen Kalman filtresinden sonraki en başarısız yöntemdir.

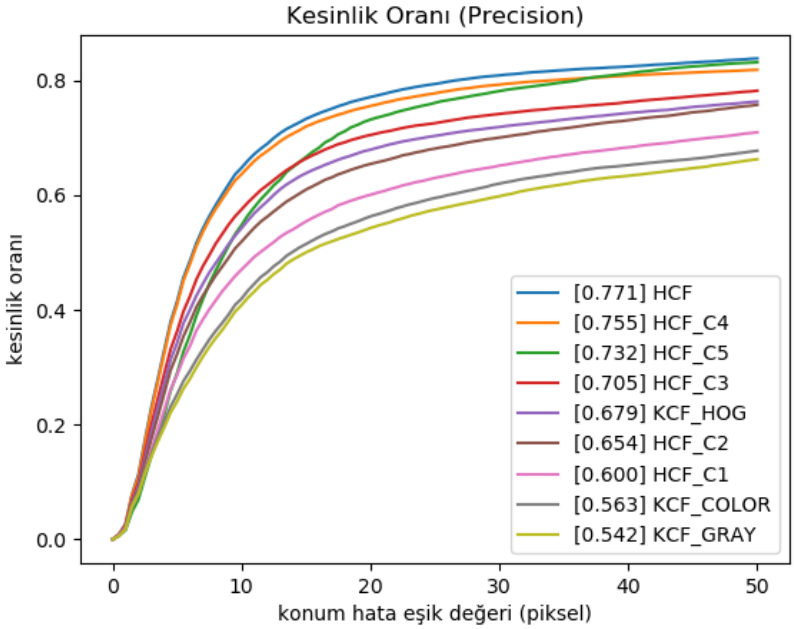
OTB-100'de bulunan ve 597 görüntü dizisine sahip woman veri seti için tekli nesne takibi sonuçları Şekil 4.8'de gösterilmektedir. Sırasıyla 10, 110, 210, 310, 410 ve 510. görüntüler için veri setinin sağladığı gerçek değer ile HCF, CSRT, KCF_HOG, BOOSTING, MIL, TLD, Kalman, MOSSE ve MedianFlow yöntemlerinin tahmin ettiği çevreleyici kutular görselde ifade edilmektedir.



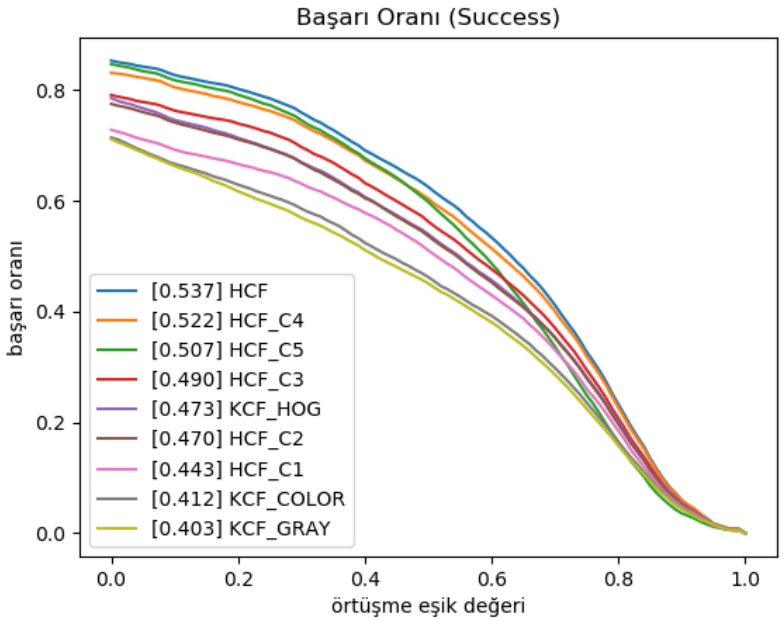
Şekil 4.8. Tekli nesne takibi woman veri seti karşılaştırma

4.2. Farklı Özniteliklerin Takibe Etkisi

Bu bölümde çekirdek korelasyon filtresi kullanılarak seçilen özniteliğin takip başarısına etkisi analiz edilmektedir. Analiz için kullanılan öznitelikler görüntü gri bilgisi, görüntü RGB renk bilgisi, HOG özelliği ve VGG-19 ağı evrişim katman çıktılarıdır. Şekil 4.5'te gösterilen block1_conv2, block2_conv2, block3_conv4, block4_conv4, block5_conv4 katman çıktılarının farklı öznitelik olarak kullanılması ile toplamda 8 öznitelik analiz edilmektedir. Şekil 4.9'da farklı öznitelikler için OTB-100 veri seti ile nesne takibi kesinlik sonucu gösterilmektedir. Şekil 4.10'da ise başarı sonucu gösterilmektedir. Sonuçlar en yüksek başarı ve kesinlik oranına sahip olan öznitelikten en düşük olana doğru sıralı şekilde grafikte ifade edilmektedir. Grafiklerde ifade edilen HCF_C1 Şekil 4.5'te belirtilen block1_conv2, HCF_C2 block2_conv2, HCF_C3 block3_conv4, HCF_C4 block4_conv4, HCF_C5 block5_conv4 özniteliği ile çekirdek korelasyon filtresi yöntemini temsil etmektedir. KCF_HOG yönelimli eğimlerin histogramı ile çekirdek korelasyon filtresi, KCF_COLOR RGB renk bilgisi ile çekirdek korelasyon filtresi, KCF_GRAY gri bilgisi ile çekirdek korelasyon filtresini temsil etmektedir. Sonuçlar OTB-100 veri seti kullanılarak elde edilmiştir.



Şekil 4.9. Çekirdek korelasyon filtresi ile farklı görünüm modellerinin karşılaştırması kesinlik sonucu



Şekil 4.10. Çekirdek korelasyon filtresi ile farklı görünüm modellerinin karşılaştırması başarı sonucu

Sonuçlara bakıldığında HOG, gri ve renk bilgisi özniteliklerinden en başarılı sonucu HOG özniteliğinin kullanımı ile elde edildiği görülmektedir. VGG-19 ile elde edilen derin öznitelikler değerlendirildiğinde ise en başarılı sonucu 4. bloğun evrişim çıktısının verdiği görülmektedir. Derin özniteliklerden 4., 5., ve 3. blokların evrişim katman çıktılarının ilk bloklardaki evrişim katman çıktılarına kıyasla daha başarılı olduğu görülmektedir. Böylelikle VGG-19 derin öğrenme ağındaki evrişim katmanlarının son bloğa doğru ilerledikçe nesne takibi başarısı arttırdığı varsayımı yapılabilmektedir. Çizelge 4.6'da farklı görünüm modellerinin fps karşılaştırması bulunmaktadır. Farklı görünüm modellerine bakıldığında en hızlı görünüm modeli KCF_GRAY iken en yavaş olanın HCF olduğu görülmektedir.

Çizelge 4.6. Farklı görünüm modelleri fps karşılaştırma

Yöntem	fps	Yöntem	fps	Yöntem	fps
KCF_GRAY	216	HCF C1	12	HCF C4	5
KCF_HOG	175	HCF C2	9	HCF_C5	5
KCF_COLOR	104	HCF C3	7	HCF	3

4.3. Çoklu Yaya Takibi

Bu bölümde çoklu yaya takibi test sonuçları verilmektedir. Şekil 3.15'te gösterildiği gibi çoklu yaya takibi yayanın derin özellikleri ya da renk histogram özellikleri ile birlikte tespit edilmesi, yayanın HOG özelliği kullanılarak çekirdek korelasyon filtresi ile takip edilmesi ve takip edilen yayalar ile yeni tespit edilen yayaların ilişkilendirilmesi olmak üzere üç kısımdan oluşmaktadır. Çizelge 3.1'de gösterilen 2D MOT 15 veri setinin eğitim bölümünde yer alan 11 farklı görüntü dizisi ile uygulama test edilmektedir. Performans sonuçları Çizelge 2.4'te açıklanan MOTA, MOTP, MT, ML, IDs, FM, FP ve FN ile ifade edilen çoklu nesne takibi başarı kriterleri cinsinden verilmektedir. Yaya tespiti için Yolov4 kullanılmakta, yaya takibi için ise hızlı ve başarılı sonuç üreten HOG özelliklerine dayalı çekirdek korelasyon filtresi kullanılmaktadır.

Bu bölümde tespit edilen yeni yayalar ile önceden takip edilen yayalar arasında ilişki kurmak için üç farklı yöntem kullanılarak analiz edilmektedir. Bunlardan ilki Yolov4 ile tespit edilen yayanın 128x1 boyutlu derin özellik vektörü kullanılarak yayaların ilişkilendirilmesidir. Takip edilen yayanın hangi yeni tespit edilen yayaya ait olduğunu bulmak için tespit edilen derin özellikler ile takip edilen yayanın derin özellikleri ilişkilendirilir. İlişkilendirilmede kullanılan uzaklık ölçütü kosinüs uzaklığıdır. Her takip edilen yayanın derin özellikleri ile yeni tespit edilen yayaların derin özellikleri arasında kosinüs uzaklığı hesaplanır. Daha sonra Macar algoritması kullanılarak maliyet analizi yapılır ve ilişkilendirmeler gerçekleştirilir.

İkinci yöntemde ilişkilendirme için kullanılan özellik RGB renk histogram bilgisidir. Histogram için kutu sayısı 32 olarak kullanılmaktadır ve elde edilen özellik 96x1 boyutlu RGB renk histogram bilgisidir. Yolov4 ile tespit edilen yayanın çevreleyici kutusu içerisindeki renk histogramı hesaplanır. Takip edilen yayanın hangi yeni tespit edilen yayaya ait olduğunu bulmak için tespit edilen yayanın histogram bilgisi ile takip edilen yayanın histogram bilgisi ilişkilendirilir. İlişkilendirilmede kullanılan uzaklık ölçütü ilk yöntemde de kullanılan kosinüs uzaklığıdır. Her takip edilen yayanın histogram bilgisi ile yeni tespit edilen yayaların histogram bilgileri arasında kosinüs uzaklığı hesaplanır. Daha sonra Macar algoritması kullanılarak maliyet analizi yapılır ve ilişkilendirmeler gerçekleştirilir.

Üçüncü yöntemde ilişkilendirme için çevreleyici kutular kullanılmaktadır. Yayalar Yolov4 kullanılarak tespit edilmektedir. Takip edilen yayanın hangi yeni tespit edilen yayaya ait olduğunu bulmak için çevreleyici kutular üzerinden IOU metriği hesaplaması gerçekleştirilir. Her takip edilen yayanın çevreleyici kutu bilgisi ile yeni tespit edilen yayaların çevreleyici kutu bilgileri arasında IOU değerleri hesaplanır. IOU maliyet hesabı yapılırken kutular arası hesaplanan IOU değerleri 1'den çıkartılır. Bunun nedeni önceki iki yöntemde kosinüs mesafesi ile hesaplanan değer 0'a yaklaştıkça benzerliğin arttığı anlaşılırken üçüncü yöntemde hesaplanan IOU değeri 0'a yaklaştıkça benzerliğin azaldığı anlaşılmaktadır. Bu nedenle IOU sonucu 1'den çıkartılarak değer 0'a yaklaştığında benzerliğin artması sağlanır. Böylece üç yöntemde de veriler arasındaki ilişki arttıkça kullanılan metriğin azaldığı varsayılmaktadır. İlişkilendirilme bu IOU

tabanlı mesafe ölçütü üzerinden maliyet analizini çözen Macar algoritması kullanılarak gerçekleştirilir.

4.3.1.Derin Özellikler ile Veri İlişkilendirme

Çoklu yaya takibi çalışmasının ilk yöntemi bu bölümde değerlendirilmektedir. 2D MOT 15 veri seti içerisindeki görüntü dizileri için sonuçlar Çizelge 4.7'de gösterilmektedir. Yöntemin adımları sırasıyla aşağıdaki maddelerde belirtilmektedir.

1. Yolov4 ile ilk görüntüdeki yayalar tespit edilir. Aralığı 0 ile 1 arasında değişen güven değeri için eşik değeri 0,3 olarak belirlenir. Yolov4 ile tespit edilen yayaların güven değeri 0,3'ten büyük ise bu yayalar seçilir, küçük ise seçilmez.
2. Seçilen yayaların 128x1 boyutlu derin özellik vektörü ile takip edilecek yayalar temsil edilir.
3. İkinci görüntüdeki yayalar yine Yolov4 ve 0,3 eşik güven değeri ile bulunur.
4. Önceden oluşturulan takip edilecek yayalar HOG tabanlı çekirdek korelasyon filtresi kullanılarak takip ettirilir ve yayaların ikinci görüntüdeki tahmini konumu hesaplanır.
5. Üçüncü adımda tespit edilen yayalar ile dördüncü adımda takip edilen yayaların ilişkilendirilmesi gerçekleştirilir. İlişkilendirme için derin özellikler, kosinüs mesafesi metriği ve Macar ilişkilendirme algoritması kullanılır.
6. Bir sonraki görüntü üzerinde benzer şekilde yayalar Yolov4 ile tespit edilir ve önceki takip edilen yayaların HOG tabanlı çekirdek korelasyon filtresi takip algoritması ile yeni konumu hesaplanır. Daha sonrasında aynı ilişkilendirme adımı gerçekleştirilir.
7. Son görüntü de tamamlanana kadar altıncı adım tekrar edilir.

Ek olarak eğer takip edilen yaya art arda 10 görüntüde tespit edilemediyse ve ilişkilendirilmediyse bu yaya takip edilecek yayalar arasından çıkartılır. Ayrıca tespit yönteminin başarısı takip yöntemini doğrudan etkilemektedir. Tespit yönteminin başarısı arttıkça çoklu yaya takibi uygulamasının başarısı artmakta, azaldıkça çoklu yaya takibi uygulamasının başarısı azalmaktadır.

Çizelge 4.7. Derin özellikler ile kosinüs mesafesi kullanılarak veri ilişkilendirme yapıldığında çoklu yaya takibi sonuçları

	MOTA↑	MOTP↑	MT↑	ML↓	IDs↓	FM↓	FP↓	FN↓
ADL-Rundle-6	32,6	0,281	0	5	120	158	583	2672
ADL-Rundle-8	13,5	0,271	5	13	70	121	1138	4659
ETH-Bahnhof	30,4	0,265	**24**	135	188	235	713	4437
ETH-Pedcross2	3,6	0,281	0	135	10	27	111	6396
ETH-Sunnyday	17,1	0,236	0	21	18	30	55	1503
KITTI-13	11,2	0,288	2	16	**9**	26	214	602
KITTI-17	41,4	0,288	0	1	**9**	**18**	**45**	404
PETS09-S2L1	**75,5**	0,287	17	**0**	75	127	567	495
TUD-Campus	37,6	0,279	1	1	10	21	51	**163**
TUD-Stadtmitte	54,6	**0,346**	5	**0**	19	28	144	362
Venice-2	23,2	0,267	3	9	93	141	976	4416
Genel	**27,4**	**0,278**	**57**	**336**	**621**	**932**	**4597**	**26109**

Sonuçlar değerlendirildiğinde farklı başarı değerlendirme ölçütleri için her görüntü dizisinin farklı sonuçlar verdiği görülmektedir. Yüksek olması beklenen MOTA, MOTP ve MT değerlendirme ölçütleri incelendiğinde en başarılı sonucu sırasıyla PETS09-S2L1, TUD-Stadtmitte ve ETH-Bahnhof dizileri vermektedir. Düşük olması beklenen ML değerlendirme ölçütü için en başarılı sonucu PETS09-S2L1 ve TUD-Stadtmitte, IDs için KITTI-13 ve KITTI-17, FM için KITTI-17, FP için KITTI-17, FN için ise TUD-Campus görüntü dizilerinin verdiği görülmektedir.

MT, ML, IDs, FM, FP, FN ölçütleri görüntü dizisinde yer alan takip edilen yaya sayısına göre değişiklik göstermektedir. Bu nedenle başarı sonuçlarını MOTA, MOTP ölçütlerine göre değerlendirmek daha anlamlı olmaktadır. Ancak IDs ölçütü çoklu nesne takibi çalışmalarında değerlendirme için yaygın kullanılan ölçütlerdendir. Çoklu nesne takibinde en büyük problemlerden bir tanesi takip edilen nesnelerin kimlik bilgilerinin değişmesidir. IDs ölçütü kimlik değişikliği bilgisini içerdiği için başarı değerlendirmesinde bunu kullanmak anlamlı olmaktadır. Bu durumda başarıların en iyilendiği görüntü dizilerinin PETS09-S2L1, TUD-Stadtmitte, KITTI-13 ve KITTI-17 olduğu söylenebilir. Yayaların birbirinden uzakta olduğu ve birbirini örtmediği

durumlarda uygulama başarısı artmakta, aksi durumlarda uygulama başarısı azalmaktadır.

4.3.2.Renk Histogramı Özellikleri ile Veri İlişkilendirme

Çoklu yaya takibi çalışmasının ikinci yöntemi bu bölümde değerlendirilmektedir. 2D MOT 15 veri seti içerisindeki görüntü dizileri için sonuçlar Çizelge 4.8'de gösterilmektedir. Yöntemin adımları sırasıyla aşağıdaki maddelerde belirtilmektedir.

1. Yolov4 ile ilk görüntüdeki yayalar tespit edilir. Aralığı 0 ile 1 arasında değişen güven değeri için eşik değeri 0,3 olarak belirlenir. Yolov4 ile tespit edilen yayaların güven değeri 0,3'ten büyük ise bu yayalar seçilir, küçük ise seçilmez.
2. Seçilen yayaların 32 kutu sayısı ile elde edilen 96x1 boyutlu RGB histogram özellik vektörü ile takip edilecek yayalar temsil edilir.
3. İkinci görüntüdeki yayalar yine Yolov4 ve 0,3 eşik güven değeri ile bulunur.
4. Önceden oluşturulan takip edilecek yayalar HOG tabanlı çekirdek korelasyon filtresi kullanılarak takip ettirilir ve yayaların ikinci görüntüdeki tahmini konumu hesaplanır.
5. Üçüncü adımda tespit edilen yayalar ile dördüncü adımda takip edilen yayaların ilişkilendirilmesi gerçekleştirilir. İlişkilendirme için RGB histogram özellikleri, kosinüs mesafesi metriği ve Macar ilişkilendirme algoritması kullanılır.
6. Bir sonraki görüntü üzerinde benzer şekilde yayalar Yolov4 ile tespit edilir ve önceki takip edilen yayaların HOG tabanlı çekirdek korelasyon filtresi takip algoritması ile yeni konumu hesaplanır. Daha sonrasında aynı ilişkilendirme adımı gerçekleştirilir.
7. Son görüntü de tamamlanana kadar altıncı adım tekrar edilir.

İlk yönteme benzer şekilde eğer takip edilen yaya art arda 10 görüntüde tespit edilemediyse ve ilişkilendirilmediyse bu yaya takip edilecek yayalar arasından çıkartılır.

Çizelge 4.8. Renk histogramı özellikleri ile kosinüs mesafesi kullanılarak veri ilişkilendirme yapıldığında çoklu yaya takibi sonuçları

	MOTA↑	MOTP↑	MT↑	ML↓	IDs↓	FM↓	FP↓	FN↓
ADL-Rundle-6	22,8	0,280	0	5	498	300	545	2822
ADL-Rundle-8	7,5	0,271	4	13	313	211	1107	4852
ETH-Bahnhof	28	0,265	**21**	137	246	258	738	4537
ETH-Pedcross2	3,4	0,277	0	135	**11**	24	105	6414
ETH-Sunnyday	16,9	0,241	1	21	21	35	57	1502
KITTI-13	5,8	0,285	0	20	79	37	175	621
KITTI-17	26,6	0,286	0	3	74	37	55	445
PETS09-S2L1	62,6	0,288	16	**0**	553	198	592	594
TUD-Campus	36,5	0,289	2	1	12	**22**	**36**	**180**
TUD-Stadtmitte	**52,5**	**0,347**	4	**0**	48	37	133	368
Venice-2	19,5	0,269	2	9	287	189	957	4505
Genel	**22,4**	**0,279**	**50**	**344**	**2142**	**1348**	**4500**	**26840**

Sonuçlar değerlendirildiğinde başarıların en iyilendiği görüntü dizilerinin MOTA ve MOTP için sırasıyla PETS09-S2L1, TUD-Stadtmitte; IDs için ETH-Pedcross2 olduğu görülmektedir.

4.3.3.Çevreleyici Kutular ile Veri İlişkilendirme

Çoklu yaya takibi çalışmasının üçüncü yöntemi bu bölümde değerlendirilmektedir. 2D MOT 15 veri seti içerisindeki görüntü dizileri için sonuçlar Çizelge 4.9'da gösterilmektedir. Yöntemin adımları sırasıyla aşağıdaki maddelerde belirtilmektedir.

1. Yolov4 ile ilk görüntüdeki yayalar tespit edilir. Aralığı 0 ile 1 arasında değişen güven değeri için eşik değeri 0,3 olarak belirlenir. Yolov4 ile tespit edilen yayaların güven değeri 0,3'ten büyük ise bu yayalar seçilir, küçük ise seçilmez.
2. Seçilen yayaların x, y, w, h şeklinde çevreleyici kutusu ile takip edilecek yayalar temsil edilir.
3. İkinci görüntüdeki yayalar yine Yolov4 ve 0,3 eşik güven değeri ile bulunur.

4. Önceden oluşturulan takip edilecek yayalar HOG tabanlı çekirdek korelasyon filtresi kullanılarak takip ettirilir ve yayaların ikinci görüntüdeki tahmini konumu hesaplanır.
5. Üçüncü adımda tespit edilen yayalar ile dördüncü adımda takip edilen yayaların ilişkilendirilmesi gerçekleştirilir. İlişkilendirme için IOU hesabı ve Macar ilişkilendirme algoritması kullanılır.
6. Bir sonraki görüntü üzerinde benzer şekilde yayalar Yolov4 ile tespit edilir ve önceki takip edilen yayaların HOG tabanlı çekirdek korelasyon filtresi takip algoritması ile yeni konumu hesaplanır. Daha sonrasında aynı ilişkilendirme adımı gerçekleştirilir.
7. Son görüntü de tamamlanana kadar altıncı adım tekrar edilir.

İlk yönteme benzer şekilde eğer takip edilen yaya art arda 10 görüntüde tespit edilemediyse ve ilişkilendirilmediyse bu yaya takip edilecek yayalar arasından çıkartılır.

Çizelge 4.9. Çevreleyici kutular ile IOU metriği kullanılarak çoklu yaya takibi sonuçları

	MOTA↑	MOTP↑	MT↑	ML↓	IDs↓	FM↓	FP↓	FN↓
ADL-Rundle-6	29,1	0,290	1	5	165	134	736	2652
ADL-Rundle-8	7,3	0,273	5	14	186	131	1416	4686
ETH-Bahnhof	27,3	0,268	**30**	132	181	132	1054	4343
ETH-Pedcross2	3	0,274	0	136	30	21	123	6403
ETH-Sunnyday	12,4	0,247	0	21	47	30	105	1514
KITTI-13	-4,8	0,292	1	21	29	16	320	625
KITTI-17	27,2	0,301	0	**0**	36	23	130	403
PETS09-S2L1	**70,1**	0,289	17	**0**	141	127	755	496
TUD-Campus	33,7	0,282	2	**0**	**18**	**15**	**72**	**148**
TUD-Stadtmitte	24,1	**0,346**	5	**0**	71	64	410	396
Venice-2	21	0,272	2	8	149	99	1190	4302
Genel	**22,7**	**0,282**	**63**	**337**	**1053**	**792**	**6311**	**25968**

Sonuçlar değerlendirildiğinde en başarılı görüntü dizilerinin MOTA ve MOTP için sırasıyla PETS09-S2L1, TUD-Stadtmitte; IDs için TUD-Campus olduğu görülmektedir.

4.3.4.Çoklu Yaya Takibi Yöntemleri Karşılaştırması

Bu bölümde çoklu yaya takibi için kullanılan yöntemlerin karşılaştırılması gerçekleştirilmektedir. Üç farklı yöntemin 2D MOT 15 veri seti üzerindeki ortalama performansı Çizelge 4.10'da gösterilmektedir. İlk satır derin özellikler ile kosinüs metriği kullanılarak gerçekleştirilen veri ilişkilendirme yönteminin sonucunu göstermektedir. İkinci satır RGB histogramları ile kosinüs metriği kullanılarak gerçekleştirilen veri ilişkilendirme yönteminin sonucunu vermektedir. Son olarak üçüncü satır ise çevreleyici kutular ile IOU metriği kullanılarak gerçekleştirilen veri ilişkilendirme yönteminin sonucunu göstermektedir.

Çizelge 4.10. Çoklu yaya takibinde kullanılan farklı veri ilişkilendirme yöntemlerinin performans karşılaştırması

	MOTA↑	MOTP↑	MT↑	ML↓	IDs↓	FM↓	FP↓	FN↓
Derin özellikler ve kosinüs metriği	**27,4**	0,278	57	**336**	**621**	**932**	4597	26109
Renk histogramı özellikleri ve kosinüs metriği	22,4	0,279	50	344	2142	1348	**4500**	26840
Çevreleyici kutular ve IOU metriği	22,7	**0,282**	**63**	337	1053	792	6311	**25968**

Çoklu yaya takibi başarısı farklı ortamlardan alınan görüntüler, ışık, görüntü bulanıklığı, kameranın hareketliliği, yayaların örtünme durumlarına göre farklılık göstermektedir. MOTA metriği incelendiğinde en başarılı sonuca sahip yöntemin derin özellikler ile ilişkilendirme yönteminin olduğu görülmektedir. Bu metrik yanlış pozitif, yanlış negatif ve kimlik değişimlerini dikkate almaktadır ve çoklu nesne takibi başarısı için büyük bir öneme sahiptir. MOTP metriği ise tahmin edilen konumlar ile eşleşen konumların ortalaması ile gerçek değerler arasında örtüşmeyi dikkate almaktadır. MOTP değeri en yüksek olan yöntemin IOU ile ilişkilendirme yöntemi olduğu görülmektedir. Bu yöntemde takip edilen nesnenin çevreleyici kutusu ile tespit edilen nesnelerin

çevreleyici kutuları arasındaki kesişiminin en yüksek olduğu nesne seçildiğinden MOTP metriği yüksek olmaktadır. MT metriği incelendiğinde en yüksek değer IOU iken ML metriği için derin özellikler yöntemidir. Üç yöntemin MOTA, MOTP ve IDs sonuçları incelendiğinde derin özelliklerin kullanılması ile başarının arttığı, aynı zamanda da takip edilen yayaların kimlik bilgilerinin değişmesinin azaldığı görülmektedir.

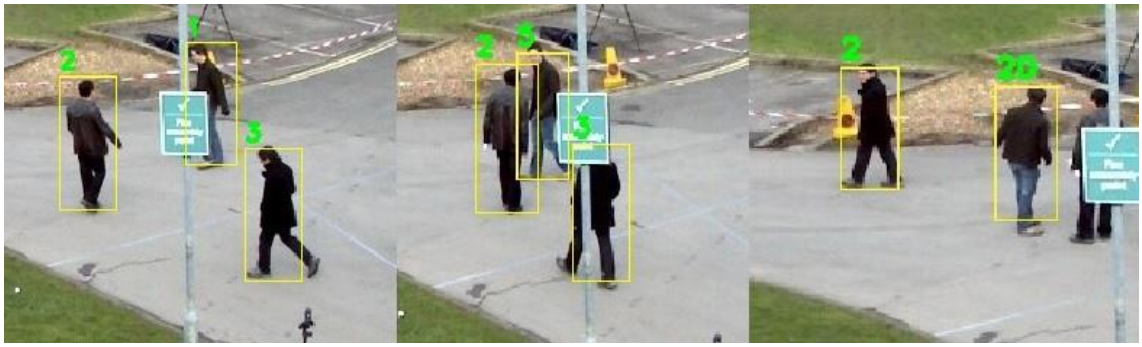
Şekil 4.11 ve 4.12'de sırasıyla 2D MOT 15 içerisinde bulunan ETH-Pedcross2 ve KITTI-13 veri setleri için çoklu yaya takibi sonuçları gösterilmektedir. Şekil 4.11' de üç yöntem için oluşan kimlik değişimi problemi gösterilmektedir. Derin özelliklerin kullanılması ile kimlik problemi azaltılmasına rağmen mevcut çalışmaların da gösterdiği şekilde tamamen önlenememektedir. Şekil 4.12'de ise kimlik değişimi problemi için IOU ve renk histogramı özellikleri yöntemleri başarısız iken derin özellikler yöntemi başarılı olmakta ve takip edilen yayalar için kimlik bilgisini korumaktadır.

DeepSORT (Wojke ve diğerleri, 2017) çalışmasında belirtildiği gibi takip edilen ve tespit edilen nesnelerin derin özellikler kullanılarak ilişkilendirilmesi ile kimlik değişimi problemi büyük ölçüde azalmıştır. Çekirdek korelasyon filtresinin HOG görünüm modeli ile kullanılması gerçek zamanlı çoklu nesne takibini mümkün kılmıştır. Böylelikle gerçek zamanlı çoklu yaya takibi sistemi sırasıyla Yolov4 ile yayaların tespit edilmesi, tespit edilen yayaların HOG ve çekirdek korelasyon filtresi ile takip edilmesi ve tespit edilen yayalar ile takip edilen yayaların derin özellikler ile ilişkilendirilmesi adımlarını ortalamada 26 fps ile gerçekleştirmiştir.

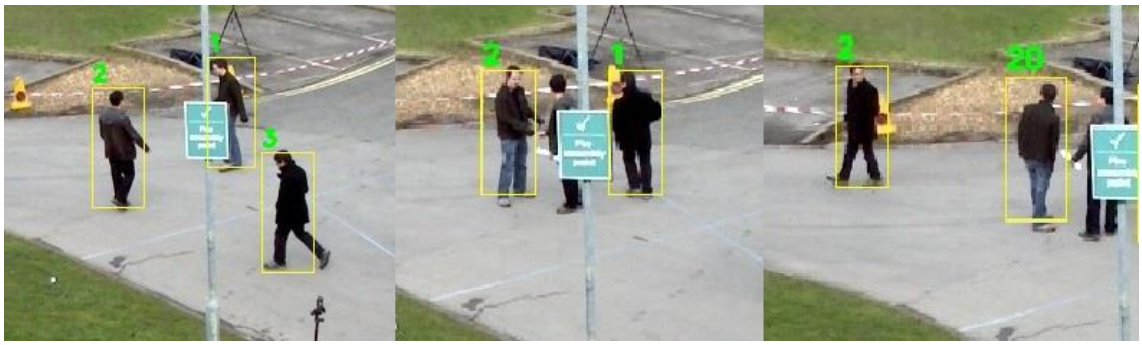
Çoklu nesne kıyaslamada bulunan ve 2D MOT 15 veri seti için sonuçları gösterilen yöntemlerden bazıları MOTICON, LP2D, CEM, RMOT, SMOT, TBD, TC_ODAL ve DP_NMS yöntemleridir. MOTA çoklu nesne izleme ölçütü ile karşılaştırma yapıldığında bu yöntemlerin en başarılısı olan MOTICON yönteminin sonucu ortalamada 23.1 olarak görülmektedir. RGB renk histogramı ve çevreleyici kutular baz alınarak ilişkilendirme gerçekleştirildiğinde başarı oranı MOTICON yönteminden daha düşük olmaktadır. Ancak ilişkilendirme için derin özellikler kullanıldığında ortalamada 27,4 olarak elde edilen MOTA değeri ile bu çalışma MOTICON yönteminden daha başarılı çalışmaktadır. Kimlik değişimi problemini temsil eden IDs ölçütü ile

karşılaştırma yapıldığında MOT'ta bulunan yöntemlerden en düşük IDs değerine sahip olan yöntem toplamda 637 ile TC_ODAL yöntemi iken en yüksek IDs değerine sahip olan yöntem toplamda 4537 ile DP_NMS yöntemidir. Mevcut çalışmada derin özellikleri temel alan veri ilişkilendirme yaklaşımı kullanıldığında ise bu değer toplamda 621'dir ve daha az sayıda kimlik değişimini ifade ettiğinden ilgili takip yöntemi daha başarılıdır. RGB renk histogramı ve çevreleyici kutuları kullanan veri ilişkilendirme yaklaşımlarıyla ise bu ölçüt toplamda sırasıyla 2142 ve 1053'tür. IDs değerlendirme ölçütüne göre derin özellikler ile veri ilişkilendirmeyi kullanan çoklu nesne takip yöntemi MOT'ta bulunan yöntemlere kıyasla daha başarılı çıkmıştır. Fakat TC_ODAL yöntemi RGB renk histogramı ve çevreleyici kutular ile veri ilişkilendirmeyi kullanan çoklu nesne takip yöntemlerinden daha başarılıdır (Leal-Taixé ve diğerleri, 2015).

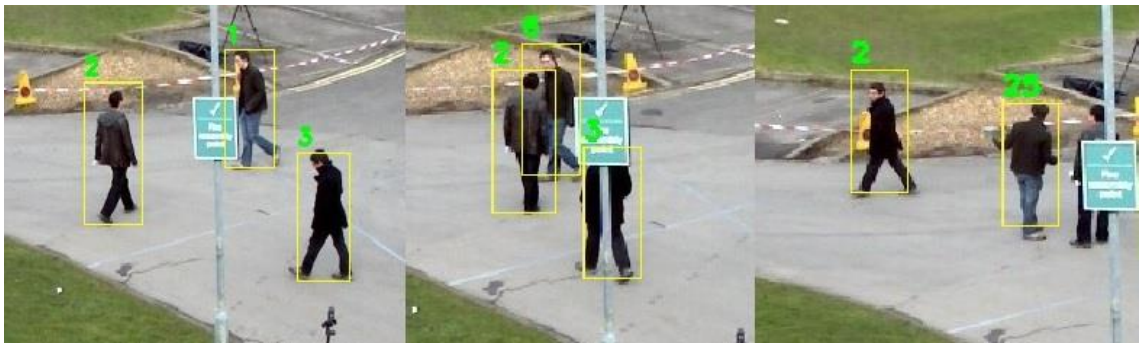
Çalışmanın gerçek zamanlı olduğunu gösteren fps değerinin 25 olduğu varsayılmaktadır (Leal-Taixé ve diğerleri, 2015). Derin özellikler ve kosinüs metriği ile gerçekleştirilen çalışma ortalama 26 fps değeri ile gerçek zamanlı çoklu yaya takibini mümkün kılar. 2D MOT 15 veri setinde bulunan Venice-2 ve ADL-Rundle-6 görüntü dizileri için ortalama 3 fps'e sahiptir. Bunun nedeni takip edilen yaya sayısının fazla olması ve görüntü boyutunun 1920x1080 olmasıdır. En yüksek fps değeri ise ETH-Pedcross2 görüntü dizisi için 95 olarak hesaplanmıştır. Bunun nedeni tespit edilen yayanın fazla olmasına rağmen ortalama 3 MOTA değeri ile başarılı takip yapılamaması ve görüntü boyutunun 640x480 olmasıdır. Renk histogramı özelliği ve kosinüs metriği ile gerçekleştirilen çalışmanın, çevreleyici kutular ve IOU metriği ile gerçekleştirilen çalışmanın da fps sonuçları sırasıyla ortalama 27 ve 28 fps'tir ve gerçek zamanlı çoklu yaya takibini gerçekleştirmektedir.



(a)



(b)



(c)

Şekil 4.11. ETH-Pedcross2 veri seti çoklu yaya takibi kimlik değişikliği problemi (a) derin özellikler (b) renk histogramı özellikleri (c) çevreleyici kutular ile veri ilişlilendirme yapıldığında takip edilen yayalar



(a)



(b)



(c)

Şekil 4.12. KITTI-13 çoklu yaya takibi kimlik değişikliği problemi (a) derin özellikler (b) renk histogramı özellikleri (c) çevreleyici kutular ile veri ilişlilendirme yapıldığında takip edilen yayalar

5. SONUÇ

Bu çalışma tekli nesne takibi ve çoklu yaya takibi çalışmalarını gerçekleştirmektedir. Tekli nesne takibi için OTB-100 veri seti kullanılmış ve takip edilecek nesnenin ilk konum bilgisi taban gerçeklik konum ile belirlenip takip işlemi gerçekleştirilmiştir. Takip algoritması olarak temelde MOSSE, Kalman, çekirdek korelasyon filtresi ile hiyerarşik çekirdek korelasyon filtresi kullanılmış ve tüm bunların hem kendi aralarında hem de OpenCV kütüphanesinde mevcut olan takip algoritmaları ile toplu bir karşılaştırması yapılmıştır. Kullanılan OTB-100 veri seti için en başarılı sonuç ortalamada 0,771 kesinlik ve 0,537 başarı oranı ile hiyerarşik çekirdek korelasyon filtresi ile elde edilmiştir. Aynı zamanda farklı görünüm modellerinin takip başarısına etkisini analiz etmek amacıyla 8 farklı görünüm modeli çekirdek korelasyon filtresi ile uygulanmıştır. Bu görünüm modelleri görüntünün gri düzey yoğunlukları, görüntünün RGB renk bilgisi, HOG özellikleri ve VGG-19 evrişimli sinir ağının block1_conv2, block2_conv2, block3_conv4, block4_conv4, block5_conv4 evrişim katmanı çıktılarıdır. OTB-100 veri seti için gri düzey yoğunlukları, RGB renk bilgisi ve HOG özelliklerinden en başarılı sonucu ortalamada 0,679 kesinlik ve 0,473 başarı oranı ile HOG özniteliğinin kullanımı sağlamıştır. VGG-19 ağının evrişim katman çıktılarından en başarılı sonuç 4. bloğun evrişim çıktısı olan block4_conv4 ile elde edilmiştir. Tüm görünüm modelleri arasında HCF yönteminden sonraki en başarılı sonuç ise yine VGG-19 ağının block4_conv4 katman çıktısı ile elde edilmiştir. Genel sonuçlar incelendiğinde Henriques ve diğerlerinin (2014) çalışmalarında belirttiği gibi evrişimli sinir ağlarının katmanlarından gelen derin özelliklerin takip edilecek nesnenin görünüm modeli olarak belirlenmesi ile takip başarısının arttığı gözlemlenmiştir.

Nesne takibi çalışmalarında sıklıkla karşılaşılan problemler aydınlatma değişiklikleri, hızlı hareket, poz değişimleri, kısmi örtünmeler ve arka plan karmaşıklığı gibi durumlardır. Bu gibi durumlarda nesne takibi zorlaşmaktadır. Mevcut çalışmalar incelendiğinde tüm zorluk durumları için başarılı bir algoritma geliştirmenin güç olduğu ve bu nedenle çalışmaların belirli bir alt problemin çözümüne odaklandığı görülmektedir. Çoklu nesne takibinde tüm bu zorlu durumlar göz önünde tutularak görüntüdeki bütün nesnelerin doğru kimlik bilgileri ile takibi denenmiştir.

Çoklu yaya takibi için 2D MOT 15 veri seti kullanılmıştır. Yolov4 ile tespit edilen yayalar HOG özniteliği ve çekirdek korelasyon filtresi ile takip edilmiştir. Bunun sebebi hiyerarşik çekirdek korelasyon filtresinin başarısının daha yüksek olmasına rağmen fps değerlerinin düşük olmasıdır. Dolayısıyla gerçek zamanlı çoklu yaya takibi için başarı, kesinlik ve fps değerlerinde bir denge sağlayan HOG özellikleri tabanlı çekirdek korelasyon filtresinin kullanımı daha uygun görülmüştür. Takip edilen yayalar ile tespit edilen yayaların ilişkilendirilerek kimlik bilgilerinin doğru atanması için 3 farklı yöntem kullanılmıştır. Bunlar yayaların derin özelliklerinin benzerliklerine göre ilişkilendirme, RGB renk histogramı özelliklerinin benzerliklerine göre ilişkilendirme ve yayaların çevreleyici kutuları arasında IOU hesaplaması yaparak örtüşme oranına göre ilişkilendirme yöntemleridir. Sonuçlar değerlendirildiğinde veri ilişkilendirmede derin özelliklerin kullanılmasının başarıları görece arttırdığı anlaşılmıştır. Aynı zamanda bu yöntem ile yayaların ilişkilendirilmesi sırasında kimlik bilgisi değiştirme (IDs) miktarının da azaldığı gözlenmiştir.

Derin öğrenme yöntemleri nesne takibi ve nesne ilişkilendirme başarısını arttırmaktadır. Ancak takip edilen nesnelerin örtünmesi ya da uzun süreli kaybolması, arka plan karmaşıklığı gibi zorlu durumlarda başarı oranı azalmaktadır. Derin öğrenme ağında katman sayısı arttıkça nesneler için elde edilecek özellikler daha anlamlı hale gelmektedir. Ancak katman sayısının artması daha fazla işlem kapasitesine sahip bilgisayarlara ihtiyacı arttırmaktadır. GPU'lu bilgisayarların işlem kapasitelerinin artması ile daha güçlü derin öğrenme ağları oluşturularak nesne takibi başarısı arttırılabilir ve kimlik değişimi problemi azaltılabilir.

KAYNAKLAR

Ayşe, A. R. I. ve Berberler, M. E. (2017). Yapay sinir ağları ile tahmin ve sınıflandırma problemlerinin çözümü için arayüz tasarımı. *Acta Infologica*, *1*(2), 55-73.

Bewley, A., Ge, Z., Ott, L., Ramos, F. ve Upcroft, B. (2016, Eylül). Simple online and realtime tracking. *IEEE International Conference on Image Processing (ICIP)*, 3464-3468. doi: 10.1109/ICIP.2016.7533003

Bochkovskiy, A., Wang, C. Y. ve Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*

Bolme, D. S., Beveridge, J. R., Draper, B. A. ve Lui, Y. M. (2010, Haziran). Visual object tracking using adaptive correlation filters. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2544–2550. doi: 10.1109/CVPR.2010.5539960

Brdjanin, A., Dardagan, N., Dzigal, D. ve Akagic, A. (2020, Ağustos). Single object trackers in OpenCV: A benchmark. *International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, 1-6. doi: 10.1109/INISTA49547.2020.9194647

Collins, B. (2012). Introduction to Data Association. Erişim adresi: http://www.cse.psu.edu/~rtc12/CSE598C/datassocPart1.pdf

Comaniciu, D., Ramesh, V. ve Meer, P. (2000, Haziran). Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2*, 142-149. doi: 10.1109/CVPR.2000.854761

Dehghan, A., Modiri, A. S. ve Shah, M. (2015). GMMCP tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 4091-4099. doi: 10.1109/CVPR.2015.7299036

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., ... Leal-Taixé, L. (2020). MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K. ve Fei-Fei, L. (2009, Haziran). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. doi: 10.1109/CVPR.2009.5206848

Dilmen, H. ve Talu, M. F. (2017). Yapısal özellikleri kullanan parçacık filtresi ile uzun süreli nesne takibi. *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji, 5*(1), 107-118.

Doğan, F. ve Türkoğlu, İ. (2019). Derin öğrenme modelleri ve uygulama alanlarına ilişkin bir derleme. *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi, 10*(2), 409–445. doi: 10.24012/dumf.411130

Dollár, P., Appel, R., Belongie, S. ve Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(8), 1532-1545. doi: 10.1109/TPAMI.2014.2300479

Emami, P., Pardalos, P. M., Elefteriadou, L. ve Ranka, S. (2020). Machine learning methods for data association in multi-object tracking. *ACM Computing Surveys, 53*(4), 1-34. doi: 10.1145/3394659

Fan, L., Wang, Z., Cail, B., Tao, C., Zhang, Z., Wang, Y., …, Zhang, F. (2016, Ağustos). A survey on multiple object tracking algorithm. *IEEE International Conference on Information and Automation (ICIA),* 1855–1862. doi: 10.1109/ICInfA.2016.7832121

Hanbay, K. ve Üzen, H. (2017). Nesne tespit ve takip metotları: Kapsamlı bir derleme. *Türk Doğa ve Fen Dergisi, 6*(2), 40-49.

Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M. M., Hicks, S. L. ve Torr, P. H. (2015). STRUCK: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(10), 2096-2109. doi: 10.1109/ICCV.2011.6126251

Henriques, J. F., Caseiro, R., Martins, P. ve Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37*(3), 583-596. doi: 10.1109/TPAMI.2014.2345390

History of Artificial Neural Networks. (2021, 4 Ocak). Erişim adresi: https://en.wikipedia.org/wiki/History_of_artificial_neural_networks

Ivakhnenko, A. G. ve Lapa, V. G. (1966). *Cybernetic predicting devices*. Purdue University School Of Electrical Engineering, Indiana.

İleri, A. (2018, 26 Ağustos). 2D Convolution [Blog yazısı]. Erişim adresi: https://medium.com/@abdulsamet.ileri/2d-convolution-ced5d339aa5

Joyce, J. (2003, 30 Eylül). Bayes' Theorem. Erişim adresi: https://stanford.library.sydney.edu.au/

Kalal, Z., Mikolajczyk, K. ve Matas, J. (2011). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis ve Machine Intelligence, 34*(7), 1409–1422. doi: 10.1109/TPAMI.2011.239

Kalman Filtresi. (2021, 4 Ocak). Erişim adresi: https://tr.wikipedia.org/wiki/Kalman_Filtresi

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering Transactions, 82*, 35-45.

Laaraiedh, M. (2012). Implementation of Kalman filter with python language. *arXiv preprint arXiv:1204.0375*.

Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., ... Hubbard, W. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, *27*(11), 41-46. doi: 10.1109/35.41400

Leal-Taixé, L., Milan, A., Reid, I., Roth, S. ve Schindler, K. (2015). MOTchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

Li, X., Wang, K., Wang W. ve Li, Y. (2010, Haziran). A multiple object tracking method using Kalman filter. *IEEE International Conference on Information and Automation*, 1862-1866. doi: 10.1109/ICINFA.2010.5512258

Liu, S., Qi, L., Qin, H., Shi, J. ve Jia, J. (2018). Path aggregation network for instance segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 8759-8768. doi: 10.1109/CVPR.2018.00913

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W. ve Kim, T. K. (2020). Multiple object tracking: A literature review. *Artificial Intelligence*, 103448. doi: 10.1016/j.artint.2020.103448

Ma, C., Huang, C. J. B., Yang, X. ve Yang, M. H. (2015). Hierarchical convolutional features for visual tracking. *IEEE International Conference on Computer Vision*, 3074-3082. doi: 10.1109/ICCV.2015.352

Makinist, S. (2018, 16 Nisan). Derin Öğrenme (Yapay Sinir Ağları-3) [Blog yazısı]. Erişim adresi: http://buyukveri.firat.edu.tr/2018/04/16/derin-ogrenme-yapay-sinir-aglari-3/

Milan, A., Leal-Taixé, L., Reid, I., Roth, S. ve Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Ozturk, Ş. ve Şahin, M. E. (2018). Yapay sinir ağları ve yapay zekâ'ya genel bir bakış. *Takvim-i Vekayi*, *6*(2): 25-36.

Peker, M., Altun, H. ve Karakaya, F. (2012). HOG temelli bir yöntem ile ölçek ve yönden bağımsız gerçek zamanlı nesne tanıma. *Otomatik Kontrol Türk Milli Komitesi'nin Otomatik Kontrol Ulusal Toplantısı*.

Poore, A. B. ve Gadaleta, S. (2006). Some assignment problems arising from multiple target tracking. *Mathematical and Computer Modelling*, *43*(9-10), 1074-1091. doi: 10.1016/j.mcm.2005.05.026

Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J. ve Yang, M. H. (2016). Hedged deep tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 4303-4311. doi: 10.1109/CVPR.2016.466

Redmon, J. ve Farhadi, A. (2016). YOLO9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition*, 7263-7271. doi: 10.1109/CVPR.2017.690

Redmon, J. ve Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, J., Divvala, S., Girshick, R. ve Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 779-788. doi: 10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R. ve Sun, J. (2017, 1 Haziran). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(6), 1137-1149. doi: 10.1109/TPAMI.2016.2577031

Rifkin, R., Yeo, G. ve Poggio, T. (2003). Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, *190*, 131–154.

Saha, S. (2018, 15 Kasım). A Comprehensive Guide to Convolutional Neural Networks [Blog yazısı]. Erişim adresi: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Saygili, A. ve Albayrak, S. (2018, Mayıs). Meniscus tear classification using histogram of oriented gradients in knee MR images. *Signal Processing and Communications Applications Conference (SIU)*, 1-4. doi: 10.1109/SIU.2018.8404375

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85–117. doi: 10.1016/j.neunet.2014.09.003

Shaikh, S. H., Saeed, K. ve Chaki, N. (2014). Moving object detection approaches, challenges and object tracking. *Moving Object Detection Using Background Subtraction*, 5-14. Springer, Cham.

Simonyan, K. ve Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Stuart, A. ve Ord, J. K. (1994). *Kendall's advanced theory of statistics: Distribution theory* (Basım 1). New York City, New York: John Wiley and Sons Inc. (s. 266-267).

Şeker, A., Diri, B. ve Balık H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. *Gazi Mühendislik Bilimleri Dergisi, 3*(3), 47-64.

The Hungarian algorithm. (2021, 4 Ocak). Erişim adresi: http://www.hungarianalgorithm.com/hungarianalgorithm.php

The Hungarian algorithm: An example. (2021, 4 Ocak). Erişim adresi: http://www.hungarianalgorithm.com/ examplehungarianalgorithm.php

Training Neural Networks. (2021, 4 Ocak). Erişim adresi: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf

Vatavu, A., Danescu, R. ve Nedevschi, S. (2014). Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters. *IEEE*

Transactions on Intelligent Transportation Systems, *16*(1), 498-511. doi: 10.1109/TITS.2014.2366248

Visual Tracker Benchmark. (2021, 4 Ocak). Erişim adresi: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html

Walia, G. S. ve Rajiv Kapoor, R. (2016). Recent advances on multicue object tracking: A survey. *Artificial Intelligence Review*, *46*(1), 1-39. doi: 10.1007/s10462-015-9454-6

Wang, C. Y., Liao, H. Y. M., Wu, Y. H., Chen, P. Y., Hsieh, J. W. ve Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 390-391. doi: 10.1109/CVPRW50498.2020.00203

Welch, G. ve Bishop, G. (1995). *An introduction to the Kalman filter*. University of North Carolina Department of Computer Science, North Carolina.

Wojke, N., Bewley, A. ve Paulus, D. (2017, Eylül). Simple online and realtime tracking with a deep association metric. *IEEE International Conference on Image Processing (ICIP)*, 3645–3649. doi: 10.1109/ICIP.2017.8296962

Woo, S., Park, J., Lee, J. Y. ve Kweon, I. S. (2018). CBAM: Convolutional block attention module. *European Conference on Computer Vision (ECCV*), 3-19.

Wu, Y., Lim, J. ve Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1834-1848. doi: 10.1109/TPAMI.2014.2388226

Yang, H., Shao, L., Zheng, F., Wang, L. ve Song, Z. (2011). Recent advances and trends in visual tracking: A review. *Neurocomputing*, *74*(18), 3823-3831. doi: 10.1016/j.neucom.2011.07.024

Yilmaz, A., Javed, O. ve Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, *38*(4), 13. doi: 10.1145/1177352.1177355

Zamir, A. R., Dehghan, A. ve Shah, M. (2012). GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs. *European Conference on Computer Vision*, 343-356. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-33709-3_25

Zeiler, M. ve Fergus, R. (2014, Eylül). Visualizing and understanding convolutional networks. *European Conference on Computer Vision*, 818-833. Springer, Cham. doi: 10.1007/978-3-319-10590-1_53

Zhang, L., Li, Y. ve Nevatia, R. (2008, Haziran). Global data association for multi-object tracking using network flows. *IEEE Conference on Computer Vision and Pattern Recognition*, 1-8. doi: 10.1109/CVPR.2008.4587584

Zhang, T., C. Xu, C. ve Yang, M. H. (2018). Learning multi-task correlation particle filters for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(2), 365-378. doi: 10.1109/TPAMI.2018.2797062

Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C., Wang, S. ve Tian, Q. (2016, Ekim). MARS: A video benchmark for large-scale person re-identification. *European Conference on Computer Vision*, 868-884. Springer, Cham. doi: 10.1007/978-3-319-46466-4_52

Zou, Z., Shi, Z., Guo, Y. ve Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.