GPU İLE HIZLANDIRILMIŞ GERÇEK / YARI GERÇEK ZAMANLI NESNE TAKİBİ

Zafer GÜLER

Doktora Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Ahmet ÇINAR

EYLÜL-2019

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GPU İLE HIZLANDIRILMIŞ GERÇEK/YARI GERÇEK ZAMANLI NESNE TAKİBİ

DOKTORA TEZİ

Yük. Müh. Zafer GÜLER

(131129204)

Tezin Enstitüye Verildiği Tarih : 05 Eylül 2019
Tezin Savunulduğu Tarih : 27 Eylül 2019

Tez Danışmanı : **Dr. Öğr. Üyesi Ahmet ÇINAR (F.Ü.)**

Diğer Jüri Üyeleri : **Prof. Dr. Ali KARCI (İ.Ü.)**

Prof. Dr. Abdulkadir ŞENGÜR (F.Ü.)

Doç. Dr. Galip AYDIN (F.Ü.)

Doç. Dr. Muhammed Fatih TALU (İ.Ü.)

Eylül-2019

ÖNSÖZ

Bu çalışma, Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak hazırlanan "GPU ile Hızlandırılmış Gerçek/Yarı Gerçek Zamanlı Nesne Takibi" isimli tezi içermektedir. Tez çalışmasında, nesne takibi sistemlerinde yaygın olarak kullanılan nokta tabanlı özellik çıkarım algoritmaları için bir altyapı sunulmuştur. Bu sistem sayesinde nesne takip sistemlerinde önemli derecede iyileştirilme sağlanmıştır.

Yüksek lisans ve doktora öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Dr. Öğr. Üyesi Ahmet Çınar'a en içten dileklerimle teşekkür ederim. Ayrıca öğrenim hayatım boyunca destekleri ile bu noktaya ulaşmamda büyük katkıları olan değerli aileme, bu süreç içerisinde her konuda sabırla bana destek veren eşim Derya Güler'e ve oğlum Ünal Yekta Güler'e sonsuz teşekkür ederim.

Tez çalışması Fırat Üniversitesi Bilimsel Araştırmalar Birimi (Proje No: MF.16.61) tarafından desteklenmektedir.

Zafer Güler

Elazığ, 2019

İÇİNDEKİLER

Sayfa No

ÖZET

Özellik tespiti için kullanılan en yaygın yöntemler, nokta tabanlı özellik tespit yöntemleridir ve SIFT yöntemi bunlardan en tanınmış olanıdır. Tez çalışmasında SIFT ve SURF yöntemleri kullanılmıştır. Ayrıca özellik çıkarımını hızlandırmak için GPU tabanlı SURF yöntemi de kullanılmıştır.

Çalışmada ilk olarak, klasik SIFT, SURF ve GPU-SURF algoritmaları imge eşleştirme ve nesne izleme uygulamalarında test edilmiştir. Yapılan testlerde bu yöntemlerin ölçeklendirme, döndürme, ışık değişim, bulanıklık ve afin dönüşümü gibi imge değişimlerine karşı iyi sonuçlar verdiği ve nesneyi doğru bir şekilde tespit edebildiği görülmüştür. Fakat bu yöntemlerin başarısına rağmen, nesne izleme uygulamalarında başarı oranının düşük olması bir problem olarak karşımıza çıkmaktadır.

Bu probleme çözüm olarak tez çalışmasında, SIFT, SURF ve GPU-SURF yöntemleri gibi nokta tabanlı özellik çıkarım algoritmaları için yeni bir nesne izleme altyapısı geliştirilmiştir. Önerilen yöntemde özellik çıkarım algoritması herhangi bir değişiklik yapılmadan kullanılmaktadır. Elde edilen özellikler kullanılarak üç adımda nesne izleme işlemini gerçekleştirmektedir. Bu adımlar; hatalı özellik tespiti, nesne modelleme ve nesne izleme adımlarıdır.

Hatalı özellik tespiti adımında, özellik çıkarım algoritması kullanılarak ayırt edici noktalar tespit edilir ve daha sonra DBScan algoritması kullanılarak hatalı özellik eşleştirmeleri tespit edilmektedir. İkinci adımda nesne modeli kutu şeklinde tanımlanmıştır. Kutu modeli altı değerden oluşmaktadır ve her bir değer için Gauss modeli kullanılmıştır. DBScan ve Gauss yöntemlerinin birleşmesi ile nesne özellik noktalarının az ve hatalı olduğu durumlarda bile nesne pozisyonu doğru bir şekilde tahmin edilebilmektedir. Son adımda, Gauss yumuşatma işlemi uygulanarak nesne izleme işlemi gerçekleştirilmiştir.

Geliştirilen nokta tabanlı nesne izleme uygulaması, SIFT ve varyantları ile uyumlu bir şekilde çalışmaktadır. Ayrıca diğer nokta tabanlı özellik çıkarım algoritmaları kısa bir entegrasyon işlemi sonucu nesne izleme altyapısına dahil edilebilir. Bilgisayar ortamında yapılan deney sonuçlarına göre önerilen izleme uygulaması çalışma zamanını etkilemeksizin nesne izleme başarısını önemli derecede iyileştirmiştir.

Anahtar Kelimeler: Özellik çıkarımı, Nesne izleme, SIFT, SURF, GPU-SURF

ABSTRACT

GPU Accelerated Real / Semi-Real Time Object Tracking

One of the most common methods used for feature detection is the point based feature extraction methods and the SIFT method is the most well-known. In this thesis, SIFT and SURF methods are used. Also the GPU-based SURF method are used to accelerate feature extraction.

In this study firstly, SIFT, SURF and GPU based SURF algorithms are used for image matching and object tracking. Experiments show that, these methods give good results against image changes such as scaling, rotation, illumination, blur and affine transformation and can detect the object accurately. However, despite the stated success of these methods, the low success rate in object tracking applications is a problem.

Against this problem, we present a novel object tracking framework for interest point based feature extracting algorithm such as SIFT, SURF and GPU-SURF methods. The proposed framework uses the feature extraction algorithm without making any changes. In the proposed system, it performs object tracking in three steps using the obtained features. These steps are outlier detection, object modeling, and object tracking.

In the outlier detection step, after the keypoints are extracted by using a feature extraction algorithm, incorrect keypoint matches are detected by using the DBScan algorithm. In the second step, the object model is defined as a bounding box. The box model has six values and each of these values have its own Gaussian model. With the combination of DBScan and Gauss methods, object position can be accurately predicted even when the object feature points are few and inaccurate. Finally, the Gaussian smoothing process is performed for object tracking.

The developed point based object tracking application works in harmony with SIFT and its variants. In addition, other point based feature extraction algorithms can be added to the object tracking framework with a short integration process. According to the results of experiments carried out in computer environment show that the proposed tracker improves the success rate of the object tracking significantly without affecting the execution time.

Key Words: Feature Extraction, object tracking, SIFT, SURF, GPU-SURF

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

AP : Ortalama Kesinlik
ASIFT : Afin SIFT
BG : Benzer Görünüm
BoBoT : Bonn Benchmark on Tracking
CUDA : Compute Unified Device Architecture
DBScan : Density Based Spatial Clustering of Applications with Noise
DLT : Direct Linear Transformation
DVM : Destek Vektör Makinesi
EHTZ : ETH Zurich
EMD : The Earth Mover's Distance
ESA : Evrişimsel Sinir Ağı
FS : F Skor
GD : Görünüm Değişikliği
GF : Gauss Farkı
GN : Gerçek Negatiftir
GP : Gerçek Pozitif
GPU : Grafik İşlemci Birimi
HA : Hassasiyet
HOG : Gradyanların Histogramı
HSV : Hue, Saturation, Value
JPDAF : Joint Probability Data Association Filtering
KE : Kesinlik
KLT : Kanade-Lucas-Tomasi
KN : Kamera veya Nesne Hareketi
LoU : Kesişim
mAP : Ortalama Yaklaşık Doğruluk
MHT : Multiple Hypothesis Tracking
NÇ : Nesne Çözünürlüğü
NTNİ : Nokta Tabanlı Nesne İzleme Uygulaması
NTÖÇ : Nokta Tabanlı Özellik Çıkartma

OpenCV : Open Source Computer Vision

PCA-SIFT : Temel Bileşen Analizi SIFT

RANSAC : Random Sample Consensus

RGB : Kırmızı Yeşil Mavi

SIFT : Scale Invariant Feature Transform

SURF : Speeded Up Robust Features

TBA : Temel Bileşen Analizi

TBN : Test Başarısızlık Nedeni

TPR : Doğru Pozitif Oran

ViBe : Visual Background Extractor

YİD : Yerel İkili Doku

YN : Yanlış Negatif

YP : Yanlış Pozitif

SEMBOLLER LİSTESİ

$\boldsymbol{B}$: İmge üzerinde bir bölge

$\boldsymbol{C}$: Kontur

$\boldsymbol{D}$: Köşegen matrisi

$\boldsymbol{e(\lambda, \vec{x})}$: Işıklandırma spektrumunu

$\boldsymbol{E_{ext}}$: Sınırlama kuvveti

$\boldsymbol{E_{int}}$: Kontur eğirisinin iç enerjisini,

$\boldsymbol{E_{im}}$: İmgenin görünüşünden elde edilen enerji

$\boldsymbol{E(\lambda, \vec{x})}$: Gözlem pozisyonundan yansıyan spektrumu

$\boldsymbol{g}$: Gauss modeli

$\boldsymbol{G}$: Graf

$\boldsymbol{GS}$: 60 boyutlu global vektörü

$\boldsymbol{h}$: Kutu gösterimi için kullanılan yükseklik bilgisi

$\boldsymbol{I(x, y)}$: Piksel değeri

$\boldsymbol{I(x, y, i)}$: Piksel özellikleri ve piksel değerleri

$\boldsymbol{I_x}$: x yönündeki imge türevi

$\boldsymbol{I_y}$: y yönündeki imge türevi

$\boldsymbol{K}$: Bölge koveryans tanımı

$\boldsymbol{N}$: Beyaz Gürültü

$\boldsymbol{M}$: Moment Matrisi

$\boldsymbol{p_f(\vec{x})}$: $\vec{x}$ noktasının Fresnel yansıması katsayısını,

$\boldsymbol{p(X_t|Z_t)}$: t zamanındaki şartlı durum yoğunluğu

$\boldsymbol{R_\infty(\lambda, \vec{x})}$: Materyal yansımasını

$\boldsymbol{s}$: C konturunun uzunluğu

$\boldsymbol{SS}$: 128 boyutlu SIFT vektörü

$\boldsymbol{s_t^{(n)}}$: Parçacık filtresinde örnek kümesi

$\boldsymbol{t}$: Zaman katsayısı

$\boldsymbol{T}$: Eşik değeri

$\boldsymbol{V}$: Graf üzerindeki düğümler

$\boldsymbol{w}$: Kutu gösterimi için kullanılan genişlik bilgisi

$\boldsymbol{w_{i,t}}$: t zamanındaki dağılımın ağırlık değeri

$\boldsymbol{W}$: Ağırlık katsayısı

$\boldsymbol{x}$: Kutu gösterimi için kullanılan x koordinat bilgisi

$\boldsymbol{x_{kp}}$: Eşleşen özellik noktalarının ortalama x değerleri

$\vec{\boldsymbol{x}}$: İmge pozisyonunu temsil eden 2 boyutlu vektör

$\boldsymbol{X_t}$: t zamanındaki nesne model değeri

$\boldsymbol{y}$: Kutu gösterimi için kullanılan y koordinat bilgisi

$\boldsymbol{y_{kp}}$: Eşleşen özellik noktalarının ortalama y değerleri

$\boldsymbol{\pi_t^{(n)}}$: Parçacık filtresinde ağırlıklar

$*$: Konvolüsyon işlem

$\boldsymbol{\omega}$: Göreceli ağırlık faktörü

$\boldsymbol{\lambda}$: Dalga boyu

$\boldsymbol{\mu_i}$: Dağılımın ortalaması

$\boldsymbol{\sum_i}$: Dağılımın standart sapması

$\boldsymbol{\sigma}$: Standart sapma değeri

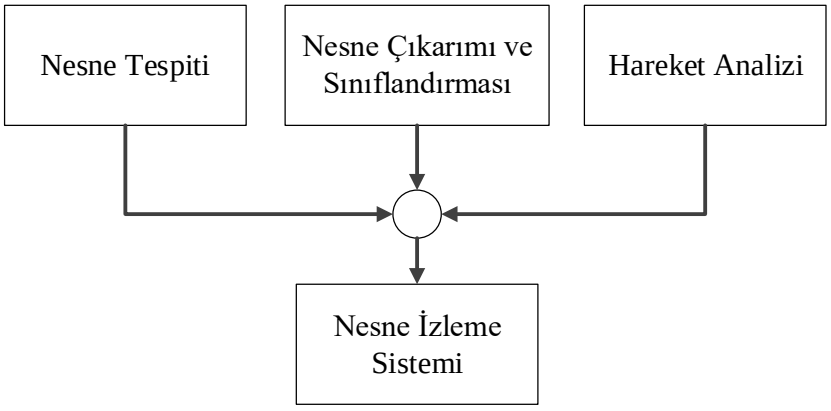
$\emptyset_c$: c çerçevesindeki kesişim değeri

1. GİRİŞ

Nesne izleme, ihtiyaç duyulan duruma göre hareketli veya sabit kamera sistemlerini kullanarak video çerçeveleri içerisinde hareket eden bir veya daha fazla sayıdaki hareketli ya da sabit nesne/nesnelerin yerlerinin belirlenerek izlenmesi için kullanılan teknikler bütünü olarak tanımlanabilir. Mevcut sistemlerde nesnelerin izlenebilirliği birbirini takip eden video çerçevelerinde çeşitli analiz metotları kullanılarak çerçeveler arasındaki farklılıklar / farkları tespit eden algoritmalar yardımı ile sağlanmaktadır. Nesne izleme sistemleri; nesne tanıma, yapay zekâ ve hareket algılama gibi birden fazla farklı yöntemlerin birbirleri ile uyumlu hale gelmesinden oluşmaktadır. Bu yöntemlerin etkili ve uyumlu kullanımı arttıkça sistemlerin doğruluk oranı artmakta ve bununla beraber sistemlerin güvenilirlik düzeyi de yükselmektedir. Otomatik nesne sınıflandırma ve izleme günümüzde birçok nesne izleme sisteminin en önemli bileşenlerinden birisi olarak kabul edilmektedir [1].

Şekil 1.1'de göründüğü üzere nesne izleme sistemleri altyapısı genel olarak;

- Hareket eden nesneleri tespit etme
- Nesne sınıflandırma ve nesne çıkarımı (İzlenmek istenen nesneyi ayırt etme ve sınıflandırma)
- İzlenen nesne hareketlerini analiz etme bileşenlerinden oluşmaktadır [1].



Şekil 1.1. Nesne takip sistemi

Nesne izleme sistemlerinde tek bir genel çözüm yoktur. Nesne izleme sisteminin her bir aşamasında birçok teknik kullanılmaktadır. Bu tekniklerin her birisi kendi güçlü yönleri ve

sınırları vardır. Nesne izleme sisteminde yaygın olarak kullanılan teknikler ve kullanım amaçları Tablo 1.1'de verilmiştir.

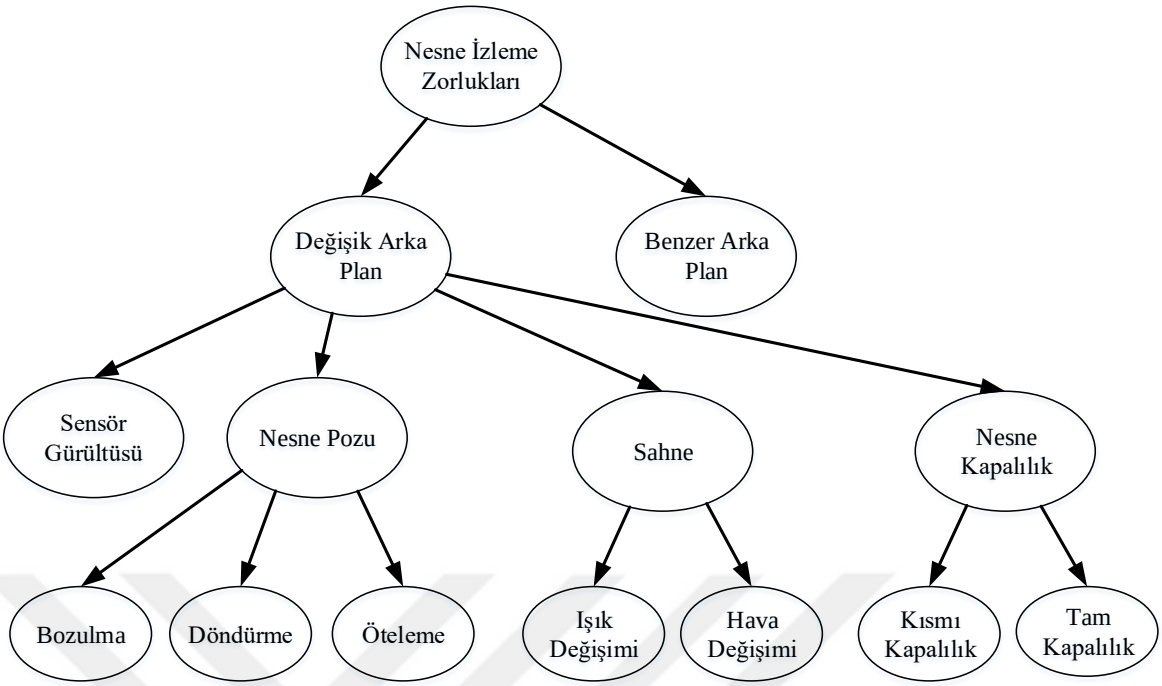
Tablo 1.1. Nesne izleme sistemlerinde yaygın olarak kullanılan yöntemler ve kullanım amaçları

Yöntem	Kullanım Amacı
Arkaplan Çıkarımı	Nesne Tespiti
Temel Bileşen Analizi	Nesne Sınıflandırma
Kalman Filtresi	Nesne Sınıflandırma
Ortalama Kayma Algoritması	Nesne Sınıflandırma
CamShift Yöntemi	Nesne Sınıflandırma
SIFT, ASIFT ve SURF Yöntemi	Nesne Özellik Çıkarımı ve Nesne Takibi

Nesne izleme uygulamalarında, nesneleri doğru bir şekilde tespit etmek zorlu bir görevdir. Özellikle nesnelerin karmaşık yapılara sahip olması, sonraki video karelerinde şekil, boyut, konum değişiklikleri nesne tespitini zorlaştırmaktadır. Son yıllarda birçok algoritma belirli bir video üzerinde yoğunlaşan birçok algoritmalar geliştirilmiştir. Geliştirilen nesne izleme uygulamaların tümünde izlenen nesnede sapmalar meydana gelmektedir. Geliştirilen nesne izleme uygulamasının başarısı ise nesne sapmalarının minimize edilmesi ile sağlanabilmektedir.

Nesne izleme uygulamalarında en önemli zorluk, takip edilmek istenen nesne ile benzer görünüşe sahip arka planın bulunduğu durumlardır. Benzer arka plan dışında nesne izleme uygulamalarında sıkça karşılaşılan başka zorluklar da bulunmaktadır. Nesne izlemede karşılaşılan zorluklar Şekil 1.2'de verilmiştir.

Son yıllarda kamera teknolojisinin gelişmesi ile birlikte birçok alanda yüksek çözünürlüklü kameraların yaygın olarak kullanıldığını görmekteyiz. Bunun sonucu olarak yüksek çözünürlüklü videolar üzerinde nesne tespiti ve nesne izleme uygulamaları zorunluluk haline gelmektedir. Bununla başa çıkabilmek için yüksek işlemci gücüne ihtiyaç duyulmaktadır. Yapılan çalışmalarda, çok işlemcili yapılar ile yüksek hesaplama güç ihtiyacını karşılamak amaçlı kullanılmaktadır [2].



Şekil 1.2. Nesne izleme zorlukları [2].

Son on yıldır çok işlemcili yapılar ortaya çıkması ve günümüzde yaygın şekilde kullanılmaya başlanması nesne takip uygulamalarında bu tür sistemlerin kullanılmasının zorunlu kılmıştır. Bu yapılar hem endüstride hem de akademik araştırmalarda kullanılmış ve problemlere ve zorluklara çözümler getirmiştir. Günümüzde bu yapılar üzerinde birçok problem, algoritma kolay bir şekilde programlanabilmektedir [3, 4]. Çok işlemcili yapılar olarak sadece ana işlemci birimini (CPU) düşünmemek gerekir. Aksine Grafik İşlemci Biriminin (GPU) asıl özelliği işlemlerin paralel çalışmasına izin vermesidir. Fakat GPU üzerinde uygulama geliştirmek bazı özel kütüphanelerin bilinmesini gerektirmektedir. Son yıllarda yapılan çalışmalar ile grafik işlemciler bile kolay bir şekilde genel amaçlı uygulamaları çalıştırılır hale gelinmiştir [5].

Grafik işlemcilerinin genel amaçlı uygulamalar için kullanılması özellikle 2007 yılında NVIDIA firması tarafından geliştirilen C programlama dili tabanlı CUDA (Compute Unified Device Architecture) mimarisi ile birlikte hızla yaygınlaşmıştır. CUDA mimarisi yazılımcılara grafik işlemci bilgisi olmaksızın genel amaçlı programlar yapmasını sağlamıştır. GPU tabanlı uygulamalar sadece bilimsel alanda değil, görüntü ve video işleme, akışkanlar dinamiği simülasyonu gibi diğer yüksek performans gerektiren alanlarda da kullanılmaktadır [6].

1.1. Tezin Amacı

Tez çalışmanın amacı, gerçek/yarı gerçek zamanlı video görüntüleri üzerinde nesne tespiti ve istenilen nesne veya nesnelerin izlenmesidir.

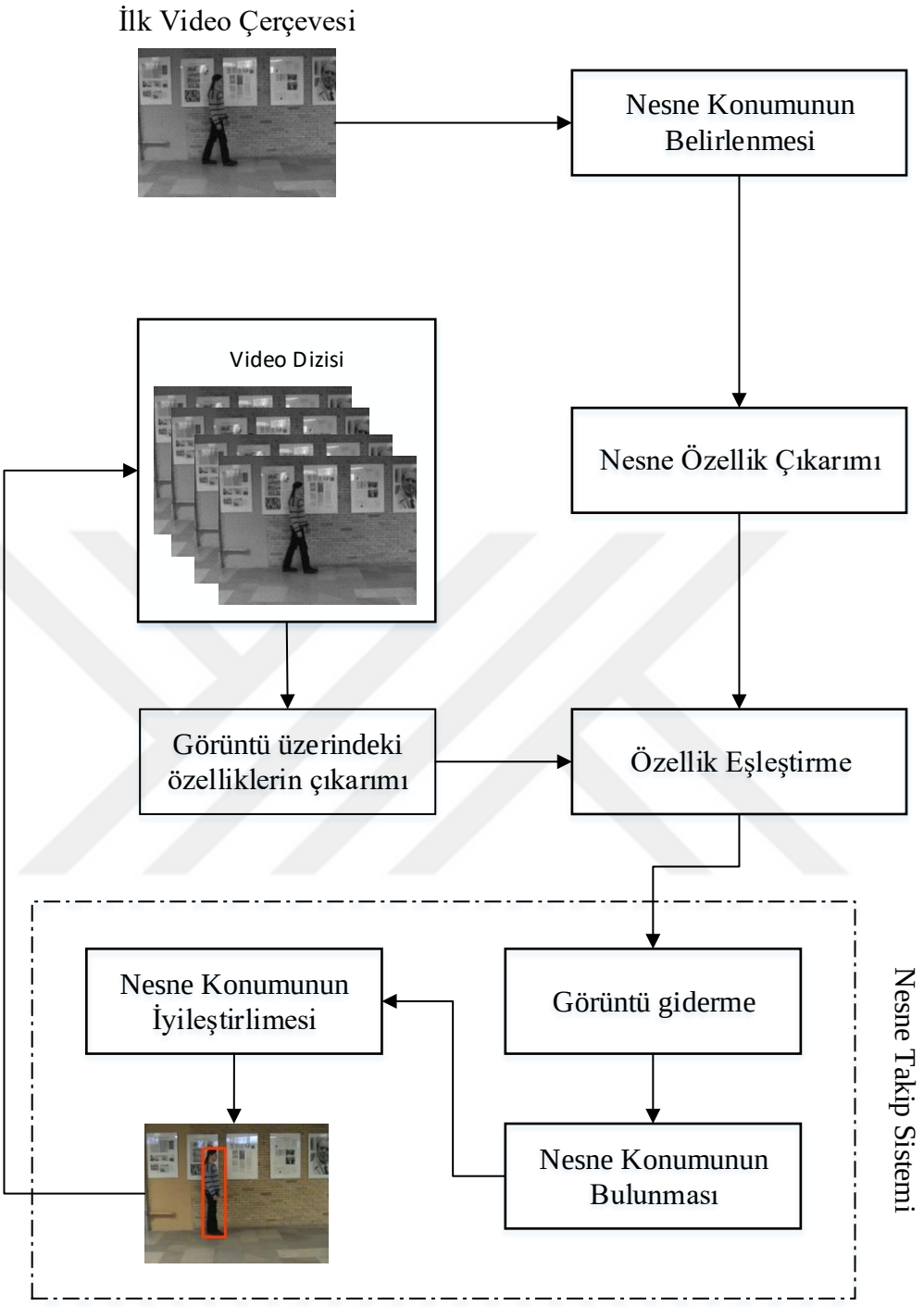
Gerçekleştirilecek uygulama gerçek/yarı gerçek zamanlı çalışacak ve hesaplama gereksiniminin yüksek olacağı kesindir. Bu yüksek hesaplama gereksinimini karşılamak amacıyla işlemler GPU tarafında çalışan uygulamalar kullanılacaktır.

Çalışmada ilk yapılacak işlem, video görüntülerinde nesne tespit işlemidir. Bu işlem için nokta tabanlı özellik çıkarım algoritmaları kullanılmıştır. Nesne tespiti, verilen referans imgesi ile eşleşen nesnenin tespit edilmesi şeklinde gerçekleştirilir. Uygulama da sadece ilgili nesnenin tespit edilmesi ve doğru bir şekilde izlenmesi gerekmektedir. Bu kapsamda ilgili olmayan nesnelerin tespiti gereksizdir. Üçüncü olarak, ilgili nesne tespit edildikten sonra bu nesnenin doğru bir şekilde izlenmesi gerekmektedir. Gerçek dünya ortamı düşünüldüğünde nesnelerin birbiri ile etkileşimleri önemlidir. Etkileşim sırasında nesne takibinin kaybedilmemesi veya kaybedilse bile etkileşimin bittiğinde nesne izlemenin tekrar devam etmesi gereklidir. Nesne izleme işlemi videonun ilk çerçevesi ile başlayacak ve özellik çıkarımı hem ilk çerçeve için hem de referans imgesi için gerçekleştirilecektir. Nesne izleme işlemi ise ikinci video çerçevesi ile başlayacaktır. Nesne kamera görüş alanından çıktıktan sonra eğer tekrar görüş alanına girerse özellik çıkarımı gerçekleştirilecek, nesne tespit edilecek ve nesne izleme işlemi başlatılacaktır.

Tez çalışmasında geliştirilecek olan nesne takip sisteminde literatürde kabul görmüş bir nesne izleme veri seti kullanılarak nesne takip sistemi çalıştırılacaktır.

1.2. Tezin Yapısı ve İçeriği

Tez çalışmasında C dili ve OpenCV (Open source computer vision) kütüphanesi kullanılarak, nesne izleme uygulaması geliştirilmiştir. Geliştirilen sistemin akış şeması Şekil 1.3'te verilmiştir. Yazılım Visual Studio 2013 ortamında geliştirilmiştir.



Şekil 1.3. Çalışmada kullanılan sistemin ana yapısına ait akış şeması

Tezin yapısı 5 ana bölümden oluşmuştur. Birinci bölümde, nesne izleme ve tespiti ile ilgili genel bilgiler verilmiştir. Tezin ikinci bölümünde nesne takip sisteminin için literatür özeti verilmiştir. Bu kapsamda nesne tespiti, nesne modelleme ve nesne izleme yöntemleri ile ilgili yapılan çalışmalar incelenmiştir. Üçüncü bölümde nesne tespiti, eşleştirme ve izleme uygulamaları için SIFT (Scale Invariant Feature Transform), SURF (Speeded-Up

Robust Features) ve GPU tabanlı SURF (GPU-SURF) algoritmalarının karşılaştırmalı analizi verilmiştir. Bu bölümde SIFT, SURF ve GPU-SURF algoritmalarının hem nesne eşleştirme için kullanımı hem de nesne izleme uygulamalarında kullanımı ayrıntılı olarak karşılaştırılmıştır. Dördüncü bölümde nokta tabanlı özellik çıkartma algoritmaları için DBScan (Density-based spatial clustering of applications with noise) ve Gauss algoritmaları ile yeni bir nesne izleme algoritması önerilmiştir. Önerilen çalışma ile klasik SIFT, SURF ve GPU-SURF yöntemleri nesne izleme uygulaması üzerinde başarı oranları incelenmiştir. Yapılan deneyler sonucunda geliştirilen uygulamanın çok daha başarılı sonuçlar elde edildiği gösterilmiştir. Son bölümde ise elde edilen sonuçlar değerlendirilmiş ve yazarların ilerde yapmayı planladıkları çalışmalara yer verilmiştir.

2. LİTERATÜR ÖZETİ

Günümüzde gerçek dünya ortamlarında gözetim uygulamalarının kullanımı sıklaşmaktadır. Bu uygulamalar ile ilgilenilen nesnenin tespit edilmesi, tanınması ve takip edilmesi büyük önem arz etmektedir. Görsel analiz ile düşük seviye imge bilgisinden yüksek seviye olay analizi elde etmek için 3 temel adım uygulanmaktadır. Bunlar:

- Nesnenin tespiti
- İlgili nesnenin her video çerçevesinde takibi
- İzleme sonuçlarının değerlendirilmesi sonucu gerçekleşen olayı ortaya çıkartmak.

Yukarıda verilen adımlar erişim kontrolü, hareket tabanlı tanıma, insan bilgisayar etkileşimi ve trafikte araç izleme gibi diğer uygulamalara da uyarlanabilmektedir. Aslında bu tip uygulamalarda temel nokta nesnenin tespit edilmesi ve video boyunca bu nesnenin takip edilmesidir. Nesne tespiti ve izleme ile ilgili literatürde birçok yaklaşım önerilmiştir. Bu yaklaşımların her birisi izlemede hangi gösterimin kullanıldığı, nesnenin hareketi ve görünümünün nasıl modellendiği, hangi imge özelliklerinin kullanıldığı gibi birçok farklılık içermektedir. Bu bahsedilen özelliklerin birbirinden farklı olduğu birçok algoritma önerilmektedir. Bu bölümün amacı, nesne tespiti ve izleme ile ilgili geliştirilen algoritmaları analiz etmek ve tez çalışmasında esinlenen konulara yer vermektir.

2.1. Nesne Tespiti

Nesne tespiti nesne izleme için gerekli bir işlemdir ve her video çerçevesi için uygulanmalıdır. Nesne tespitinin genel kullanımı, tek bir video çerçevesi ile nesne tespitidir. Fakat hatalı nesne uyarılarını azaltmak için bazı nesne tespit uygulamaları birden fazla video çerçevesi ile hesaplama yapılmaktadır. Bu bilgi genel olarak video çerçeve farkları ile tespit edilir ve nesnenin olduğu bölge nesne izleyicisine verilir. İmgede verilen nesne bölgesi dikkate alındığında nesne izleme uygulaması, nesnenin bir video çerçevesinden diğerine hareketini gerçekleştirir. Nesne tespiti ile ilgili birçok yöntem önerilmiştir.

2.1.1. Nokta Tabanlı Özellik Tespiti

Nokta tabanlı özellik tespit yöntemleri, imge üzerindeki ayırt edici özellikleri bulmak için kullanılırlar. Literatürde birçok nokta tabanlı özellik tespit algoritması vardır. Bu algoritmaların istenilen özelliği ise ışık, ölçek ve kamera bakış acısı değişiminden etkilenmemesidir. Yaygın olarak kullanılan nokta tabanlı özellik tespit uygulamaları aşağıda verilmiştir [7].

- 1979 yılında Moravec tarafından geliştirilen Moravec Operatörü [8],
- 1988 yılında Harris ve Stephens tarafından geliştirilen Harris Nokta Detektörü[9],
- 1994 yılında geliştirilen Shi ve Tomasi tarafından geliştirilen KLT (Kanade-Lucas-Tomasi) Detektörü [10],
- 2004 yılında Lowe tarafından geliştirilen SIFT Detektörü [11].

Operatörlere baktığımızda her birisinin gelişimi devam etmiş ve bu yöntemlerin zayıf tarafları geliştirilmiştir. Örneğin SIFT yönteminden esinlenerek ASIFT (Afin SIFT) [12] ve SURF [13], RIFF[14] gibi birçok yöntem literatüre kazandırılmıştır.

Moravec operatörü ayırt edici noktayı bulmak için, görüntü yoğunluklarının yatay, dikey, çapraz ve ters çapraz yönlerde 4 × 4 boyutunda değişimi hesaplar ve bu dört hesaplamanın minimum değerini, pencereleri temsil etmek için seçer. Böylece nesnenin aslında kenar haritası çıkarılmış olur. Daha sonra kenar haritasındaki tüm kenarların maksimum sonuçları hesaplanır. Böylece komşusuna göre daha yüksek sonuca erişen nokta ayırt edici nokta olarak seçilir [15].

Harris kenar bulma algoritması da ayırt edici nokta bulmak amaçlı kullanılmaktadır. Temel olarak, (u, v) koordinatlarının her yöne kayması için yoğunluk farkı Denklem 2.1 ile bulunur [9].

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{2.1}$$

$w(x,y)$ pikseller için ağırlık değerleridir. Nokta tespiti için $E(u,v)$ fonksiyonu maksimize edilmesi gerekir. Yukarıdaki denkleme Taylor açılımı uygulanır ve sonuç olarak Denklem 2.2 elde edilir [9].

$$E(u,v) \approx [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.2}$$

M ifadesi Denklem 2.3'te verilmiştir.

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_y I_x \\ I_x I_y & I_y I_y \end{bmatrix} \tag{2.3}$$

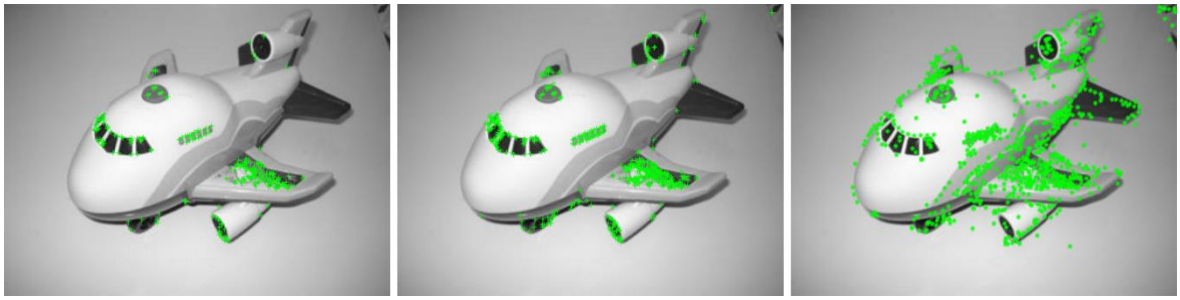
Burada I_x ve I_y, x ve y yönlerinde imge türevleridir. Bu işlemden sonra belirlenen çerçevenin ayırt edici nokta içerip içermediği Denklem 2.4 ile belirlenir.

$$R = \det(M) - k(tr(M))^2 \tag{2.4}$$

Burada k sabittir ve belirlenen eşik değeri R değerini geçerse nokta *önemli nokta* olarak belirlenmektedir [9].

Denklem 4.2'de verilen M moment matrisi KLT dedektörü için de kullanılmaktadır. Aynı şekilde T eşik değeri olarak kullanılır ve aday noktalar T eşik değerini geçen noktalar arasından seçilir. Son olarak KLT birbirine yakın aday noktaları ortadan kaldırır. KLT ve Harris benzer biçimde yoğunluk farklılıklarını vurgular. Pratik olarak farkları KLT yönteminde ek olarak ayırt edici noktaların birbirine uzaysal olarak belli uzaklıkta olmasını zorlar [10].

Teorik olarak M matrisi döndürme ve ötelemeye karşı dayanıklıdır fakat afin dönüşümüne ve kamera açısının değişimine karşı dayanıklı değildir. Farklı dönüşümlere karşı dayanıklı bir ayırt edici nokta algoritması olarak SIFT yöntemi önerilmiştir. SIFT yöntemi ile KLT ve Harris yöntemlerine göre daha fazla özellik noktası bulunabilmektedir [7]. Bunun nedeni SIFT algoritması ile farklı ölçekte ve farklı çözünürlükte özelliklerinde hesaplanmasıdır. Ayrıca diğer algoritmalara göre de yine imge deformasyonlarına daha dayanıklı olduğu yapılan çalışmalarda ortaya konulmuştur. Şekil 2.1'de Harris, KLT ve SIFT operatörlerinin aynı imge üzerinde bulduğu özellik noktaları gösterilmiştir. Şekilde görüldüğü gibi SIFT yöntemi ile daha fazla özellik noktası bulunabilmektedir.



(a) (b) (c)

Şekil 2.1. (a) Harris, (b) KLT ve (c) SIFT yöntemleri ile bulunan özellik noktaları

2.1.2. Arka Plan Çıkarımı

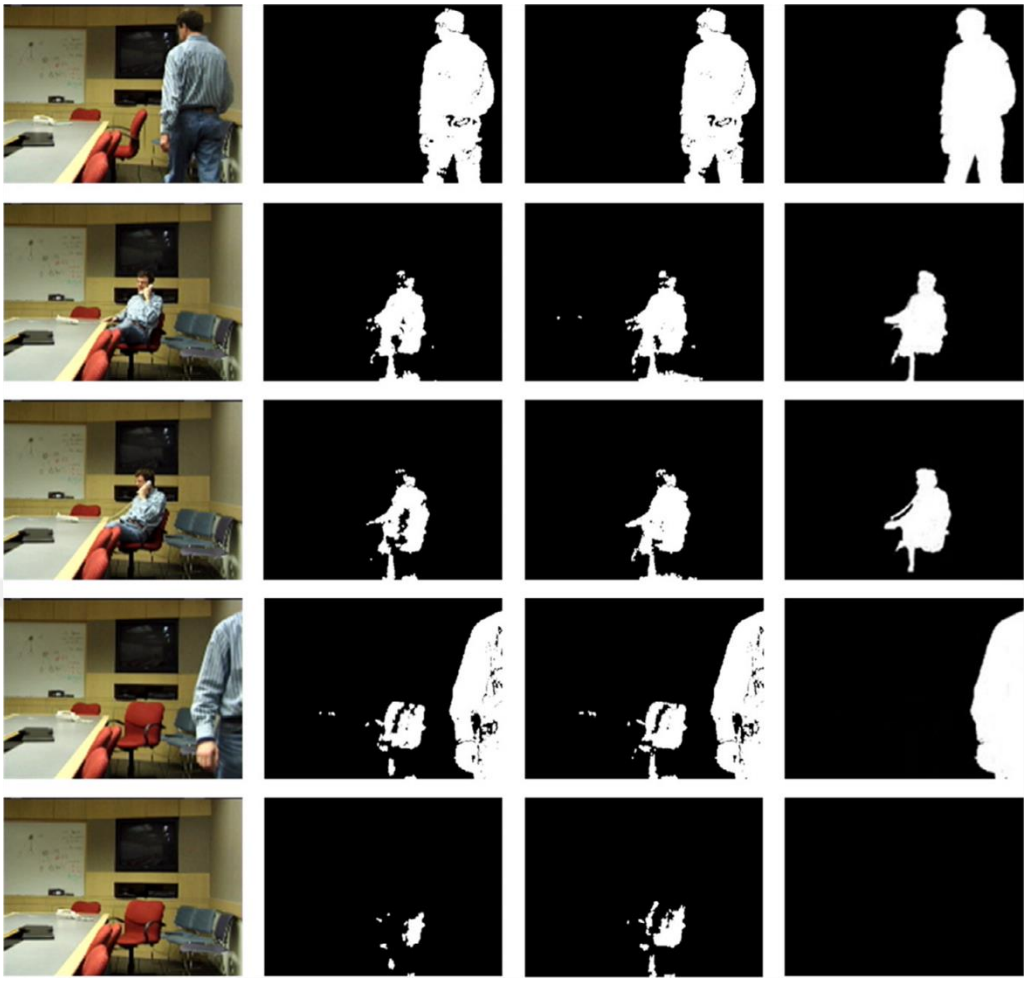
Nesne tespiti için bir başka model, sahnenin genel hali kullanılarak arka plan modeli oluşturulmasıdır. Ardından gelen her video çerçevesi için oluşturulan modelden sapmalar incelenerek nesne tespiti gerçekleştirilir. Arka plan modelinden belli miktarda sapma meydana geldiğinde bu hareket eden bir nesneyi işaret etmektedir. Değişim olan pikseller bir sonraki işlem için işaretlenir ve nesnelere ait olan bölgenin belirlenmesi için arka plan çıkarımı işlemi uygulanır.

Arka plan çıkarımı işlemi için kullanılan en temel yöntem 1979 yılında Jain ve Nagen [16] tarafından geliştirilen yöntemdir. Bu yöntemde birbirini takip eden iki video çerçevesinin farkı alınarak hareket eden nesnelerin tespit edilmesine dayanmaktadır. Fakat bu yöntem ışık değişimine duyarlıdır ve sabit nesneleri yakalamada başarılı değildir. Arka plan çıkarımı Wren ve dig. [17] tarafından geliştirilen yöntem ile popülerlik kazanmıştır. Bu yayında zaman içinde gerçekleşen kademeli değişimleri öğrenmek için, her bir $I(x, y)$ pikseli için renk modellemesinin yapılması önerilmiştir. Önerilen modelde sabit bir arka plan ve *YUV* renk uzayında 3 boyutlu Gauss kullanılmıştır. Gauss model parametreleri $\mu(x, y)$ ve $\sum(x, y)$ önceki video çerçevesinin renk gözlemlerinden öğrenilmektedir. Giriş çerçevesindeki her bir (x, y) pikseli için öğrenme işlemi tamamlandığında, $N(\mu(x, u), \sum(x, y))$ hesaplaması sonucu elde edilen değer ile pikselin benzerliği hesaplanır. Arka plandan sapan pikseller nesne pikselleri olarak işaretlenir. Ancak tek bir Gauss kullanmak gerçek dünya ortamları için pek uygun değildir. Örneğin Stauffer ve Grimson [18] piksel renkleri için Gauss karışım yöntemini kullanmıştır. Bu yöntemde çok modelli istatistiksel bir model kullanılarak arka plan modeli tanımlanmış ve gelişim sağlanmıştır. Gauss karışım yöntemi, o anki video çerçevesindeki piksel ile karışımdaki her modeli

karşılaştırır. Eğer eşleşme bulunursa Gauss modelindeki ortalama ve varyans değerleri güncellenir. Aksi takdirde o anki piksel rengine eşit bir ortalamaya sahip olan Gauss modeli bazı başlangıç değerleri ile tanımlanır ve karışım içerisine eklenir. Bu işlemler sonucunda her piksel Gauss dağılımı ile eşleşip eşleşmediğine göre sınıflandırılır. Eşleşmeyen pikseller hareket eden pikselleri göstermektedir [18].

Bir başka yöntem ise renk tabanlı bilgiyi kullanmak yerine, Elgammal ve dig. [19] tarafından önerilen bölgeye dayalı sahne bilgisini kullanmaktır. Bu modelde tek bir piksele yoğunlaşmak yerine, Gauss karışım modeli komşu pikselleri de dâhil edilecek şekilde genişletilmiştir. Dolayısıyla çıkartma işlemi sadece arka plan modelindeki ilgili piksel ile değil, yakınındaki pikseller de işleme dâhil edilerek gerçekleşir. Böylece kamera sarsıntısı ve arka plandaki küçük hareketler uygulamanın çalışmasını etkilememektedir [19].

2010 yılında Olivier ve Marc Van [20] tarafından geliştirilen yöntemde ViBe (Visual Background Extractor) olarak isimlendirilen evrensel arka plan çıkarımı algoritması tanıtılmıştır. Önerilen yöntemde üç yenilikçi teknik birleştirilmiştir. İlk olarak, yeni bir sınıflandırma modeli önerilmiştir. Bu modelde arka plan pikselleri için bir model oluşturmak yerine, her arka plan pikselini bir dizi örnekle modellemeye dayanmaktadır. İkinci olarak, ViBe algoritmasının başlaması için sadece bir video çerçevesi yeterlidir. Böylece gömülü sistemler gibi hemen cevap almanız gereken uygulamalar için uygun bir algoritma geliştirilmiştir. Son olarak, yeni bir arka plan güncelleme algoritması sunulmuştur. Arka plan modelindeki örnekleri belli zaman tutup silmek yerine, geliştirilen model rastgele bir değer seçerek bu işlemi gerçekleştirir. Bu işlem piksel örneklerinin yavaş değişmesine olanak tanırken, arka planın hızlı değiştiği durumlarda modelin uygun davranmasına neden olmaktadır [20]. 2015 yılında Lixia Qin ve dig. [21] tarafından yapılan çalışmada video üzerinde arka plan çıkarımı işlemi Gabor detektör kullanılarak gerçekleştirilmiştir. Gerçekleştirilen yöntem ViBe algoritması tabanlıdır fakat yöntemde ek olarak Gabor dalgacıkları ile elde edilen kenar bilgileri de kullanılmaktadır. Ayrıca algoritmanın hızlı olması için arka planın ön plandan çıkarılması, arka plan modelinin başlatılması ve arka planın güncellenmesi işlemleri GPU üzerinde gerçekleştirilmiştir. Şekil 2.2'de örnek bir video üzerinde ViBe algoritması ile Lixia Qin tarafından geliştirilen yöntemin arka plan çıkarımı sonuçları verilmiştir [21].



Şekil 2.2. ViBe algoritması ile Lixia Qin tarafından geliştirilen yöntemlerin arka plan çıkarımı sonuçları. (a)Orijinal video çerçevesi (b) Lixia Qin yöntemi (c) ViBe (d) Zemin gerçeği [21].

2.1.3. Bölütleme

İmge bölütlemenin amacı, imge üzerindeki benzer yoğunluklu bölgelerin ayırma işlemidir. Her bölütleme algoritması imge üzerindeki bölgeleri daha etkili ve başarılı bir şekilde ortaya koymaya çalışmaktadır. Bu bölümde nesne izleme ile ilgili yaygın olarak kullanılan bölütleme algoritmaları incelenecektir.

İmge bölütleme için kullanılabilecek temel yöntemlerden biri *k*-ortalama kümeleme yöntemidir. Diğer kümeleme algoritmalarına göre daha hızlı ve daha basit olduğu için imge bölütleme amaçlı kullanılmaktadır. Yöntem parametre olarak bölünecek küme sayısını almaktadır. Bu parametre yönteme esneklik sağlar fakat farklı parametrelerde sonuçlar farklı olmaktadır. Yöntem başlatılırken küme merkezleri rastgele olarak veya dışarıdan parametre olarak alınabilmektedir. Yine küme merkezlerine göre de farklı sonuçlar elde edilmektedir.

Dolayısıyla k-ortalama kümeleme yöntemi ile imge bölütleme işlemini uygun parametreler ile başlatmak büyük önem arz etmektedir [22]. 2013 yılında Pallavi ve Ritesh [23], k-ortalama kümeleme algoritmasına yönelik yeni ve verimli bir yaklaşım geliştirmişlerdir. Geliştirilen yöntemde uygulama çalışma zamanından ödün verilmeden, oluşan kümelerdeki ortalama karesel hatayı azaltmaktadır. Alan ve dig. [24] tarafından geliştirilen yöntemde ise beyin tümörlerinin bölütlenmesi amacıyla k-ortalama kümeleme ve fuzzy C-ortalama algoritmaları kullanılmıştır. Yöntemde üç ana işlem bulunmaktadır. Bunlar, imgenin önişlemden geçirilmesi, geliştirilmiş k-ortalama ve fuzzy c-ortalama algoritmalarının uygulanması ve son olarak özellik çıkarımı adımlarıdır. İlk adımda filtreler uygulanarak görüntünün kalitesi iyileştirilir. Önerilen k-ortalama kümeleme yöntemi uygulanır ve imge fuzzy c-ortalama algoritması kullanılarak parçalara ayrılır. Son olarak ortaya çıkan bölümlenmiş imgedeki ilgilenilen bölüm için özellik çıkarılma işlemi uygulanmaktadır [24].

2002 yılında Comaniciu ve Meer [25] tarafından geliştirilen yöntemde bölütleme için ortalama kayma yaklaşımı önerilmiştir. Yöntemde $[l, u, v]$ renk bilgilerinin yanı sıra $[x, y]$ mekânsal bilgiler de kullanılmıştır. Bir imge verildiğinde, algoritma, verilerden rastgele seçilen çok sayıda rastgele küme merkezi ile başlatılır. Daha sonra her küme içerisinde yer alan noktaların ortalaması ile küme merkezi güncellenir. Eski ve yeni küme merkezinin kayması sonucu ortalama kayma vektörü oluşturulur. Ortalama kayma vektörü yenilemeli şekilde küme merkez noktaları değişmeyene kadar hesaplanır. Ortalama kayma yönteminin doğru bir şekilde bölütleme elde etmek için seçilen parametreler önemli yer tutar. Örneğin renk ve mekânsal parametrelerin aralıkları ve bölge boyutları için eşik değeri bölütleme sonucuna etki etmektedir.

İmge bölütlemenin bir diğer gösterimi graf bölümleme olarak da formüle edilebilmesidir. Bir G grafında düğümler $V = \{u, v, \dots\}$ şeklinde gösterilir. İmge olarak düşünürsek düğümler pikselleri ve graf da imge olarak düşünülebilir. Verilen G grafının ağırlıklı kenarları budanarak birbirinden bağımsız N tane alt grafa parçalanır. İki alt graf arasındaki budanmış kenarların toplam ağırlığına kesme denir. Kenar ağırlıkları düğümler arasındaki renk, parlaklık veya doku benzerliği ile hesaplanmaktadır. Wu and Leahy [26] tarafından geliştirilen algoritmada amaç "kesimi" en aza indiren bölümleri bulmak olduğu için minimum kesim kriterini kullanmıştır. Bu algoritmada ağırlıklar renk benzerliğine göre tanımlanmaktadır. Minimum kesim işleminde ise ortaya aşırı bölütleme problemi çıkabilmektedir. Shi ve Malik [27] aşırı bölütleme problemi ile başa çıkmak için normalize kesim kullanmaktadır. Bu yaklaşımda her piksel bir düğüm olarak alınır ve her piksel çifti

ise bir kenara bağlanarak G grafı oluşturulur. Sadece piksel konum ve yoğunluk bilgileri kullanılarak kenar ağırlığı tanımlanmaktadır. Her piksel çifti arasındaki ağırlıklar hesaplandıktan sonra, W ağırlık matrisi ve D köşegen matrisi oluşturulur. Bölütleme işlemi için önce öz vektörler ve öz değerler Denklem 2.5'te verilen genelleştirilmiş öz değer sistemi ile hesaplanır.

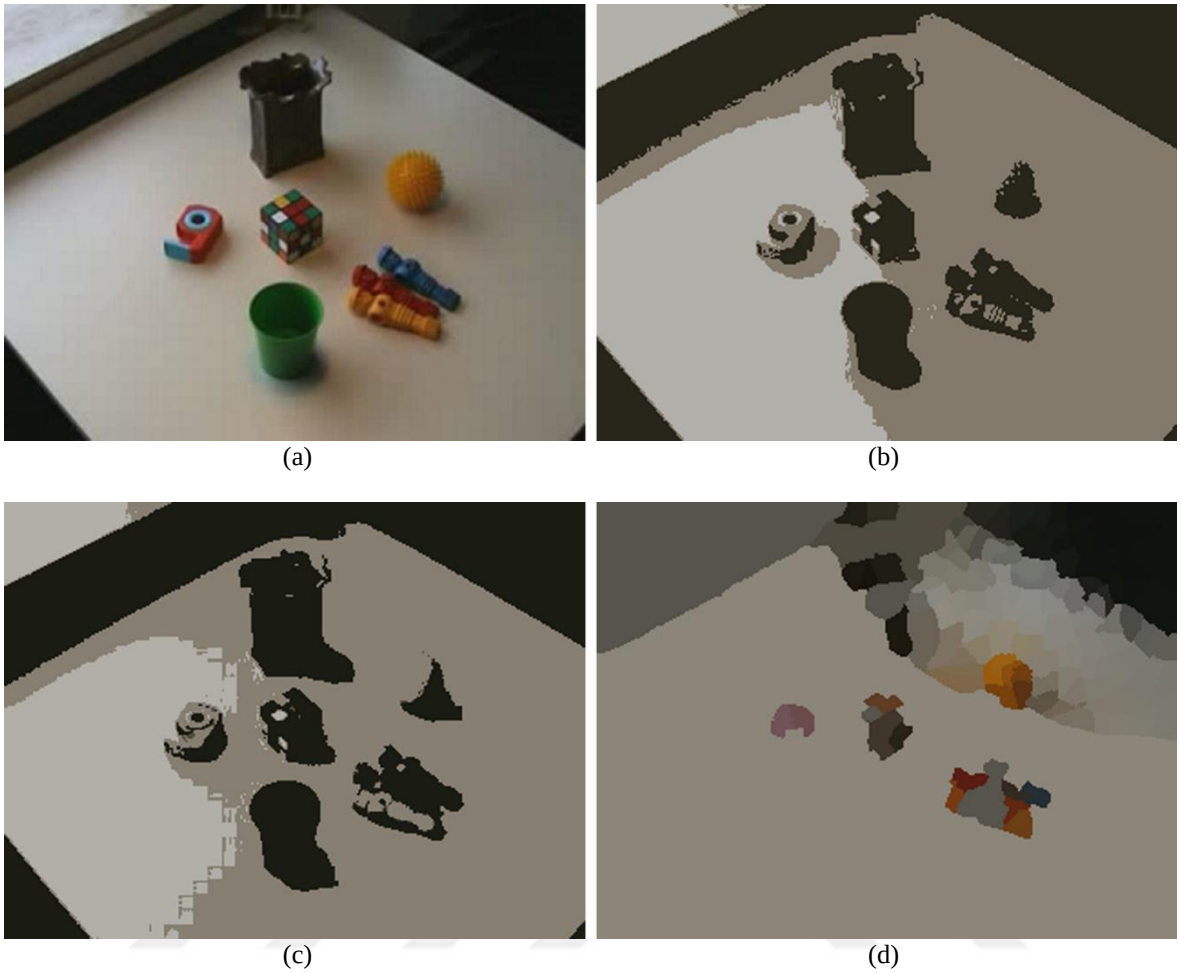
$$(D - W)y = \lambda Dy \tag{2.5}$$

Yöntemin devamında, ikinci en küçük öz vektör kullanılarak imge iki bölüme ayrılır. Her yeni bölüm için, bu işlem belli bir eşik değerine ulaşıncaya kadar tekrar edilir. Fakat bu algoritma ile büyük imgelerde çözüm üretmek yüksek seviyede hesaplama kapasitesi ve yüksek hafıza gerektirmektedir. Buna karşın, ortalama kayma yöntemine göre daha az sayıda elle parametre seçimi gerektirmektedir [27]. Ayrıca normalize edilmiş kesim yöntemi nesne konturlerinin izlenilmesi amaçlı da kullanılmıştır [28]. Şekil 2.3'te örnek bir imge için k-ortalama kümeleme (b), ortalama kayma (c) ve normalize edilmiş kesim (d) yöntemleri ile elde edilmiş bölütleme sonuçları verilmiştir.

Aktif kontur yöntemi bir diğer bölütleme yöntemidir. Nesne bölütleme işlemi, nesne sınırlarını saracak sıkı ve kapalı bir konturün geliştirilmesine dayanmaktadır. Konturun gelişimi, nesne olduğu varsayılan bölgeye uygunluğunu tanımlayan bir enerji fonksiyonu ile kontrol edilir. Kontur gelişimi için kullanılan enerji fonksiyonu Denklem 2.6'da verilmiştir.

$$E(C) = \int_0^1 E_{int}(v) + E_{im}(v) + E_{ext}(v)ds \tag{2.6}$$

Burada s, C konturunun uzunluğu, E_{int} düzenleme enerjisini yani eğrinin eğinden dolayı oluşan iç enerjisini, E_{im} imgenin görünüşünden elde edilen enerjiyi ve E_{ext} ise sınırlama kuvvetini belirtmektedir. E_{int} genellikle en kısa konturu bulmak için birinci dereceden ∇v ve ikinci dereceden $\nabla^2 v$ süreklilik terimlerini içerir. E_{im} ifadesi ise lokal veya global olarak hesaplanabilir. Bölgesel bilgi genel olarak kontur çevresinde imge yoğunluğundaki yönlü değişimin hesaplanması ile elde edilir. Global bilgi ise nesne bölgesinin içi veya dışı olarak hesaplanmaktadır.



(a) (b)

(c) (d)

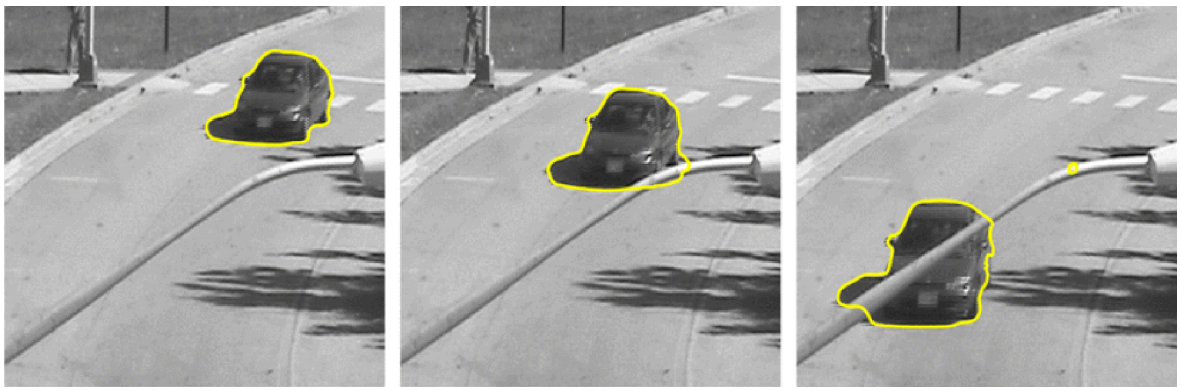
Şekil 2.3. Örnek bir imge için bölütleme sonuçları. (a) orijinal imge, (b) *k*-ortalama kümeleme, (c) ortalama kayma, (d) normalize dilmiş kesim.

Enerji fonksiyonunun biraz değiştirilmesi ile birlikte farklı yaklaşımlar geliştirilmiştir. Caselles ve dig. [29] tarafından geliştirilen yöntemde, E_{im} olarak sadece imge üzerindeki yönlü değişimleri kullanmıştır ve E_{ext} ifadesini denklemden çıkartmıştır. Fakat imge gradyanların lokal bilgi içerdiğinden dolayı lokal minimum değerlere duyarlıdır. Bununla başa çıkmak için Zhu ve Yuille [30] tarafından önerilen algoritmada bölgesel tabanlı imge enerjisi önerilmiştir. Fakat bu yöntemde de nesne konturunun nesneye tam olarak yerleşmemesi olarak sonuç vermiştir. Paragios ve Deriche [31] tarafından önerilen yöntemde ise E_{im} ifadesi için her iki yaklaşımın da Denklem 2.7 ile birleştirilmiştir. Ayrıca $E_{bölge}$ ifadesindeki görünümü modellenmesi için Gauss karışım yöntemi uygulanmıştır [7].

$$E_{im} = \lambda E_{sınır} + (1 - \lambda) E_{bölge} \tag{2.7}$$

Kontur tabanlı yöntemlerde en büyük problem konturun başlatılmasıdır. İmge Gradyanı kullanılan yöntemlerde kontur nesneyi içine alacak şekilde başlatılması gerekmektedir. Daha sonra ise kontur küçülerek nesneye uymaktadır. Nikos ve Rachid [31] tarafından geliştirilen yöntemde bu kısıtlama giderilmiştir. Böylece konturun başlaması nesne içinde veya dışında olabilir. Konturun nesneyi kestiği durumlarda ise konturun bir bölümü genişlerken diğer bölümü daralmaktadır. Fakat bu yöntemde de nesnenin veya arka planın ön bilgisine ihtiyaç duyulmaktadır [31].

Seviye kümesi yöntemi ile bölütleme işlemi de kontur tabanlı yöntemlere örnek olarak verilebilir. Klasik seviye kümesi yönteminde eğri gelişimi, eğrinin geometrik özelliklerine dayanmaktadır. 2006 yılında Marc ve dig. [32] tarafından geliştirilen yöntemde ise kontur eğrisine hız alanı özelliği eklenmiş ve eğri dinamik olarak geliştirilmiştir. Normal geometrik eğri gelişimi için kısmi seviye kümesi yaklaşımı kullanılmıştır. Kısmi seviye kümesi yöntemi yaklaşımında, topolojik değişikliklere izin vermek için seviye kümesi formülü kullanılır. Kontur üzerindeki her noktaya ilişkin hız bilgisi, yayılım denklemleri ile kontur boyunca yayılmaktadır. Kısmi seviye kümesi yaklaşımı, klasik seviye kümesi yöntemine göre daha az hesaplama kapasitesine ihtiyaç duymaktadır [32]. Şekil 2.4'te yöntemin örnek bölütleme sonuçları gösterilmiştir.



Şekil 2.4. Marc ve dig. tarafından geliştirilen yöntemde örnek bir imge serisi için bölütleme sonuçları [32].

2012 yılında Weiming ve dig. [33] tarafından yapılan çalışmada seviye kümesi kullanan kontur tabanlı izleme yöntemi sunulmuştur. Yöntemin bileşenleri aşağıda verilmiştir:

- İzlemenin başlaması: Kontur tabanlı izlemenin başlaması için ilk video çerçevesinde çalışacak optik akış tabanlı bir algoritma geliştirilmiştir.
- Renk tabanlı kontur gelişimi: Komşu piksellerin değerleri arasındaki ilişki Markov rastgele alan teorisi kullanılarak oluşturulur. Böylece renk tabanlı algoritma, arka

plan bozulmalarına karşı dayanıklı olmasının yanı sıra sıkı ve pürüzsüz hatlara ulaşması sağlanır.

- Uyarlanabilen şekil tabanlı gelişim: genel şekil bilgisi ile renk bilgisi birleştirilerek esnek bir şekil güncelleme modeli oluşturulur.
- Dinamik şekil tabanlı gelişim: Markov model tabanlı dinamik şekil modeli oluşturulmuştur. İzleme sırasında kontur renk bilgisi ile gelişir ve şekil modeli o anki şekil modelini tahmin etmeye başlar.
- Ani hareketler: Parçacık sürü optimizasyonu ile nesne hareketi yakalanmıştır. Bir sonraki çerçevede konturun üretilmesi için o anki çerçevede hesaplanmaktadır.

Yapılan çalışmanın örnek bir bölütleme sonucu Şekil 2.5'te verilmiştir. Şeklin ilk satırında el ile kapatılan yüz bölgesi için sadece renk tabanlı kontur gelişimi algoritmasının uygulaması sonucu elde edilen izleme verilmiştir. İkinci satırda uyarlanabilen şekil tabanlı kontur gelişim uygulanmıştır. Üçüncü satırda ise hem el hem de yüz için iki ayrı uyarlanabilen şekil tabanlı kontur gelişimi tanımlanmıştır [33]. 2017 yılında Modova ve Akbarizadeh [34] tarafından yapılan çalışmada ise seviye kümesi yöntemi ile radar görüntülerinden kıyı şeridinin tespiti yapılmıştır. Önerilen yöntemde başlangıç için konturun ilk şeklinin belirlenmesine gerek duymamaktadır. Ayrıca yapılan testlerde düşük çözünürlükte ve gürültülü imgelerde de başarılı sonuçlar elde edilmiştir.

Şekil 2.5. Weiming ve dig. tarafından yapılan yöntemde örnek bir imge serisi için bölütleme sonuçları [33].

İmge bölütleme için kullanılan bir değer yöntem hareket bölütleme yöntemidir. Bu yöntemde bir grup pikselin hızını ve yönünün belirlenmesi ile hareket modellenmektedir. Hareket bölütleme de genel olarak ilk işlem görüntüler üzerindeki optik akışın hesaplanmasıdır. Optik akış, imge üzerindeki farklı nesnelerin ayırt etmede yaralıdır. Optik akış hem kamera hareketini hem de nesne hareketini gösterebilir. İmgeler üzerindeki ışık değişiminin yumuşak olduğunu varsayarsak, imge yoğunluklarındaki değişim ile akış hızı az bir hata ile bulunabilmektedir. Akış tabanlı hareket bölütleme, hareket tahmini problemleri ve iki ardışık video çerçevesindeki hareket analizi gibi birçok işlemde kullanılmaktadır. Optik akışın doğru bir şekilde hesaplanabilmesi için ortam ışığının ve nesne hızının sabit olduğu varsayılmaktadır. Dolayısıyla bu durumlardan sapma olursa optik akış hesaplaması hatalı olabilmektedir. Örneğin gerçek ortam şartlarında sıkça hatalar olabilmektedir. Ortaya çıkan hatalar tüm optik akış boyunca yayılmaktadır [35].

2.1.4. Sınıflandırıcılar

Nesne tespiti, nesne görünümünün örnek eğitim verileri ile farklı versiyonlarının öğrenilmesi ile gerçekleştirilebilir. Sınıflandırıcıların performansı seçilen eğitim verileri ile direk olarak bağlantılıdır. Seçilen verilerde sınıf ayrımının iyi bir şekilde yapılması gerekmektedir. Veriler seçildikten sonra sinir ağları, karar destek ağaçları, destek vektör makinaları gibi farklı öğrenme yaklaşımları uygulanabilmektedir. Öğrenme algoritmaları ile nesnenin hangi sınıfa ait olduğunun belirlenmesi amaçlanmaktadır [35].

Hızlandırıcılarda amaç zayıf sınıflandırıcıların birleşimi ile güçlü bir sınıflandırıcı elde etmektir. İteratif bir yöntem olup temel sınıflandırıcıların birleşimi ile çok güçlü sınıflandırıcılar elde edilebilir. Freund ve Schapire [36] tarafından geliştirilen Adaboost yöntemi birden fazla zayıf öğreticiyi tek bir güçlü öğretici elde etmek için kullanır. Adaboost algoritmasındaki zayıf öğreticiler karar kütüğü şeklinde isimlendirilen tek bir bölümlenmiş karar ağacıdır. Adaboost ile ilk karar kütüğü oluşturulduğunda, tüm gözlemler eşit olarak ağırlıklandırılır. Oluşan hataları düzeltmek için, yanlış sınıflandırılmış gözlemeler doğru sınıflandırılmış gözlemlerden daha fazla ağırlık taşımaktadır. Böylece Adaboost algoritması bir sonraki yinelemede, yanlış sınıflandırılmış veriler üzerinde daha iyi performans gösteren başka bir sınıflandırıcının seçilmesini teşvik etmektedir. Model bu şekilde hatayı en aza indirecek şekilde bir tahminde bulunana kadar çalışmaya devam eder. Adaboost algoritması gibi Gradyan hızlandırıcı [37] algoritması da bir önceki adımdaki oluşan hatayı düzeltmeye

yönelik işlemler gerçekleşir. Adaboost gibi yer yinelemede her yanlış sınıflandırılmış gözlem için ağırlıkları değiştirmek yerine Gradyan hızlandırıcı yönteminde önceki tahminde oluşan hataya uymaya çalışan yöntemi seçmeye çalışır. Yöntem ilk olarak veriye uymaya çalışır, oluşan hata için uygun yöntem oluşturulur ve böylece yeni bir model oluşturulur. Böylece zaman içerisinde oluşan hata azaltılması sağlanır. Tianqi ve Carlos [38] tarafından geliştirilen XGBoost ağaç tabanlı hızlandırma işlemi için ölçeklenebilir bir makine öğrenmesi sistemi geliştirmiştir. Normalde boosting işlemi yapısı gereği kullanılan model eğitim işlemlerinden dolayı oldukça yavaştır. XGBoost sistemi ise bu açığı kapatmak için model performansına ve çalışma hızına odaklanılmıştır. Yapılan çalışmalarda XGBoots sistemi diğer var olan hızlandırıcılardan 10 kat daha hızlı çalıştığı görülmektedir.

Sınıflandırma tabanlı nesne tespitinde yaklaşımlar, arama tekniğine göre iki gruba ayrılabilir. Birinci yöntem, verilen bir imgenin tüm olası görünümlerinin sıralı olarak bir sınıflandırıcıya uygulanmasına dayanmaktadır. İkinci yöntemde ise, nesne parçaları veya ortak şekilleri tespit etmek ve bu yerel özellikleri nihai modeli oluşturmak için birleştirilmesine dayanır.

Sınıflandırma tabanlı nesne tespit yaklaşımlarında ilk gruba örnek olarak Papageorgiou ve Poggio [39] tarafından önerilen çalışma verilebilir. Önerilen yöntemde, nesne tanımlayıcıları olarak Haar dalgacıkları kullanılarak elde edilen bir polinom Destek Vektör Makinesi (DVM) öğrenilmiştir. Daha sonra, yöntem çok sayıda sınıflandırıcı kullanılarak geliştirilmiş, nesne parçalarını tespit etme amaçlı kullanılmıştır [40]. 2005 yılında Dalal ve Triggs [41] tarafından önerilen algoritmada, görüntü gradyan yönelimlerinin normalleştirilmiş yerel histogramlarının yoğun bir ızgarada değerlendirilmesi ile insan tespiti yapılmaktadır. Benzer bir yaklaşım da Zhu ve dig. [42] tarafından önerilmiştir. Bu yaklaşımda yönlendirilmiş gradyanların histogramı (HOG) kullanılarak kademeli bir modelin eğitimi ile gerçek zamanlıya yakın algılama performansları elde edilmiştir. Tuzel ve dig. [43] tarafından 2006 yılında önerilen yöntemde ise Haar dalgacıkları veya HOG özellikleri yerine, bölge kovaryans matrisleri nesne tanımlayıcıları olarak kullanılmıştır. Bahsedilen bölge uzamsal konum, yoğunluk değerleri ve yüksek dereceli türevler gibi görüntü özelliklerinin kovaryans matrisi ile temsil edilmektedir. Kovaryans matrisleri ise bağlı bir Riemann manifoldu olarak gösterilmiştir. Bu tanımlayıcılar bir vektör uzayında olmadığından, sınıflandırıcıların öğrenmesi için geleneksel makine öğrenme teknikleri uygun değildir. Dolayısıyla önerilen manifold öğrenme yöntemi ile Riemann manifoldundaki noktaların sınıflandırılması yapılmıştır [43].

Sınıflandırma tabanlı nesne tespit yaklaşımlarında ikinci gruba örnek olarak Mikolajczyk ve dig. [44] tarafından geliştirilen yöntem verilebilir. Bu yöntemde parçalar yerel yönelim özellikleri ile temsil edilmiş ve AdaBoost kullanılarak her parça için ayrı detektörler eğitilmiştir. Nesne konumu geometrik ilişkilere göre birleştirilmiş parçaların olasılık düzeyini maksimum yapacak şekilde belirlenmiştir. Kalabalık sahneler için insan tespit sistemi Leibe ve dig. [45] tarafından sunulmuştur. Bu yaklaşımda, yerel görünüm özellikleri ile geometrik ilişkileri birleştirilmiştir.

Eğitimli öğrenme yöntemlerinde genel olarak büyük verilerin eğitilmesi gereklidir. Bu gereksinimi azaltmak için eş eğitim tekniği kullanılabilir [46]. Bu yöntemde temel fikir küçük boyutta etiketlenmiş veriler ile iki ayrı sınıflandırıcının eğitilmesidir. Eğitim sağlandıktan sonra her bir sınıflandırıcı, etiketlenmemiş verileri diğer sınıflandırıcının eğitim setine atamak için kullanılır. Eş eğitim, AdaBoost ve destek vektör makineleri yöntemlerinde, eğitim için gerekli olan manuel etkileşim miktarını azaltmak için kullanılmıştır. Zhiquan ve dig. [47] tarafından geliştirilen yöntemde çevrimiçi nesne tespiti için OMILBoost şeklinde isimlendirilen hızlandırıcı bir algoritma önerilmiştir. Bu çalışmada Levin ve diğ. [48] tarafından önerilen eş eğitim yaklaşımı kullanılmıştır. Hareket eden bölgelerin tespiti için ise Javed ve dig. [49] tarafından önerilen yaklaşım kullanılmıştır. İmge parçalarını tanımlayacak olan iki farklı küme elde etmek için, hem Haar benzeri özellikler hem de HOG kullanılmıştır.

2.2. Nesne Modelleme

Nesne izleme uygulamalarında nesne, yoldan geçen arabalar veya insanlar, bir fabrikadaki üretilen ürünler gibi her şey olabilir. Takip edilecek nesnenin konumunu ve diğer özelliklerini takip etmek için nesne özelliklerini uygun bir şekilde seçilecek bir nesne modeline ihtiyaç duyulmaktadır. Bu bölümde nesne izlemede yaygın olarak kullanılan nesne modelleme teknikleri incelenecektir. Nesne modelleme üç alt başlık altında toplanabilir. Bunlar, model gösterimi, model tanımı ve model özellikleridir.

2.2.1. Model Gösterimi

Nesne gösterimi genel olarak uygulamada izlenecek nesneye göre belirlenmektedir. Seçilen model nesnenin şekilsel olarak limitini ve hareketini belirlemektedir. Örneğin elips

gibi bir geometrik modelde nesne hareketine, öteleme, benzerlik, afin veya izdüşüm dönüşümleri uygulanabilir. Ama nesne gösterimi sadece bir nokta ile temsil ediliyorsa nesne hareketine sadece öteleme uygulanabilir. Nokta ile gösterim ve geometrik şekiller genel olarak sert ve değişmez nesnelere uygulanmakla birlikte şekil değiştirebilen nesnelere de uygulanabilir. Şekil değiştiren nesneler için genel olarak siluet ve kontur gösterimi tercih edilmektedir [35].

Nesne gösteriminin en temel yöntemi nokta olarak göstermektir. Nokta nesnenin merkezini gösterebilir veya birden fazla nokta ile şekil temsil edilebilir. Nokta gösterimi genel olarak nesnenin imgeye göre küçük olduğu durumlarda kullanılmaktadır. Bir başka temel gösterim yöntemi de nesneyi dikdörtgen veya elips şeklinde temsil edilmesidir. Şekilsel gösterim değişmez nesneler için en uygun gösterim modeldir [7].

Siluet gösterimi, nesnenin kıvrım sınırının içindeki bölgeyi temsil etmektedir. En yaygın siluet gösterimi, ikili bir fonksiyon ile gösterimdir. Genel olarak nesne bölgesi 1, nesne olmayan bölge ise 0 olarak işaretlenir. Kontur tabanlı gösterimde, siluet açık veya kapalı olarak temsil edilebilir. Açık temsilde siluet sınırını bir dizi kontrol noktası belirlerken, kapalı temsilde ise siluet, tanımlanmış bir fonksiyon aracılığıyla tanımlanır. En yaygın kullanılan kapalı temsil seviye kümesi yöntemidir [50]. Bu yöntem ile seviye kümeyi yöntemi, topoloji değişikliklerine uyum sağlama kabiliyeti, kontur eğriliğinin doğrudan hesaplanması ve seviye kümesi formülünde değişiklik olmadan daha yüksek boyutlara genişletilebilmesi nedeniyle büyük ilgi görmüştür.

Nesne gösteriminde bir başka yöntem ise, nesneyi eklemli parçaları bir araya getirmesi ile nesne oluşturmaktır. Seçilen nesne parçalarına göre nesne oluşturulur. Örneğin bir insan vücudu için nesne, gövde, baş kollar ve bacaklar ile oluşturulabilir. Parçalar dikdörtgen, silindir veya elips ile modellenirken, parçalar arası bağlantılar ise kinematik hareket modeli ile elde edilir. Andriluka ve dig. [51] tarafından 2009 yılında yapılan çalışmada nesne tespiti ve eklemli poz tahmini için genel bir model önerilmiştir. Çalışmada, vücut parçalarının yapılandırılması için esnek bir kinematik ağaç modeli ve ayırt edici biçimde eğitilmiş bir görünüm modeli ve esnek bir kinematik ağaç modeli önerilmiştir. Çalışmada elde edilen örnek poz tahmin sonuçları Şekil 2.6'da verilmiştir [51].

Şekil 2.6. Andriluka ve dig. tarafından yapılan yöntemde örnek bir veri seti için poz tahmin sonuçları [51].

Nesne modeli için iskelet modeli de kullanılabilir. İskelet modeli, bir dizi eğriyi birbirine bağlayan eklemli bir yapıdır. Öğrenilmiş iskelet yapı modelleri birçok olası kullanıma sahiptir. Örneğin, tüm vücut izleme algoritmalarında detaylı ve elle yapılan iskelet modelleri önemli bir bileşendir. İskelet yapısının öğrenilmesi ile birlikte, süreci otomatikleştirmeye yardımcı olabilir ve potansiyel olarak elle yapılan modellere göre daha esnek ve daha doğru modeller üretebilir. Ayrıca iskelet modelindeki özellik noktaları eklem açılarının hesaplanması için gereklidir. Böylece hareket halindeki nesneler de iskelet modeli ile modellenebilir [52].

2.2.2. Model Tanımı

Model tanımı aslında nesnenin matematiksel olarak ifade edilmesidir. Model tanımının elde edilmesinde, bölgenin büyüklüğü, dinamik aralık ve imge üzerinde oluşan gürültüler önemli rol oynamaktadır. Genel olarak nesne bölgesi büyüdükçe, nesne tanımı da daha ayrımcı olacaktır. Çoğu tanımlayıcı için en büyük sorun hem istatistiksel hem de mekânsal özellikleri yakalayan etkili bir benzerlik oranının olmamasıdır. Çoğu yaklaşım sadece yapısal modellere veya renk dağılımına göre işlem yapmaktadır. Nesneleri tanımlamak için birçok farklı gösterim önerilmiştir. Bu bölümde nesne izlemede sıkça kullanılan şablon, histogram, HOG, SIFT, bölge koveryans, öz uzay ve görünüş modellerine yer verilecektir.

Şablon modelleri sezgiseldir ve sıklıkla kullanılmaktadır. Genellikle geometrik şekillerden ve siluetlerden oluşur. Model içerisinde nesne görünüş gözlemlerinin sıralı bir

listesinin olması şablon modelinin benzersiz özelliğidir. Bu özellik şablon tanımlayıcısının, hem mekânsal hem de görünüş bilgisini taşımasına olanak sağlamaktadır. Model tanımı için genel olarak dikdörtgen, kare elips gibi temel geometrik şekiller kullanılmaktadır. Video çerçeveleri arasındaki hareketi ise öteleme gibi parametrik yöntemler kullanılmaktadır. Şablon modelleri tek bir görünüşten modellenir. Dolayısıyla sadece bakış acısının değişmediği veya çok az ve yavaş değiştiği durumlarda kullanılmalıdır.

Histogram modeli yaygın olarak tercih edilen bir başka yaklaşımdır. Temel olarak histogram hesaplama işlemi, tanımlanmış olan niceleme seviyelerine göre, ilgili seviyeye uyan gözlem sayısını sayarak hesaplamaktır. İmge için histogram, renk veya yoğunluk değerleri ile oluşturulur, Fakat aydınlatma, gölge gibi istenmeyen etkileri ortadan kaldırmak için, renk ayarı gibi ön işlemler ile bu etkilerin azaltılması gerekebilir [35].

Renk değerlerinin kullanımına alternatif olarak, imgedeki değişimler de tanımlayıcı oluşturmak için kullanılabilir. Bu bağlamda SIFT ve HOG kullanılan iki yakın yaklaşımdır. Bu iki yaklaşımda da imge üzerindeki yoğunluk değişimleri hesaplanır ve değişim büyüklüğüne bağlı olarak ağırlıklandırılmış yönlü değişim histogramı hesaplanmaktadır. SIFT yönteminde ayrıca, ortaya çıkan histogramdan yönelim miktarı hesaplanır ve döndürmeye karşı değişmezlik için hesaplanan yönelimden çıkarılır. SIFT yöntemi ile diğer önemli nokta algoritmalara göre daha fazla özellik bulmaktadır. Bunun nedeni, özellik noktalarının farklı ölçeklerde ve farklı çözünürlüklerde hesaplanmasıdır. Böylece SIFT imge bozulmalarını daha dayanıklı olması sağlanmıştır [11]. SIFT yöntemi gri seviye imgeler üzerinde çalışmaktadır. Son yıllarda renk tabanlı SIFT yöntemleri de önerilmiştir [53]. Şekil 2.7.'de örnek bir imge için HOG ve SIFT özellikleri verilmiştir. Görüldüğü gibi HOG sadece imge yoğunluk değişimlerinde anlamlı bilgi verirken, SIFT farklı noktalar da özellikler bulabilmektedir.

(a) (b) (c)

Şekil 2.7. Örnek bir imge için HOG (b) ve SIFT (c) ile bulunan özellikler

Verilen bir B bölgesi için, $I(x, y, i)$ piksel değerleri olarak varsayalım. $i = 1, \dots, d$ pikselin özellikleri, koordinatları, yoğunluk değeri, dokusu ve eğimi, gibi pikselle ilgili herhangi bir şey olabilir. K bölge koveryans tanımı Denklem 2.8'de verilmiştir ve dxd koveryans matrisi ile B bölgesinini piksel özelliklerini temsil eder [35].

$$K = \begin{bmatrix} k_{11} & \dots & k_{1d} \\ \vdots & \ddots & \vdots \\ k_{d1} & \dots & k_{dd} \end{bmatrix} \quad ve \quad k_{ij} = \frac{1}{N_B - 1} \sum_{n=1}^{N_B} (I(x_n, y_n, i) - \mu_i)(I(x_n, y_n, i) - \mu_j) \qquad (2.8)$$

Burada μ bölgedeki tüm N_B pikselleri için karşılık gelen özelliğin ortalamasıdır.

Koveryans matrisleri birçok özelliği bir araya getirdiği için bölge tanımlayıcısı olarak kullanmak büyük avantaj sağlamaktadır. Bölgeden hesaplanan koveryans matrisi ile genellikle bölgeyi farklı pozlar ve görünümlerle eşleştirmek için yeterlidir. Bölge dağılım koveryansının hesaplanması, bölgeyi diğer dağılımlardan ayırt etmek için yeterlidir.

Topluluk tanımlayıcıları, zayıf veya parça tanımlayıcılardan oluşur. Bu nedenle bir uzmanlar topluluğu olarak görülebilir. Her uzman çözülmesi gereken sorunu farklı bir bakış acısı ve bağımsız bir şekilde ele alır. Tabii tüm bu bağımsız görüşlerin kurallar ile bir araya getirilmesi ve bir sonuca varılması gereklidir [54].

Topluluk tanımlayıcılarına nesne izleme uygulamalarına bakacak olursak, Avidan tarafından [55] 2007 yılında geliştirilen izleme yönteminde, nesneyi arka plandan ayırmak için sürekli olarak zayıf sınıflandırıcı kümesi güncellenmektedir. Sınıflandırıcılar herhangi bir zamanda eklenip çıkarılarak nesne ve arka plan değişimi takip edilmektedir. Bundan dolayı nesne direk olarak temsil edilmek yerine pikselin nesne olup olmadığı belirlenmektedir. Her zayıf sınıflandırıcı doğru ve yanlış veriler ile eğitilerek hesaplanır.

Doğru veriler nesnenin örneklerinden gelirken yanlış veriler arka plan örneklerinden gelmektedir. Bir sonraki video çerçevesinde piksel güven haritasını oluşturmak için güçlü sınıflandırıcılar kullanılır. Haritada zirve noktası nesnenin yeri olarak varsayılır. O anki video çerçevesinde nesne tespit işlemi tamamlandıktan sonra, yeni video çerçevesi için yeni zayıf sınıflandırıcılar eğitilir ve topluluğa eklenir. Bu işlem her yeni video çerçevesinde tekrarlanır. Bai ve diğ. [56] tarafından ise rastgele topluluk tanımlayıcı nesne izleme amaçlı kullanılmıştır. Yapılan çalışmada nesnenin konumu hızı gibi nesne durumu değil, sınıflandırıcıların konumunu tahmin etmeye odaklanılmıştır. Yapılan çalışma önerilen yöntemler aşağıda verilmiştir [56].

- Sınıflandırıcı ağırlıklarının durağan olmayan dağılımını tahmin etmek için Bayes tahmin teorisini kullanan bir sınıflandırıcı topluluğu önerilmiştir.
- Rastgele sınıflandırıcı zayıf sınıflandırıcılar arasından diğerlerine göre daha güvenilir olan sınıflandırıcıyı seçmeye çalışır. Yani hangi nesne özelliklerinin daha ayırt edici olduğunu olasılıksal bir öneri yapmaktadır.
- Sonuç olarak önerilen topluluk izleyicisi nesnenin en ayırt edici bölgesini bulurken, kısmı kapalılığa karşı da dayanıklı olmasını sağlar.

Öz uzay yaklaşımı ise bir dizi imge verildiğinde, eğitim kümesindeki değişimlerin çoğunu karakterize eder ve eğitim imgelerinden herhangi birine yaklaşmak için kullanılabilecek bir görüntü kümesi oluşturur. Eğitim kümesindeki her p imgesi için, imgelerin taranması sonucu bir boyutlu sütun vektörü oluşturulur. Oluşturulan bir boyutlu vektörlerin her biri veri matrisinde bir sütun olarak kaydedilir. Eğitim veri sayısının piksel sayısında daha az olduğu kabul edilerek, veri matrisinin ayrıştırılması için Tekil Değer Ayrıştırma yöntemi kullanılır. Böylece matristeki veriler aşağıdaki şekilde ayrıştırılmış olur.

- Eğitim setindeki temel bileşen yönlerini gösteren ve veri matrisi ile aynı boyutta olan dikgen matris.
- Köşegen boyunca azalan şekilde sıralanmış ve tekilde değerler içerek köşegen matris.
- Veri matrisinin her sütununu genişletirken kullanılacak katsayıları içeren bir dikgen matris.

Öz uzay bir dizi girdi imgesinde öğrenilen bütün bir görünüm tabanlı nesne gösterimi olarak düşünülebilir. Daha önce gözlemlene nesne görüntüsüne, temel vektörlerin doğrusal kombinasyonu ile yaklaşılabilir. Bu öz uzay ile imge arasındaki eşleşme olarak düşünülür [35].

Bir diğer modelleme biçimi, aktif görünüş modelleridir. Aktif görünüş modelleri, nesne biçimini ve görünümünü eş zamanlı olarak modelleyerek üretilir [57]. Genelde nesne şekli bir dizi yer işareti işle tanımlanır. Aktif kontur gibi yer işaretleri, nesne sınırlarında veya alternatif olarak nesne bölgesinin içinde olabilir. Yer işaretlerinin her biri için renk, doku ve eğim gibi bilgiler ile bir görünüş vektörü oluşturulur. Temel bileşen analizindeki gibi aktif görünüş modeli de örnekler kullanılarak hem şekil olarak hem de görünüş olarak eğitilmesi gerekmektedir.

2.2.3. Model Özellikleri

Nesne izleme uygulamalarında seçilen özellik, uygulama performansını büyük ölçüde etkilemektedir. Genellikle nesneler arasında ve arka plan ile nesne arasında ayrın yapacak özelliklerin seçilmesine dikkat edilmektedir. Çoğu nesne izleme uygulamaları önceden seçilen özelilerin bir bileşimini kullanarak sonuç üretmeye çalışır. Aşağıda yaygın olarak kullanılan görsel özellikler verilmiştir:

- **Renk:** Uygulamalarda genel olarak Kımızı Yeşil Mavi (RGB) renk modeli kullanılmaktadır. Fakat RGB modeli algısal olarak tek biçimli olmadığı için insanlar tarafından algılanan renk farklılıklarına tam olarak karşılamamaktadır. RGB modelinin yerine YUV ve LAB renk modelleri algısal olarak tek biçimli, HSV (Hue, Saturation, Value) renk modeli ise yaklaşık olarak tek biçimli bir renk uzayıdır. Fakat bu renk uzayları gürültüye karşı duyarlıdır. Sonuç olarak nesne izleme uygulamaları için hangi renk uzayının etkili sonuç vereceği ile ilgili kesin bir kanıya varılamaz. Tüm renk uzayları da farklı nesne izleme çalışmalarında kullanılmıştır [35].
- **Eğim:** Nesne sınırları genellikle imge üzerinde güçlü değişiklikler oluşturur. Bu değişimler ile kenar eğimleri oluşturulur. Kenar özellikleri ışık değişimine diğer özelliklere göre daha az duyarlıdır. Nesne sınırlarını izleyen uygulamalar genel olarak kenar özelliklerini kullanmaktadır. Canny Kenar bulma algoritması doğruluğu ve sadeliği nedeniyle genel olarak tercih edilmektedir [58].
- **Optik Akış:** Optik akış bir bölgedeki her pikselin hareketini tanımlayan bir yer değiştirme vektörüdür. Ardışık video çerçevelerinin sabit parlaklık olduğu varsayılır ve imge türevi kullanılarak hesaplanır [59, 60]. Optik akış, hareket tabanlı bölümlendirme ve izleme uygulamalarında kullanılmaktadır. Birçok optik akış

algoritması önerilmiştir. Optik akış yöntemlerinin karşılaştırmaları 2010 yılında Sun ve dig. [61] tarafından 2016 yılında Vesdapunt ve Sinha [62] tarafından yapılmıştır.

- **Doku:** Doku bir yüzeyin yoğunluk değişiminin ölçüsüdür. Renk özelliğiyle karşılaştırıldığında, doku özelliğini oluşturmak için bir ön işleme ihtiyaç vardır. Gri seviye eş oluşum matrisi, dalgacık, Gabor filtreleri gibi çeşitli doku tanımlayıcıları bulunmaktadır. Doku özelliği renk özelliğine göre ışık değişimine daha dayanıklıdır. Ayrıca doku özelliğiyle ilgili kapsamlı araştırmalar da yapılmıştır [63, 64].
- **Köşe Noktalar:** Köşe noktaları gerçekleştirimi kolay ve hesaplama gereksinimi az olduğu için en yaygın kullanılan özelliklerdendir. Harris köşe bulma yöntemi [9] dönme değişiminden etkilenmez fakat ölçek değişiminden etkilenir. Harris köşe bulma yöntemi ile imgeye, farklı ölçeklerde Gauss filtre setini uygulamak gibi farklı ölçek uzayı uygulandığında, farklı ölçeklerde de özellik noktası bulunabilir. Böylece imgedeki ölçekten bağımsız özellik noktaları da bulunabilir.

Çoğunlukla, özellikler geliştirilecek uygulamaya bağlı olarak kullanıcı tarafından manuel olarak seçilir. Fakat otomatik özellik seçme işlemi literatürde büyük dikkat çekmiştir. Otomatik özellik seçme yöntemleri iki gruba ayrılır. Bunlar filtre yöntemleri ve sarıcı yöntemlerdir. Filtre yöntemleri, genel bir kritere göre birbiri ile ilişkisiz özellikleri seçmeye çalışır. Sarıcı yöntemler ise özelliği belli bir problem alanında kullanışlı olup olmadığına bağlı olarak özelliği seçmektedir [65]. Temel Bileşen Analizi (TBA) özellik azaltma uygulamasıyla filtre yöntemlerine bir örnek olarak verilebilir. TBA yöntemi birbiri ile ilişkili değişken sayısını azaltarak, birbiri ile ilişkisiz durama getirir. Bu birbiri ile ilişkisiz değişkenler ise temel bileşen olarak adlandırılır.

Sarıcı yöntemlere örnek olarak arttırıcılar verilebilir. Örneğin Adaboost yönteminde zayıf sınıflandırıcıların bir kombinasyonu ile güçlü bir sınıflandırıcı elde edilir. Geniş bir özellik seti göz önüne alınırsa, her bir özellik için bir sınıflandırıcı eğitilir. Adaboost, algoritmasının performansını maksimum yapacak, zayıf özellikler ile elde edilmiş sınıflandırıcıların ağırlıklı bir kombinasyonunu bulur. Bu ağırlıklar o uygulama için, özelliğin ne kadar ayırt edici olduğunun bir ölçüsüdür.

2.3. Nesne İzleme

Nesne izlemede, özel varsayımlar ile sınırlandırılmamış, katı olmayan ve deforme olan, hızlı hareket eden nesnelerin sağlam ve doğru bir şekilde takip edilmesi en önemli

problemdir. Yapılan çalışmalara bakıldığında nesne izleme yöntemlerinde genel olarak nesne hareketlerinin ani değişmediği ve sabit hızda olduğu varsayılmaktadır. Tabi ki bu kısıtların yanı sıra nesnenin sayısı, büyüklüğü, görünümü ve şekli gibi başka kısıtlar da eklenmektedir.

2.3.1. Nokta Tabanlı İzleme Yöntemleri

Nesne izleme işlemi, noktalarla teslim edilen nesnelerin video çerçeveleri boyunca izlenmesine dayanmaktadır. Nokta tabanlı izleme yöntemlerinde iki çerçeve arasındaki noktaların birbiri ile eşleştirilmesi amaçlanmaktadır. Bu işleme benzeştirme problemi denilmektedir ve temel olarak iki farklı yaklaşım bulunmaktadır. Birinci grup yöntemler benzeşme problemini deterministik yöntemler ile çözmektedir. İkinci grup yöntemler ise istatistiksel tabanlı yöntemlerdir.

Nokta tabanlı nesne izleme yöntemlerinin nasıl değerlendirilmesi, doğru noktanın oluşturulup oluşturulmadığına göre yapılmaktadır. Nesne için zemin gerçeği verildiğinde, performans kesinlik ve hassasiyet hesaplamaları ile değerlendirilir. Nokta tabanlı izleme olarak düşünülürse kesinlik ölçütü Denklem 2.9 ile ve hassasiyet ölçütü Denklem 2.10 ile hesaplanabilir [35].

$$Kesinlik = \frac{Doğru\ ilişki\ sayısı}{Kurulan\ ilişki\ sayısı} \tag{2.9}$$

$$Hassasiyet = \frac{Doğru\ ilişki\ sayısı}{Gerçek\ ilişki\ sayısı} \tag{2.10}$$

Gerçek ilişki sayısı zemin gerçeğinde verilen mevcut ilişki sayısını göstermektedir. Ayrıca nesne izleyicileri, yeni nesne giriş veya mevcut nesnenin çıkışı, nesne kapalılığı ve benzeşme probleminin çözümü için kullanılan kaynakların optimizasyonu gibi niteliksel değerlendirmelerin de yapılması gerekmektedir.

a) Deterministik Yöntemler

Benzeşme problemi birden fazla imge üzerinde aynı fiziksel noktanın bulunması olarak tanımlanabilir. Nokta tabanlı izleme yöntemlerinde video çerçeveleri boyunca bu noktaların

benzeşmesi ile nesne konumu tespit edilir ve izlenir. Özellikle nesne kapalılığı, yanlış nesne tespiti ve sahneye nesne giriş çıkışlarında nokta benzeşmesi daha da önem kazanmaktadır.

Literatürde benzeşme problemini deterministik yöntemler ile ele alan nokta tabanlı nesne izleme yöntemlerine bakacak olursak Sethi ve Jain tarafından [66] 1987 yılında yapılan çalışmada nesnenin imge dizisi içerisinde çok yer değiştirmediği ve katı olduğu varsayılmış ve açgözlü yaklaşım uygulanarak özellik noktaları ile nesne takibi uygulanmıştır. Algoritma, iki ardışık çerçeveyi göz önüne alır ve en yakın komşu kriteri temel alınarak başlatılır. Bu algoritmada benzeşim problemi, bir optimizasyon problemi olarak ele alınmakta ve bir görüntü dizisindeki özellik noktalarının yeni yerlerini bulmak için yinelemeli bir algoritma önermektedir. Geliştirilen yöntem, nesne kapalılığı ve nesne giriş çıkışlarını desteklememektedir [66]. 1990 yılında Salari ve Sethi [67] tarafından yapılan çalışma, benzeşim probleminin çözümü için kullanılan optimizasyon işlemlerinde değişiklikler önerilerek Sethi ve Jain tarafından geliştirilen algoritmanın nesne kapalılığı ve nesne giriş çıkışları için çözümler sunulmuştur.

Rangarajan ve Shah [68] tarafından yapılan çalışmada benzeşme problemi, gerçek dünyadaki nesnelerin çoğunun düzgün yollar izlemesi ve yavaş hareket etmesi varsayımı ile çözülmektedir. Önerilen yöntemde ana tema, iki çerçeve arasında optik akış hesaplanmasıdır. Eğer tespit edilen noktalar azalırsa yanlış tespit veya nesne kapalılığı olarak işleme devam edilir. Fakat yöntem nesne giriş çıkışını dikkate almamaktadır. Intille ve dig. tarafından [69] 1997 yılında geliştirilen yöntemde Rangarajan and Shah tarafından geliştirilen yöntemin değiştirilmiş bir versiyonu önerilmiştir. Önerilen yöntemde düzensiz hareket eden ve nesne kesişmelerinin yaygın olduğu durumlarda, birden fazla katı olmayan cisimleri eşzamanlı olarak izleyebilir. Yöntemde kapalı bir dünya ortamı test edilmektedir ve nesne tespiti için arka plan çıkarımı yöntemi kullanılmıştır. 2001 yılında Veenman ve diğ. [70] benzeşme problemi için ortak hareket kısıdı getirerek Sethi ve Jain, ve Rangarajan ve Shah'in çalışmalarını genişletmiştir. Ortak hareket kısıdı, aynı nesnede bulunan noktaların tutarlı bir şekilde izlenmesi için güçlü bir kısıtlama sağlarken, farklı yönlerde hareket eden cisimler üzerindeki özellik noktaları için uygun değildir. Algoritma, iki geçişli bir algoritma kullanarak nokta izlemenin üretilmesiyle başlar. İki video çerçevesi arasındaki maliyet fonksiyonu, Macar atama yöntemi ile minimize edilir. Bu yaklaşım kapalılık ve yanlış nesne tespiti ile başa çıkabilmektedir, fakat sabit nesne sayısı kullanıldığından nesne giriş çıkışına izin vermez. Ayrıca nokta takibi otomatik olarak başlayabilmektedir. Yapılan deneyler ile genişletilmiş yöntemin verimli olduğu, parametrelere bağımlı olmadığı ve diğer

yöntemlerden daha başarılı olduğu gösterilmiştir [70]. Shafique ve Shah [71] 2003 yılında yapılan çalışmada, hız ve konumun zamansal tutarlılığını korumak için çok çerçeveli bir yaklaşım önermektedir. Bu çalışmada benzeşme problemi, graf teorisi olarak formüle edilmiştir. Böylece yöntemde çoklu çerçeveler arasında benzeşmeleri bulmak için, her $P_i = \{x^0, \dots, x^k\}$ noktasının en iyi yolu bulunmaya çalışılır. İlgili çerçevelerdeki bağlantıların eksik oluşması durumunda, yanlış nesne veya nesne kapalılığı durumu ortaya çıkmaktadır. k video çerçeveleri ile oluşturulan yönlü graf, her düğümü (nesne) $+$ ve $-$ olarak iki düğüme bölerek iki parçalı bir graf oluşturulur. Daha sonra noktalar arası eşleştirme, açgözlü yaklaşımı ile sağlanmaktadır. Önerilen algoritmanın açgözlü olması, izleme ve gözetim gibi gerçek zamanlı sistemlerde kullanılmasına izin vermektedir. Ayrıca önerilen algoritma kapalılık, yanlış nesne tespiti ve nesne tespit edememe ile başa çıkabilmektedir. Nokta takibi için geliştirilen açgözlü algoritmaların çoğu, sahneden noktaların girişine ve çıkışına izin vermezken, önerilen algoritma buna izin vermektedir [71].

2012 yılında Ki-Yeol Eom ve dig. [72] tarafından yapılan çalışmada, ardışık çerçevelerde noktaların en muhtemel benzeşimlerini bulmak için bulanık kümeleme kullanan sezgisel bir algoritma önerilmiştir. Önerilen yöntem aşağıda verilen üç aşamadan oluşmaktadır:

- Asama-1: Aday noktaların arama uzayı azaltılır
- Aşama-2: Maliyet tahmini için ikili ilişkilendirme
- Aşama-3: Takip eden çerçeveler boyunca her özellik noktasının ilişkilendirilmesi

İlk aşamada bulanık kümeleme kullanılarak $t-1$ çerçevesindeki tüm noktalar birkaç gruba ayrılır. Bu noktalar arasından benzerlik ölçüsü olarak Öklid uzaklığı kullanılmaktadır. Bir sonraki t çerçevesinde noktalar, bir önceki çerçevedeki küme merkezleri kullanılarak aynı sayıda kümeye bölünür. Ardışık çerçevelerde, sadece aynı grupta olan noktalar arasında ilişki kurulmasına izin verilir. İkinci aşamada, t çerçevesindeki bir nokta ile $t-1$ çerçevesindeki bir nokta arasındaki ilişkinin maliyeti, her noktanın hız vektörüne ve yönelim açısına bağlı olarak tahmin edilir. Son aşamada, ardışık çerçevelerdeki özellik noktaları arasında tüm olası ilişkileri araştırılır. Bu amaçla bir arama ağacı oluşturulmuştur. Arama ağacında son düğümden ilk düğüme doğru giderek optimum çözüm bulunmaktadır. 2013 yılında Saleemi ve Shah [73] tarafından yapılan çalışmada, veri ilişkilendirme maliyetlerinin global olarak en aza indirilmesi probleminden kaçınır ve bunun yerine her nokta izleme için birden fazla nesne merkezli ilişki kurar. Nesne durumunun çoktan çoğa bir ilişki modeli ile temsil edilmesi önerilmiş ve benzeşim problemini izlenebilir kılmak için izlemeler arasında

tespitlerin paylaşılmasını sağlayan yeni sınırlamalar önerilmiştir. Nesne kapalılığını, yanlış tespit ve ayrılmış veya birleşmiş nesne tespiti durumları için ağırlıklı varsayımsal ölçütler önerilmiştir. Ayrıca eşzamanlı olarak hareketli nesne tespiti gerçekleştiren, iki çerçeve farkı alma işlemi sunulmuştur. Önerilen algoritmanın, 1-1 veri ilişkilendirme yöntemlerine göre daha başarılı ve hızlı olduğu vurgulanmıştır [73]. 2017 yılında Xu yang ve dig. [74] tarafından yapılan çalışmada benzeşme problemi, yeni bir üçüncü dereceden graf eşleştirme algoritması ile çözülmesi önerilmiştir. Diğer çok dereceli graf tabanlı eşleştirme yöntemlerine göre önerilen yöntem ciddi derecede hafıza kullanımını azaltmıştır. Ayrıca yöntem hem yönlü hem de yönsüz graf için uygundur. Benzeşme, her grafiğin üçüncü dereceden yapısal bilgisini kodlayan bitişik tensörler arasındaki eşleşmeyle formüle edilmiş ve daha sonra izlenebilir bir matris formuna dönüştürülmüştür. İki farklı gradyan tabanlı optimizasyon yöntemi benzeşme problemini çözmek için genelleştirilerek kullanılmıştır [74].

b) İstatistiksel Yöntemler

Benzeşme problemini istatistiksel yöntemler ile çözen yaklaşımlarda nesne tahmini sırasında ölçüm ve model belirsizlikleri dikkate alınmaktadır. Videoların elde edildiği kameralar her zaman gürültü içerir. Ayrıca nesne hareketi bozulmaya da maruz kalabilmektedir. İstatistiksel yöntemler konum, hız ve ivme gibi nesne özelliklerini modellemek için durum uzayı yaklaşımını kullanmaktadır.

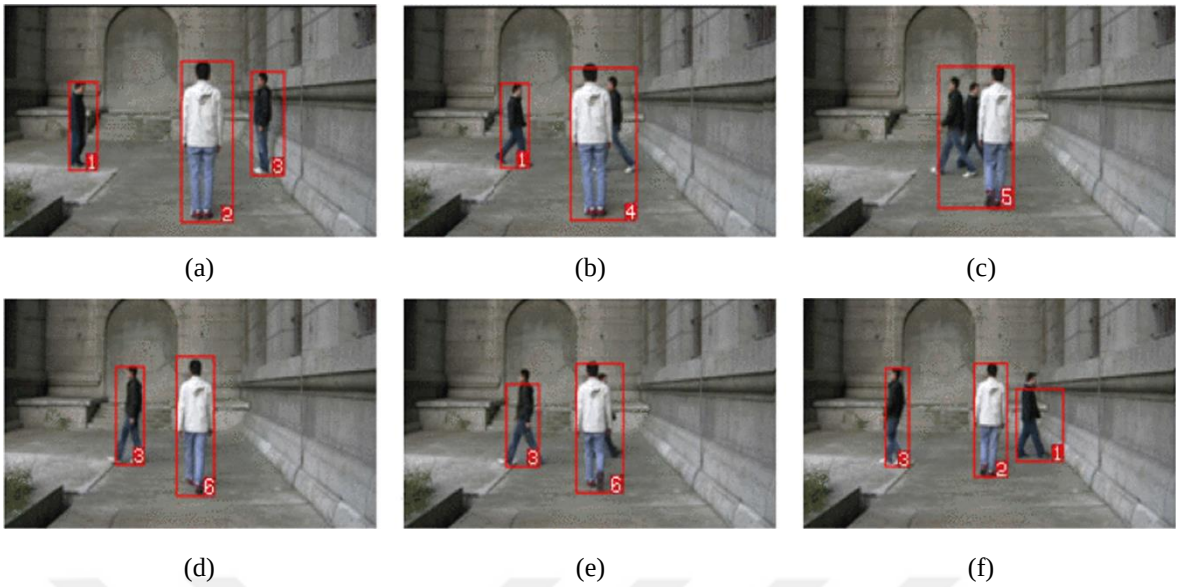
Kalman filtresi [75] istatistiksel yöntemlerle gerçekleştirilen nokta tabanlı izleme yöntemlerine örnek olarak verilebilir. Kalman filtresi lineer sistemin durumunu tahmin etmek için kullanılmaktadır. Kalman filtresinde lineer sistemin durumu Gauss dağılımında olduğu varsayılır. Yöntemde hareketli bir nesnenin konumunu X_t durum dizisi ile tanımlanmaktadır. Durumun zaman içindeki değişimi Denklem 2.11 ile tanımlanmaktadır.

$$X_t = A_t X_{t-1} + n \tag{2.11}$$

Burada n beyaz gürültüdür. İzlemedeki amaç, o ana kadarki tüm ölçümleri verilen X_t durumunu tahmin etmek veya $p(X_t|Y_1, \dots, Y_t)$ olasılık yoğunluk fonksiyonunu oluşturmaktır. Teorik olarak en uygun çözüm, problemi iki aşamada çözen özyinelemeli Bayes filtresi kullanmaktır. Tahmin aşamasında mevcut durum, dinamik bir denklem ve $t -$

1 zamanındaki daha önceden hesaplanan ölçüm kullanarak türetilir. Daha sonra düzeltme aşaması ile o anki ölçümün olasılık dağılımı hesaplanır.

Kalman filtresi, gürültülü imgelerde noktaları izleme ve 2 boyutlu hareketten 3 boyutlu tahmin gibi nesne izleme uygulamalarında yaygın olarak kullanılmıştır [76]. 2010 yılında Xin Li ve dig. [77] tarafından yapılan çalışmada kalman yöntemi ile çoklu nesne takibi önerilmiştir. Önerilen sistem hiçbir manuel giriş gerektirmez ve tamamen otomatik olarak başlamaktadır. Gerçekleştirilen sistemde tek bir sabit kamerayla hareketli nesneler için özellik merkezi ve nesne alanı ile kalman filtre modeli oluşturulmuştur. Yöntem nesnelerin kesişmesi veya ayrılması gibi durumlar ile başarılı bir şekilde başa çıkmaktadır. Şekil 2.8'de kalman filtresi ile gerçekleştirilen uygulamasının çoklu nesne takibi örneği için çalışması gösterilmiştir [77]. 2014 yılında Jong-Min Jeong ve dig. [78] tarafından, kalman filtre tabanlı çoklu nesne izleme yöntemi önerilmiştir. Kalman filtresi sahne içerisindeki hareket eden nesne sayısı kadar kullanılmıştır. Sahne içerisinden birden fazla hareket eden nesne varsa, biden fazla ölçüm elde edilmiş olur. Dolayısıyla çoklu nesne takibi için veri ilişkilendirmesinin de yapılması gerekmektedir. Çoklu nesne takibinde bir başka problem nesne kapalılığı durumudur. Nesne kapalılığı incelenen problemlerden ikisi, nesnelerin birleşmesi ve nesnelerin bölünmesi problemidir. Nesnelerin birleşmesi durumu, birleşmiş nesnelerin birleşmemiş nesneye göre en boy oranının farklı olacağı gerçeği ile ilişkilendirilmiştir. Eğer en boy değişimi belli bir eşik değerini geçerse nesne birleştirme işlemi yapılır. Nesnelerin bölünmesi problemi ise nesnelerin birleşmesi sonucu ortaya çıkmaktadır. Burada ise birleşmiş nesnenin en boy oran tek nesnenin en boy oranına yaklaştığında bölünme işlemi gerçekleşmektedir [78]. 2018 yılında Shengbo Eben Li ve dig. [79] tarafından yapılan çalışmada, karayolu taşıtlarını izlememek için doğrusal ultrasonik sensörler kullanan sistem Kalman filtre tabanlı olarak geliştirilmiştir. Önerilen sistemin iki amacı vardır. Birinci amaç, trafik ortamı için, trafik işaretlerini ve yol durumunu doğru ve güvenilir bir şekilde tanıyacak ultrasonik sensör modeli geliştirmektir. İkinci amaç ise ultrasonik sensörler kullanacak bir nesne izleme algoritmaları tasarlanmasıdır [79].



(a) (b) (c)

(d) (e) (f)

Şekil 2.8. Xin Li ve dig. tarafından yapılan yöntem ile dış ortamda üç insanın izlenmesi [77].

Kalman filtresi durum değişkenini tahmin ederken Gauss dağılımını kullanır. Eğer durum değişkeni gauss dağılımına uymuyorsa, kalman filtresi zayıf sonuç verecektir. Parçaçık filtresi ile bu durumla başa çıkılabilir [80]. Parçacık filtresinde, t zamanındaki $p(X_t|Z_t)$ şartlı durum yoğunluğu $\{s_t^{(n)}: n = 1, \dots, N\}$ örnek kümesi ve $\pi_t^{(n)}$ ağılıkları ile temsil edilir. Burada $s_t^{(n)}$ parçacık olarak ve $\pi_t^{(n)}$ ise örnekleme olasılığı olarak ifade edilir. Ağırlıklar, dağılımın önemini yani gözlem sıklığını tanımlamaktadır [81]. Hesaplama karmaşıklığını azaltmak için her bir $s^{(n)}, \pi^{(n)}$ bağlantısında $c^{(N)} = 1$ olacak şekilde $c^{(n)}$ ağırlıkları kaydedilmektedir. t zamanındaki yeni örnekler, $t-1$ zamanındaki farklı örnekleme şemalarına göre Denklem 2.12 kullanılarak alınmıştır [82].

$$S_{t-1} = \left\{\left(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}\right): n = 1, \dots, N\right\} \tag{2.12}$$

En yaygın kullanılan örnekleme şeması aşağıdaki gibi ifade edilebilir:

- Seçme: S_{t-1}:'den rastgele N tane örnek seçelim.
- Tahmin: Seçilen her bir örnek için yeni bir örnek oluştur.
- Düzeltme: Yeni örnekteki ilgili ağırlıklar bir ölçüm fonksiyonu ile hesaplanır.

Yeni oluşturulan S_t örneği ile yeni nesne pozisyonu tahmin edilebilmektedir.

Kalman filtre ve parçacık filtresi her t zamanında tek bir hesaplama yaptığı varsayılmaktadır yani tek bir nesnenin konumu tahmin edilebilir. Birden fazla nesneyi

izlemek, veri ilişkilendirmesi ve durum tahmini sorunlarının ortak çözümünü gerektirir. Literatürde birkaç istatistiksel veri ilişkilendirme tekniği önerilmiştir. Veri ilişkilendirmesi için JPDAF (Joint Probability Data Association Filtering) [83] ve MHT (Multiple Hypothesis Tracking [84] yöntemleri sıkça kullanılan yöntemlerdir. JPDAF yöntemi yeni nesnenin sahneye girmesi veya nesnenin sahneden çıkması durumlarında çalışmamaktadır. Sadece sabit nesne sayısı olduğu durumlarda veri ilişkilendirmesi yapabilmektedir. MHT yöntemi ise bu probleme çözüm önermiştir. MHT yöntemi her bir nesne için her bir video çerçevesinde birkaç ilişki oluşturur. Nesnenin son izi gözlemin süresi boyunca en muhtemel ilişkiler ile oluşturur. Yöntemde sahneye yeni nesne girdiğinde izleme oluşturulur, sahneden nesne çıktığında ise izleme sona erdirilir. Ayrıca nesne kapalılığına karşı dayanıklıdır. MHT yöntemi yinelemeli bir algoritma kullanır ve hem çalışma zamanı bakımından hem de hafıza bakımından üsteldir. Hesaplama kapasitesini düşürmek için Streit and Luginbuhl [85] tarafından olasılıksal MHT yöntemi önerilmiştir.

2.3.2. Çekirdek Tabanlı İzleme Yöntemleri

Çekirdek tabanlı izleme yöntemleri genel olarak, temel bir şekille temsil edilen nesne bölgesinin hareketini bir video çerçeveden diğerine hesaplayarak gerçekleştirilir. Nesne hareketi genellikle öteleme, ölçeklendirme ve afin gibi parametrik hareketlerle veya yoğunluk akışının hesaplanması ile gerçekleştirilir. İzleme yöntemleri görünüm modellemesi, izlenen nesne sayısı ve nesne hareketini tahmin edecek yöntem gibi parametreler ile birbirinden ayrılır. İzleme yöntemleri yoğunluk tabanlı izleme yöntemleri ve makine öğrenmesi tabanlı izleme yöntemleri olarak ikiye ayrılabilir [7].

Bu bölüdeki nesne izleme uygulamalarında nesne temsili için genel olarak temel geometrik şekiller kullanılır. Nesne hareketi genel olarak öteleme, ölçeklendirme ve afin dönüşümleri ile sağlanır. Nesne hareketi ise önceki çerçevedeki nesne ile o anki çerçeve arasındaki benzerliği maksimum yaparak bulunur. Temel geometrik şekillerin kullanımının sıkıntısı, nesnenin belirli bir parçasının belirlenen bölge dışında kalması ve arka planın bölge içinde kalmasıdır. Bu olay nesne görünümü veya nesnenin şekli değiştiğinde ortaya çıkabilir. Bu durumda model benzerliğinin en üst seviyeye çıkarılarak tahmin edilen nesne hareketi doğru olmayabilir. Bu problemle başa çıkmak için nesnenin tamamının bölge içinde kalmasını sağlamak yerine sadece nesne merkezinin bölge içinde kalması sağlayan bir yaklaşım geliştirilebilir.

a) Yoğunluk Tabanlı İzleme Yöntemleri

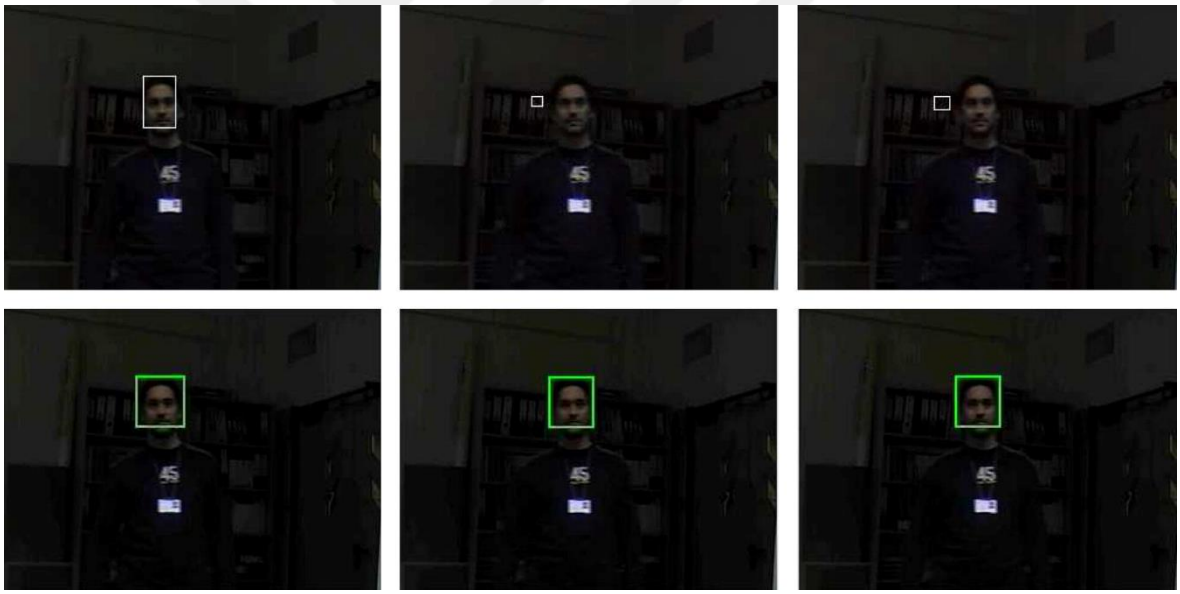
Bu kategorideki en yaygın kullanılan yöntem şablon(blob) eşleştirme yöntemidir. Şablon eşleştirme yöntemi, bir önceki video çerçevesinde tanımlanan nesne şablonuna en uyan bölgeyi bulmak için kaba kuvvet arama algoritmasını kullanmaktadır. Genel olarak şablon imgenin renk değerleri ve yoğunluk değerleri ile oluşturulur. Yoğunluk değerleri ışıklandırmaya karşı dayanıksız olduğundan imge eğim değerleri de şablon için kullanılabilir. Ayrıca histogram ve gauss modeli gibi diğer nesne tanımlayıcıları da eşleştirme için kullanılabilir. Bu işlemde dikdörtgen veya elips şeklinde ifade edilen nesne bölgesindeki pikseller ile hesaplama yapılmaktadır. Şablon eşleştirme yöntemlerinin zayıf tarafı ise gereksinim duyduğu yüksek hesaplama kapasitesidir. Bu yüksek maliyeti düşürmek için genellikle önceki video çerçevesindeki nesne konumu ve çevresinde arama yapılmaktadır.

Çekirdek tabanlı izleme yöntemlerinden ortalama kayma yöntemi [86] nesnenin geçerli bölgedeki renk histogramına en çok benzeyen bölgeyi tahmin etmek için kullanılan yoğunluk tabanlı bir yöntemdir. Ortalama kayma yöntemi, nesnesinin histogramı ile aday nesne bölgesinin histogramını yinelemeli bir şekilde karşılaştırarak görünüm benzerliğini en üst düzeye çıkarır. Nesne ve aday nesne bölgeleri arasındaki histogram benzerliği Bhattacharya mesafesi cinsinden tanımlanır ve yöntemin amacı mesafeyi en aza indirmektedir. Her bir yinelemede aday bölge Bhattacharya mesafesini azaltacak yönde kaydırılır. Yineleme işlemi mesafe azaldığı sürece devam eder. Bu işlemde genellikle 5 veya 6 yinelemede yeni nesne bölgesi bulunur. Ortama kayma yöntemi ile çekirdek tabanlı izleme yöntemlerindeki yüksek hesaplama gerektiren kaba kuvvet algoritması yoktur. Fakat nesnenin bir parçasının, bir önceki çerçeve bölgesi içinde olması gerekmektedir. Bu engele karşılık ek geliştirilmeler önerilmesine rağmen [87], ortalama kayma algoritmasının başarısı büyük ölçüde histogramın ayırt ediciliğine bağlıdır. Ayrıca ortalama kayma algoritması Huiyu Zhou ve dig. [88] tarafından 2009 yılında SIFT algoritması ile birleştirilerek başarımı arttırılmıştır. SIFT yöntemi video çerçeveleri arasındaki benzerliği bulmak için kullanırken, ortalama kayma yöntemi renk histogramı ile benzerlik araması yapmaktadır. Bu iki ölçüm sonucu nesneye en benzer bölgeyi bulmak için kullanılır. Ayrıca iki yöntemden her hangi biri dengesiz sonuçlar verdiğinde diğer yöntem izlemenin devam etmesine olanak sağlar. Algoritmanın tüm çalışma mekanizması aşağıda verilmiştir:

1. Videoda ilk çerçeveye izlenecek bölgeyi temsil edecek bir dikdörtgen tanımlayın.

2. Bu bölge için renk histogramı tanımlanır ve SIFT özellikleri çıkarılır.
3. İkinci video çerçevesinde, ilk konumdan başlanarak ortalama kayma algoritması ile benzerlik ölçüsü hesaplanır. Ayrıca yeni hesaplanan SIFT özellikleri ile ilk video çerçevesindeki SIFT özellikleri arasında karesel fark yöntemi hesaplanır.
4. SIFT ve ortalama kayma yöntemleri ile elde edilen ölçümler arasındaki farkı en aza indirecek bölge aranır.
5. Aradaki fark belirlenen eşik değerinden küçük olana kadar yukarıdaki adımlar tekrarlanır.

Ayrıca Şekil 2.9'da örnek bir video serisi için ortalama kayma ve Huiyu Zhou ve dig. tarafından önerilen algoritmanın karşılaştırmalı sonuçları verilmiştir. Şeklin ilk satırında ortalama kayma ile nesne takibi sonuçları verilirken, ikinci satırda önerilen algoritmanın sonuçları verilmiştir [88].



Şekil 2.9. Ortalama kayma ve Huiyu Zhou ve dig. tarafından geliştirilen algoritmalarının nesne izleme sonuçlarının karşılaştırması. İlk satırda ortalama kayma, ikinci satırda önerilen algoritma verilmiştir [88].

Ayrıca ortalama kayma algoritmasında histogramları karşılaştırmak için Bhattacharyya yöntemini yanı sıra Euclidean ve EMD (The Earth Mover's Distance) gibi farklı çözümler de önerilmiştir [89]. Ido Leichter [90] tarafından 2012 yılında yapılan çalışmada, ortalama kayma yöntemine çapraz kutu ölçütleri kullanarak gelişim sağlamıştır. Yapılan çalışmada 1 boyutlu özellik histogramları arasındaki uzaklık ölçüsü için EMD ölçütü kullanılmıştır. Çok boyutlu özellik histogramları için ise çaprak kutu ölçütlerini kullanmıştır [90].

Temel geometrik şekillerle tanımlanmış bir alanı izlemek için kullanılan bir başka yaklaşım, optik akış kullanılarak bölgenin hareketinin hesaplanmasıdır. Optik akış ile her pikselin yoğun akış alanı sabit bir ışıklandırmada altında Denklem 2.13 yardımıyla hesaplanır [59].

$$I(x, y, t) - I(x + dx, y + dy, t + dt) = 0 \tag{2.13}$$

Bu hesaplama her zaman pikselin komşuları ile geometrik [91] veya cebirsel [60] olarak gerçekleştirilir. Dikdörtgen bir bölgenin hareketi için optik akış yöntemini genişletmek oldukça kolay bir işlemdir. 1994 yılında Shi ve Tomasi [10] tarafından yapılan çalışmada bir bölgenin (d_u, d_v) hareketini hesaplamak için ayırt edici noktaların merkezde olduğu KLT izleyicisini önermiştir (Denklem 2.14).

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} d_u \\ d_v \end{bmatrix} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{2.14}$$

Denklem 2.14 Lukas ve Kanade [60] tarafından önerilen optik akış yöntemine benzemektedir. Yeni ayırt edici noktanın (bulunan özellik noktası) bulunması ile birlikte, KLT izleyicisi ardışık çerçevelerdeki karşılık genel bölgeler arasında afin dönüşümü uygulayarak izlenen bölgenin kalitesini değerlendirir. Eğer geçerli bölge ile bulunan bölge arasındaki karesel fark toplamı küçükse izlemeye devam edilir aksi takdirde yeni özellik ortadan kaldırılır.

Optik akış ve KLT izleme yöntemleri literatürde nesne izleme algoritmalarında şıkça kullanılmaktadır. 2005 yılında Jeongho Shin [92] tarafından yapılan çalışmada optik akış kullanılarak özellik tabanlı nesne izleme algoritması sunulmuştur. Önerilen yöntem 3 aşamadan oluşmaktadır. Birinci aşama ilgili nesnenin yerini tespit etme, ikinci aşama uzaysal ve mekânsal bilgileri kullanarak nesnenin konumunun tahmin edilmesi ve düzeltilmesi, son aşamada ise nesne özelliklerini kullanarak nesne kapalılığının olup olmadığının belirlenmesidir. Önerilen yöntem hem katı hem de deforme olabilen nesneleri izlerken, nesnenin ani hareketlerine karşı da dayanıklıdır. Çünkü nesnenin hem özellikleri hem de hareket yönü takip edilmektedir [92]. 2008 yılında Christopher Zach [93] tarafından KLT izleme yöntemi grafik işlemci birimi kullanılarak hızlandırılmıştır. Yapılan testlerde 720x576 çözünürlüğünde bir video üzerinde saniyede 200 çerçeve işlenmiştir. Ayrıca işlem

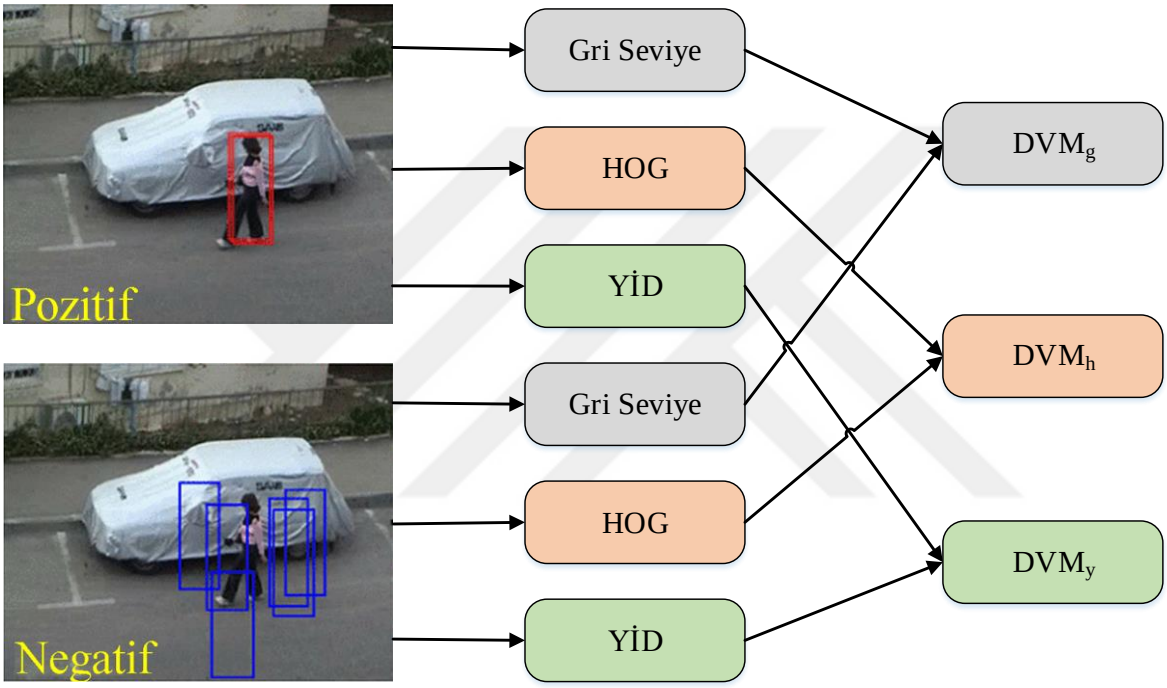
gücü düşük makine ve telefonlarda bile gerçek zamanlı izleme yapılabileceği belirtilmiştir. 2011 yılında Sudipta N. Sinha ve dig. [94] tarafından yapılan çalışmada ise hem KLT yöntemi hem de SIFT yöntemi GPU üzerinde geliştirilmiştir. Böylece gerçek zamanlı bir video izleme sistemi oluşturulmuştur. Yöntemde gerçekleştirilen işlemler paralel çalışacak şekilde tekrar dizayn edilmiş ve CPU versiyonlarına göre büyük derecede hız artışı sağlanmıştır. Gerçekleştirilen GPU tabanlı KLT izleme yöntemi CPU versiyonuna göre 20 kat hızlandırılmıştır. SIFT yöntemi ise CPU versiyonuna göre 10 kat hızlanmıştır. 2013 yılında Sepehr Aslani ve Homayoun Mahdavi-Nasab [95] tarafından yapılan çalışmada sabit kamera kullanılan dijital videolarda hareketli nesnelerin algılanması için bir sistem geliştirilmiştir. Geliştirilen sistemde hareket algılama ve izleme sistemi, optik akış tahminine dayanarak geliştirilmiştir. Ayrıca süreci geliştirilmesi amacıyla diğer görüntü işleme ve bilgisayar görmesi algoritmasında da yararlanılmıştır. Gürültüleri gidermek için ortanca filtresi ve istenmeyen nesnelerden kurtulmak için de morfolojik işlemler ve eşik değer kullanılmıştır. Ayrıca blob analiz kullanılarak nesne türü belirtilmiştir [95].

b) Makine Öğrenmesi Tabanlı İzleme Yöntemleri

Şu ana kadar bahsedilen çekirdek tabanlı izleme yöntemleri görünüş modeli histogram, şablon gibi çevrimiçi olarak elde edilmektedir. Dolayısıyla bu modellerde en son gözlemlerde elde edilen bilgiler ile nesne temsil edilmektedir. Nesne farklı açılardan farklı görünebilir ve nesne görünümü izleme sırasında ciddi şekilde değişirse görünüş modeli artık geçerli olmayabilir ve nesne takip edilemez. Bu sorunun üstesinden gelmek için nesnenin farklı görünümleri çevrimdışı olarak öğrenilebilir ve nesne izleme amaçlı kullanılabilir.

2001 yılında Shai Avidan [96] tarafından yapılan çalışmada nesne izleme uygulaması geliştirilmiştir. Geliştirilen uygulamada DVM sınıflandırma amaçlı kullanılmıştır. DVM'ye pozitif ve negatif örnekler verildiğinde, iki sınıfı birbirinden ayıran en iyi hiper düzlemi bulmaktadır. Nesne takip sisteminde DVM sınıflandırıcıları ve optik akış yöntemi entegre edilmiştir. Sistem ardışık çerçeveler arasındaki farkı en aza indirerek çalışmak yerine, DVM sınıflandırıcısının skorunu en yukarı taşımaya çalışır. Ayrıca görüntü üzerinde nesnenin ani hareketlerini tolare edebilmek için her destek vektörü için destek vektör piramidi şeklinde adlandırılan Gauss piramidi oluşturulmuştur. 2015 yılında Shunli Zhang ve dig. [97] tarafından yapılan çalışmada ise çoklu DVM önerilmiştir. Nesneyi temsil edecek özellikler olarak nesnenin gri seviye, HOG ve Yerel İkili Doku (YİD) değerleri kullanılmıştır. Her bir

DVM görünün modelini oluşturmak için pozitif ve negatif örnekler ile eğitilmektedir. Bu üç özellik nesne için yerel özelliklerdir ve nesneye has özellikler içerir dolayısıyla nesne kapalılığına karşı da dayanıklıdır. Şekil 2.10'da görünüm modeli verilmiştir. Pozitif örnekler nesne çevresinden elde edilirken, negatif örnekler nesneden uzak bölgelerden alınmaktadır. Her örnekteki gri seviye, HOG ve YİD özellikleri çıkarılarak ilgili DVM'ye gönderilir. Ayrıca yapılan çalışmada DVM'lerin ağırlıklarını belirlemek için yeni bir entropi kriteri önerilmiştir [97].



Şekil 2.10. Pozitif ve negatif örnek seçme, özelliklerin çıkarımı ve DVM'lerin eğitimi [97].

Makine öğrenmesi ile nesne izleme uygulamaları sadece DVM ile sınırlı değildir. *k*-en yakın komşuluk, derin öğrenme ve Evrişimsel Sinir Ağı (ESA) gibi diğer makine öğrenme yöntemleri de nesne izlemede şıkça kullanılmaktadır [98, 99].

Derin öğrenme modelleri nesne izlemenin çeşitli aşamalarında kullanılmaktadır. Özellikle derin öğrenmenin yüksek seviye ve anlamlı özellik çıkarımı konusundaki başarısı, derin öğrenmenin nesne izleme uygulamalarında sıklıkla kullanılmasına neden olmuştur. Kullanılan en yaygın yaklaşım ESA yaklaşımıdır. ESA yaklaşımını kullanmak yerine, en ayırt edici özellikleri bulmak için siyam ESA da kullanılabilir [100].

SORT algoritması yaya tespiti için geliştirilen ilk ESA tabanlı çoklu nesne takip uygulamalarındandır [101]. Daha sonra derin öğrenme teknikleri ile geliştirilerek

DeepSORT [102] yöntemi geliştirilmiştir. Yöntemde kalman filtresi ile nesne konumunu tahmin etmek için ve Macar algoritması da ilişkilendirme problemini çözmek için kullanılmıştır. Kalman filtresi etkili bir yöntem olmasına rağmen kapalılık, farklı bakış açıları gibi gerçek dünya senaryolarında başarılı sonuçlar elde edememektedir. Dolayısıyla yöntemde derin öğrenme tabanlı nesne görünümüne bağlı yeni bir ölçüt kullanılmıştır [102].

Bir başka yaklaşımda ise farklı nesne örneklerini en ideal şekilde ayırabilecek özellik kümesini öğrenmek amaçlanmaktadır. Bu amaçla ESA farklı imgeler ile gelen bilgileri birleştiren, kayıp fonksiyonu ile eğitilmektedir. Bu ağlar genel olarak siyam ağları olarak isimlendirilmektedir. Kim ve dig. [103] tarafından geliştirilen yöntemde kontrast kaybı kullanılarak eğitilen bir siyam ağı önerilmiştir.

Birçok çalışmada ESA tarafından çıkarılan özellikler üzerinde bazı mesafe ölçütleri kullanılarak izler ile tespitler arasındaki ilişki hesaplanmaktadır. Mesafe ölçümü yapmak yerine derin öğrenme kullanılarak doğrudan yakınlık derecesi elde edilebilir. Milan ve dig. [104] tarafından önerilen yöntemde tekrarlayan sinir ağı kullanılmıştır. Geliştirilen Bayes filtre algoritması aşağıdaki üç adımda taklit edilmektedir:

- Hareket tahmini: Nesnenin daha önceki durumlarını giriş olarak alan ve öğrenen hareket modeli, nesne tespit işleminden çıkan sonuçları dikkate almadan bir sonraki video çerçevesindeki nesne konumunu tahmin etmektedir.
- Durum tahmininin iyileştirilmesi: Yeni video çerçevesindeki tahminler ve tüm bu tahminlerin hedef nesne ile ilişkilendirilme olasılığını içeren ilişki vektörü, nesne tahmininin iyileştirilmesi için kullanılmaktadır. Böylece gerçek nesne ile tespitler arasında yakınlık ilişkisi kurulabilmektedir.
- İzlemenin oluşturulması ve silinmesi: yeni video çerçevesinde nesnenin var olma olasılığını tahmin etmek için önceden toplanan bilgileri kullanmaktadır.

Bu bölümde bahsedilen nesne izleme yöntemlerinde amaç nesne hareketini tahmin etmektir. Bölgesel tabanlı nesne gösteriminde hesaplanan nesne hareketi, nesnenin o anki konumunu ve bir sonraki çerçevede tahmini nerede olacağı hakkından bilgi verir. Yapılacak olan uygulamanın amacına göre sadece nesne veya sadece nesne hareketi incelenebilir. Örneğin nesne hareket analizi yapılacaksa sadece nesnenin hareketinin izlenmesi yeterli olabilir fakat nesne tanımlama yapılacaksa nesnenin kapsadığı bölge de önem kazanmaktadır.

2.3.3. Siluet Tabanlı İzleme Yöntemleri

Siluet tabanlı bir nesne izleyicinin amacı, önceki çerçeveler kullanılarak oluşturulan nesne modelini ile nesne bölgesini bulmaktır. Siluet izleyiciler iki kategoride sınıflandırılabilir. Bu kategoriler eşleştirme ve kontur gelişimi. Eşleştirme yönteminde, geçerli çerçevedeki nesne siluetini aranır. Kontur gelişimi yöntemlerinde ise, durum uzayı modeli veya bir enerji fonksiyonu minimizasyonu ile konturun yeni konumu bir önceki konum kullanılarak hesaplanır.

Şekil eşleştirme yöntemlerinde nesne silueti o anki video çerçevesinde aranması dolayısıyla şablon eşleştirme yöntemi ile benzemektedir. Arama işlemi aslında, daha önceki çerçevelerden elde edilen bilgilerin sentezlenmesi sonucu oluşturulan nesne modeli ile o anki video çerçevesinde elde edilen nesnenin benzerlik ölçüsüdür. Bu yaklaşımda nesnenin sert bir cisim olduğu varsayılarak sadece video çerçeveleri arasında öteleme işleminin olduğu varsayılmaktadır. Genellikle nesne modeli, bir kenar haritası olarak modellenir ve her video çerçevesinde nesne konumlandırıldıktan sonra görünün değişiklikleri ile başa çıkabilmek için tekrar nesne modelleme işlemi yapılır. Böylece görüş açısı, aydınlatma değişimleri ve esnek şekil modelleri desteklenmektedir. 1993 yılında Huttenlocher ve dig. [105] kenar tabanlı şekil eşleştirme yöntemi geliştirmiştir. Yapılan çalışmada Hausdorff mesafesi kullanılır [106]. Hausdorff mesafesi iki veri kümesi arasındaki mesafe ölçülür. Bu ölçümde bir kümedeki her nokta için, diğer veri kümesindeki her nokta ile karşılaştırılarak en kısa mesafe bulunur. Daha sonra her nokta için hesaplanan en kısa mesafelerden en büyüğü seçilir. Bu seçilen uzaklık Hausdorff mesafesi olarak adlandırılır. Dolayısıyla kenar tabanlı modellerde Hausdorff ölçüsü birbirine en uzak kenarın ölçümüdür. Bu nedenle bu ölçüm, kenar haritasında nesne hareketini büyük oranda etkilenmeyen kısımlarını vurgulamaktadır. Örneğin yürüyen bir insan düşünülürse, baş ve gövde şekilsel olarak fazla değişmez fakat bacaklar ve kollar yürüme esnasında ani hareketler yapmaktadır. Dolayısıyla kol ve bacaklara karşılık gelen kenarları çıkarmak izleme performansını arttıracaktır.

Şekil eşleştirmede kullanılan birçok farklı yaklaşım geliştirilmiştir. Önerilen yaklaşımlardan biri iki ardışık çerçevede birbiri ile karşılık gelen siluetleri bulmaktır. Bu yaklaşımda şekil eşleştirme, siluetin tamamı kullanılarak gerçekleştirilir. Siluet çıkarımı genellikle arka plan çıkarımı ile sağlanmaktadır. Nesne silueti çıkarıldıktan sonra eşleştirme işlemi yine bir uzaklık ölçüsü kullanılarak yapılır. Nesne modelleri genellikle renk veya kenar histogramı şeklinde, kenar tabanlı bir siluet şeklinde veya her ikisinin karışımı

şeklinde oluşturulabilir. 2004 yılında Kang ve arkadaşları [107] nesne modelleri olarak renk ve kenar histogramlarını kullanılmıştır. Bu yaklaşımda geleneksel histogramın yerine, referans çemberindeki kontrol noktalarını merkez olarak kabul eden eş merkezli farklı yarıçaplı çemberler ile histogram oluşturulmuştur. Referans çemberi, silueti içine alan en küçük çemberdir. Dolayısıyla bu yaklaşım ile konumsal bilgi elde edilmektedir. Elde edilen renk ve kenar histogramı döndürme, öteleme ve ölçeklendirme dönüşümlerine karşı dayanıklı hale getirmektedir.

Ardışık çerçevelerde eşleşme aramak yerine, şekil eşleştirme işlemi siluet içindeki tüm pikseller için optik akışın hesaplanması ile de yapılabilir. Siluetin tümünde baskın olan akış, nesne hareketi hakkında bilgi vermektedir [108]. 2001 yılında Li ve dig. [109] tarafından yapılan çalışmada nesne izleme işlemi için, optik akış hesaplanmıştır. Ayrıca Hausdorff mesafesi poz tahmini ve konum doğrulama amaçlı kullanılmıştır. Nesne izleme işlemi için, siluet içinin ortalama optik akışını hesaplanır ve bu hesaplanan değerin yeni nesne konumunu vereceği varsayılmaktadır. Nesne konum doğrulama işleminde ise tüm olası nesne pozisyonları kenar tabanlı olarak tutulmaktadır ve bu bilgiler nesne konumunun doğruluğu için kullanılır [109].

Kontur tabanlı izleme yöntemlerinde, önceki çerçevedeki kontur geçerli çerçevedeki yeni konumuna yinelemeli olarak geliştirilir. Kontur gelişiminde mevcut çerçevedeki nesnenin bir kısmının önceki karedeki nesne ile kesişmesi gerekmektedir. Kontur gelişiminde iki farklı yaklaşım bulunmaktadır. İlk yaklaşım, kontur şeklini ve hareketini modellemek için durum uzayı modellerini kullanır. İkinci yaklaşımda ise, kontur enerjisini en aza indirerek doğrudan konturu geliştirmeye odaklanılmaktadır.

Durum uzayı modelinde nesnenin durumu, konturun şekli ve hareket parametreleri ile tanımlanır. Nesne durumu her seferinde bir önceki çerçevedeki nesne ile güncel çerçevedeki nesnenin benzerliğine bağlıdır. Bu benzerlik konturun kenar bilgisine uzaklığı ile ölçülür. Terzopoulos ve Szeliski [110] nesne durumunu dinamik kontrol noktalarının ile tanımlamaktadır. Kontrol noktalarının yeni durumu Kalman filtresi yardımıyla hesaplanmaktadır. Isard ve Blake [111] nesne durumunu eğri parametreleri ve hareket parametreleri ile tanımlamıştır. Bu yöntemde nesne durumu parçacık filtresi kullanılarak güncellenir. Filtredeki ilk sonuçlar, eğitim aşamasında hesaplanan konturun durum değişkenleri ile hesaplanmaktadır. Test aşamasında mevcut durum değişkenleri, kontur noktalarında yapılan kenar gözlemlerine dayanan parçacık filtresi kullanılarak tahmin edilir. MacCormick ve Blake [112] çalışmayı geliştirerek çoklu nesne izleme ve nesne kapalılığı

sorunlarına çözüm getirmiştir. Chen ve dig. [113] konturu elips şeklinde ifade edilmiştir. Bu yaklaşımda her kontur noktası gizli Markov modelleri ile temsil edilmiştir.

İkinci yaklaşım olan kontur gelişimi yönteminde hem bölütleme hem de nesne izleme yapılabilmektedir. Her iki yaklaşımda da kontur enerjisinin minimizasyonu çeşitli algoritmalar ile yapılmaktadır. Kontur tabanlı izleme yöntemlerinde optik akış sıkça kullanılmaktadır. Mansouri tarafından [114] yapılan çalışmada optik akışın kısıdı kullanılmıştır. Optik akış kısıdı sabit ışıklandırma sınırlamasında Denklem 2.15 ile elde edilmektedir [7].

$$I^{t+1}(x,y) - I^{t}(x-u, y-v) = 0 \tag{2.15}$$

Burada I imgeyi, t zamanı ve (x,y) ise x ve y yönlerinde akış vektörünü ifade etmektedir. 2000 yılında Bertalmio ve diğ. [115] tarafından yapılan çalışmada, konturu ardışık çerçevelerde geliştirmek için bu kısıtlamayı kullanır. Bu çalışmada amaç, seviye kümesi yöntemi kullanarak her bir kontur noktasının konumu için u ve v değerlerini hesaplamaktır. Bertalmio ve diğ. tarafından yapılan çalışmada optik akış sadece nesne sınırlarında hesaplanmaktadır. Mansouri tarafından yapılan çalışmada ise optik akış hesaplaması, nesne bölgesi içinde kalan tüm piksellerde akış vektörünün kaba kuvvet arama yöntemi kullanılması ile elde edilmektedir. Parlaklık kısıtına dayanan akış vektörü hesaplandıktan sonra kontur enerjisi güncellenir. Bu işlem enerji minimum olana kadar yenilemeli şekilde devam eder. Optik akış yerine nesne bölgesinin içinde ve dışında hesaplanan istatistiksel bilgileri kullanmaktır. Bu yaklaşımda geçerli çerçevedeki kontur bir önceki çerçevedeki kontur ile başlatılır.

2004 yılında Yilmaz and Shah [116] nesne sınırları boyunca geliştirdikleri renk ve doku tabanlı model ile kontur gelişimini sağlamıştır. Diğer yaklaşımların aksine, geliştirilen yöntemde seviye kümesi tabanlı şekil modeli ile nesne modeli ve değişimleri modellenir. Ayrıca seviye kümesi tabanlı model ile izleme sırasında oluşan nesne kapalılığı sorunu çözülmüştür. Shi ve Karl [117] tarafından yapılan çalışmada kontur eğrisinin gelişimini iki bağlı liste arasında eleman değişimi ile sağlanabileceğini önermektedir. Önerilen yöntemde kontur gelişimi seviye kümesi yöntemi ile sağlanmaktadır. Fakat seviye kümesi yönteminde olan kısmi diferansiyel denklemleri çözmek yerine kontur gelişimi iki bağlı liste arasında eleman değişimi ile sağlanmaktadır. Ayrıca Gauss filtresi kullanılarak eğri üzerinde yumuşatma işlemi uygulanmaktadır. Yöntemin en büyük avantajı kısmi diferansiyel

denklem çözülmediği için hızlı çalışması ve gerçek zamanlı nesne takibine uygun olmasıdır. Seviye kümesi yönteminin en büyük avantajı, çoklu nesne takibi sırasında meydana gelebilecek birleşme ve bölünme gibi topolojik değişiklikleri desteklemektedir. 2010 yılında Dzyubachyk ve dig. [118] tarafından yapılan çalışmada seviye kümesi yönteminin bu avantajı mikroskobik görüntülerde hücre izleme algoritması amaçlı kullanılmıştır.

Siluet izleme bir nesnenin tüm parçaları veya bölümleri izlenmesi gerektiğinde kullanılmaktadır. Bölgesel tabanlı bir izleme olduğu için nesne izleme yöntemleri değerlendirilirken kesinlik ve hassasiyet ölçütleri kullanılabilir. Bulunan nesne ile gerçek nesne bölgesi arasındaki kesişme düşünülürse; kesinlik, kesişen bölgenin bulunan nesne bölgesine oranı ve hassasiyet ise, kesişmenin gerçek nesne bölgesine oranıdır.

Siluet tabanlı izleme yöntemlerinde hangi özelliklerin kullanıldığı, nesne kapalılığının nasıl çözüldüğü ve bir eğitim işleminin gerekip gerekmediği gibi bazı önemli etkenler vardır. Ayrıca bazı yöntemler sadece nesne sınır bölgelerini kullanırken bazı yöntemler ise nesne bölgesinin tamamını kullanır. Genel olarak siluet tabanlı izleme yöntemleri görülüye dayanıklıdır ve en büyük avantajı izleme sırasında nesne şekil değiştirse bile bu şekle uyum sağlanabilmesidir [7].

2.4. Sonuç

Nesne izleme uygulamaları genel olarak, nesne görünümünün çok az değiştiği, nesnenin yavaş hareket ettiği gibi bazı kısıtlamalar ile birlikte ancak kararlı davranabilmektedir. Fakat nesne izleme uygulamaları nesne kapalılığı, devam eden nesne görünüm değişikliği gibi durumları açık bir şekilde desteklemeyebilir.

Nesne kapalılığı üç bölüm olarak düşünülebilir. İlk olarak öz kapalılık nesnenin bir kısmının nesnenin başka kısımlarını kapandığı durumu ifade etmektedir. Özelikle eklemli nesnelerde sıkça karşılaşılan bir durumdur. İkinci kapalılık durumu nesnenler arası kapalılık durumudur. İzlenen iki nesneden birinin diğerini kapadığı durumlarda meydana gelir. Üçüncü durum ise arka plan kapalılığı durumudur. Bu durumda arka plandan bulunan sütun gibi bazı parçalar izlenen nesnenin üzerine gelmesi durumudur. Genel olarak nesneler arası kapalılık söz konusu olduğunda, izleyiciler nesnelerin konum ve görünüm bilgisini kullanarak nesne kapalılığı ile başa çıkmaya çalışmaktadır. Nesnenin kısmen kapalılığı durumunda ise bunun izleyiciler tarafından tespit edilmesi oldukça zordur. Çünkü bu durumda izleyici nesnenin görünümünün değiştiğini düşünebilir. İzleme sırasında tam

kapalılık olduğu durumda ise genel olarak ortak yaklaşım, nesne hareketinin tutarlı olacağının varsayılması ve nesne tekrar görünene kadar tahminde bulunulmasıdır. Örneğin nesne hareketini tahmin etmek için Kalman filtresi kullanılabilir. Ayrıca nesne kapalılığı durumu kameranın üste yerleştirilmesi gibi kamera pozisyon seçimi ile azaltılabilir. Bir başka yaklaşım ise çoklu kamera kullanmaktır. Bu durumda bir kamerada nesne kapalılığı oluştuğunda diğer kamaralar kullanılarak nesne izleme devam ettirilir.

Nesne izleme sırasında nesnenin görünüm değişikliği de bir başka problem olarak karşımıza çıkmaktadır. Bu problemle ile gürültülerin ve diğer nesne görünümlerinin nesne modeline tanıtılması ile başa çıkılabilir. Buna rağmen modelin tamamen dönmesi gibi durumlarda izleyici kötü performans gösterebilir hatta izleme kaybolabilir. Bu sorun ile başa çıkmak için kullanılan yaklaşımlardan biri nesnenin farklı görünümlerinin öğrenilmesi ve gerekli olduğu durumlarda kullanılmasıdır. Ayrıca nesne şekli ve hareketi üzerine genel kısıtlamalar kullanmak nesne şeklinin ve hareketinin çevrimiçi olarak öğrenilmesi nesne izleme performansını oldukça artırmaktadır.

Tek bir kameradan katı olmayan nesne ve nesnelerin eğitimsiz şekilde izlenmesi halen çözülemeyen bir problemdir. Bu problem için geliştirilen yaklaşımlarda genel olarak nesnenin önceki bilgileri kullanılarak çıkarımlar yapılabilir. Bu problem için tez çalışmasında SIFT, Gauss ve DBScan algoritmaları tabanlı bir çözüm önerilmiştir.

3. NESNE EŞLEŞTİRME VE İZLEME UYGULAMALARI İÇİN KULLANILAN ALGORİTMALARIN (SIFT, SURF, GPU-SURF) KARŞILAŞTIRMALI ANALİZİ

Bu bölümde, SIFT, SURF ve GPU tabanlı SURF uygulamaları performans açısından kıyaslanmıştır. GPU-SURF algoritması için NVIDIA CUDA mimarisi kullanılmıştır. Kısım 3.1'de özellik çıkarımı, nesne eşleştirme ve izleme ile ilgili bilgiler verilmiştir. Kısım 3.2'de SIFT, SURF algoritmaları verilmiştir. Ayrıca yeni nesil GPU mimarisi incelenmiştir. Kısım 3.3'te uygulamanın gerçekleştirim detayları ve imge eşleştirme için SIFT, SURF ve GPU-SURF algoritmalarının sonuçları verilmiştir. Kısım 3.4'te ise gerçek zamanlı nesne takip uygulaması için algoritmalar ile elde edilen başarı sonuçları sunulmuştur.

3.1. Giriş

Otomatik, doğru ve sağlam bir özellik çıkarımı ve takibi imge işleme ve imge eşleştirmede önemli bir yere sahiptir. Bu bölümde SIFT, SURF ve GPU tabanlı SURF algoritması ile özellik eşleştirme ve özellik takip uygulaması geliştirilmiştir. Geliştirilen uygulama ile iki farklı alanda yöntemler karşılaştırılmıştır. İlk olarak algoritmalar, bulunan ve eşleştirilen özellik sayısı dikkate alınarak incelenmiştir. İkinci olarak gerçek zamanlı nesne izleme uygulamalarında algoritmaların başarım oranı ve çalışma zamanı karşılaştırılmıştır.

Özellik çıkarımı ve imge eşleştirme bilgisayar görme uygulamalarında sıkça kullanılmaktadır. Günümüzde özellik çıkarımı ve imge eşleştirme uygulamaları nense izleme, görüntü erişim sistemleri, imge çakıştırma gibi birçok alanda etkin bir şekilde kullanılmaktadır. Her ne kadar literatürde özellik çıkarımı ve imge eşleştirme ile ilgili birçok yöntem önerilmesine rağmen, evrensel olarak kabul edilmiş bir yöntem bulunmamaktadır. Genel olarak literatürdeki yöntemler bazı durumlarda bazı başarılı sonuçlar vermesine rağmen bazı özel durumlarda başarılı olamamaktadır. Özellik çıkarımı ve izleme için sıkça kullanılan bazı parametreler ölçeklendirme, kapalılık, yönlendirme ve ortam ışık değişimleridir [119].

Literatürde, ilk özellik çıkarım işlemi 1998 yılında Harris ve Stephens [9] tarafından gerçekleştirilmiştir. Bu algoritma daha sonra "Harris Kenar Bulma Algoritması" olarak

adlandırılmıştır. Harris Kenar Bulma Algoritmasının popülaritesi sayesinde 1990'larda birçok kenar bulma algoritması geliştirilmiştir [120, 121]. 1997 yılında FAST algoritması Trajkovic ve Hedley [122] tarafından sunulmuştur. 2004 yılında ise SIFT algoritması Lowe tarafından yayınlanmıştır [11]. Bu algoritma ile ayırt edici değişmez özelliklerin bulunması hedeflenmiştir. Fakat SIFT algoritması yavaş çalışmaktadır. Bu problemle başa çıkmak amaçlı 2004 yılında Ke ve Sukthankar [123] tarafından PCA-SIFT algoritması önerilmiştir. PCA-SIFT yöntemi hızlı çalışmasına rağmen SIFT algoritmasına göre daha az ayırt edici özellik bulmaktadır. 2008 yılında SURF algoritması Bay ve dig. tarafından sunulmuştur [13]. SURF algoritması, hem SIFT algoritması ile benzer özellik bulma sonuçlarına erişirken hem de PCA-SIFT yöntemi kadar hızlı çalışmaktadır. Daha sonraki yıllarda popülerlik kazanan GPU teknolojileri ile birlikte hem SIFT hem de SURF algoritmalarının GPU versiyonları gerçekleştirilmiştir. Böylece daha da hızlanan algoritmalar, gerçek zamanlı nesne izleme gibi alanlarda da kullanılmaya başlanıştır [124, 125].

Nesne izleme sistemleri de özellik çıkartma algoritmaları gibi bilgisayar görmesi uygulamalarının en önemli konuları arasında yer almaktadır. Nesne izleme sistemleri aslında bazı yöntemlerin bir arada birbirleri ile uyumlu bir şekilde çalışması ile ortaya çıkan sistemlerdir. Bu yöntemlerin etkili ve tutarlı bir bileşimi ile doğruluğu ve güvenilirliği yüksek bir nesne izleme sistemi elde edilebilir. Bu sistemlerde en büyük problem, otomatik nesne tespiti ve sınıflandırılmasıdır.

Nesne tespiti ve nesne izleme video çözümleme ve bilgisayar görmesi uygulamalarında sıkça kullanılmaktadır. Askeri, sosyal güvenlik, eğlence ve ticari gibi birçok alanda nesne tespiti ve nesne izleme uygulamalarına sıkça başvurulmaktadır. Bu konuda asıl zorluk gerçek zamanlı olarak nesnelerin tespiti ve izlenmesidir.

Her nesne izleme uygulaması bir nesne tespit yöntemine gereksinim duymaktadır. Nesne tespiti için iki temel yöntem bulunmaktadır. İlk yöntem tek video çerçevesi kullanmaktır. Bu yöntemde daha önce belirlenen arka plan resmi ile video çerçevesi farkı alınır ve elde edilen fark resmi ile nesne tespiti gerçekleştirilir. İkinci yöntem ise birden fazla video çerçeve bilgisi kullanarak arka planın dinamik olarak değişmesidir. Böylece arka plan değişimleri dinamik olarak yakalanabilmektedir [35].

Nesne tespiti için literatürde birçok yöntem önerilmiştir. Bu yöntemler dört ana gruba ayrılarak incelenecektir:

- **Nokta Detektörleri:** Bu gruptaki yöntemler imge üzerinde ayırt edici nokta bulmak için kullanılmaktadır. Literatürde, SIFT [11], SURF [13] ve KLT detektörler [10] bu

amaçla kullanılan en bilinen yöntemlerdir. Bu algoritmalar genel olarak imge ve nesne eşleştirme amaçlı kullanılmaktadır.

- **Bölütleme:** Bölütleme işlemi bir imge bileşenlerine ayırma işlemidir. Literatürde nesne tespiti için genel olarak kullanılan bölütleme yöntemleri aktif kontur [29, 126], graph-cuts [27] ve ortalama kayma [25] yöntemleridir.
- **Arka Plan Modelleme:** Arka plan modellemede her piksel için arka plan modellenmektedir. Hareket eden görüntü ise arka plandan çıkartılarak bulunur. Kullanılan popüler yöntemler Gauss Karışım Yöntemi [127] ve dinamik arka plan modelleme [128] yöntemidir.
- **Denetimli Öğrenme ile Sınıflandırma:** Denetimli öğrenme mekanizması, birçok nesne örneğinden yararlanarak nesnenin görünümünün öğrenmesi işlemidir. Denetimli öğrenme uygulamalarının en popüler yöntemleri Destek Vektör Makineleri [129], Sinir ağı [130] ve derin öğrenmedir [131].

Çok işlemcili yapıların hızla yaygınlaşması ile beraber nesne tespiti ve nesne izleme yöntemlerinde çokça kullanılmaya başlanıştır. Günümüzde çok işlemcili yapılar sadece akademik camia tarafından değil endüstri tarafından da kullanılmaktadır. Bu yapılar ile bazı problemler ve zorluklar çok daha etkili ve hızlı bir şekilde çözüme kavuşturulabilmektedir. Artık günümüzde işlemciler üzerinde bile birçok işlemi paralel olarak çalıştırmak mümkündür. Çok işlemcili yapılar dediğimizde ilk aklımıza gelen mimari GPU mimarisidir. GPU mimarisi 2007 yılında CUDA mimarisinin geliştirilmesi ile birlikte hızla yaygınlaşmıştır. CUDA ile grafik bilgisine ihtiyaç duyulmadan genel amaçlı programlar tasarlamak mümkündür. GPU içerisinden bulunan işlemcilerin hızları görece yavaştır fakat bu işlemciler paralel bir şekilde aynı işlem için çalıştırıldıklarında ortaya muazzam bir güç çıkmaktadır. İmge ve video işleme ve akışkanlar dinamiği gibi yüksek hesaplama gerektiren uygulamalar GPU tabanlı programlar ile çok daha hızlı bir şekilde çalıştırılabilmektedir. Dolayısıyla, GPU mimarisinin kullanımı gerçek zamanlı nesne tespiti ve nesne izleme uygulamaları için büyük bir fayda sağlayacağı acıktır [6].

3.2. Kullanılan Yöntemler

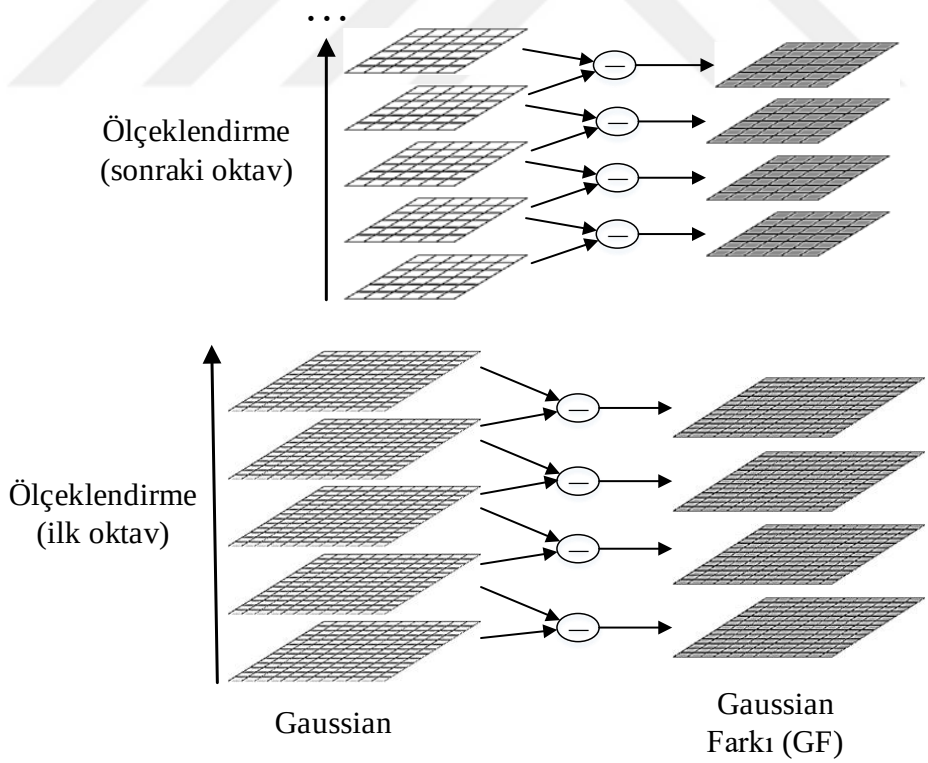
Nesne eşleştirme uygulamaları için iki temel yöntem SIFT ve SURF yöntemleridir. SURF yöntemi SIFT yöntemine göre daha hızlı çalışmasına rağmen, gerçek zamanlı

uygulamalar için yeterli hıza sahip değildir. Dolayısıyla bu bölümde SURF yönteminin GPU versiyonu da kullanılmıştır.

3.2.1. SIFT Algoritması

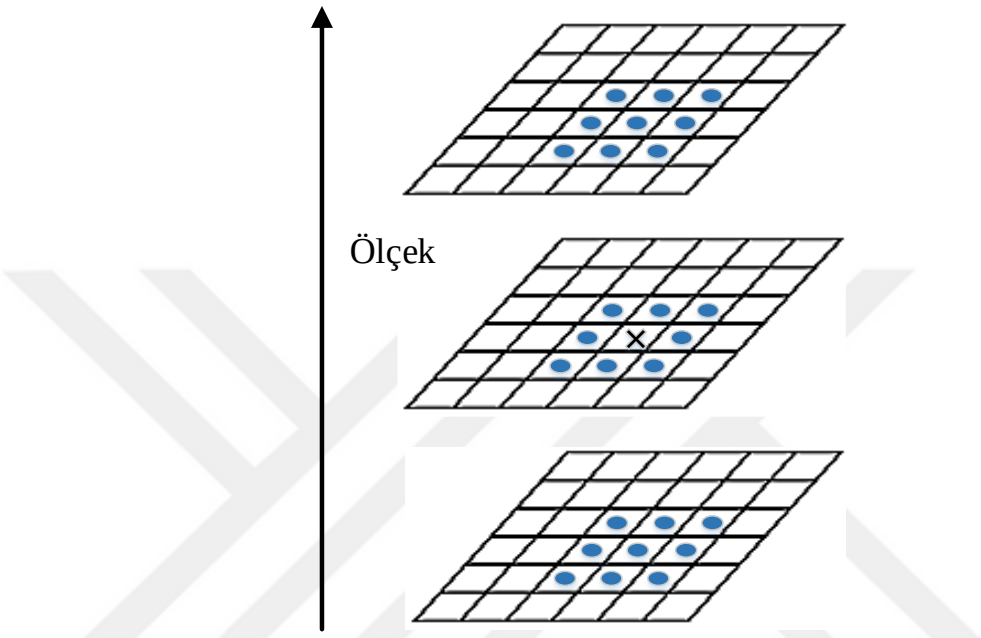
SIFT algoritması, nesne tespiti, nesne izleme ve robotik haritalama gibi birçok bilgisayar görme algoritmalarına başarılı bir şekilde adapte edilebilmektedir. SIFT algoritması ölçeklendirme ve döndürmeye karşı dirençli bir algoritmadır, fakat yüksek hesaplama kapasitesine ihtiyaç duymaktadır. Dolayısıyla gerekli olan bu yüksek hesaplamadan dolayı gerçek zamanlı uygulamalar için uygun bir algoritma olmamaktadır. [119]

SIFT algoritmasında ilgili noktaların tespiti için yapılan ilk işlem konvolüsyon işlemidir. Konvolüsyon işlemi farklı ölçeklerdeki görünüşlerin Gauss filtrelenmiş hali ile komşu imgelerin Gauss Farkı (GF) arasında hesaplanır (Şekil 3.1).



Şekil 3.1. Konvolüsyon işlemi için Gauss Farkı hesaplaması [132].

İlgili nokta keypoint olarak adlandırılır ve GF ölçekleri arasında lokal minimum ve lokal maksimum olarak tanımlanır. GF imgesindeki her bir piksel aynı ölçekteki ve komşu ölçekteki komşuları ile karşılaştırılır. Eğer piksel lokal minimum veya lokal maksimum ise bu nokta aday nokta olarak seçilir. Aday nokta seçim işlemi Şekil 3.2'de verilmiştir.



Şekil 3.2. Lokal minimum ve lokal maksimum hesaplama işlemi [132].

Her bir aday nokta için aşağıdaki 4 adım gerçekleştirilir [11, 133]:

Adım-1: Bu aşamada, ölçek uzayı belirlenmektedir. Ölçek uzayı $L(x, y, \sigma)$ fonksiyonu ile tanımlanır ve imgenin farklı ölçeklerine Gauss filtre işlemi uygulanmaktadır. Gauss filtre fonksiyonu $G(x, y, \sigma)$ ve giriş imgesi $I(x, y)$ konvolsyon işlemine tabi tutulur ve sonuç olarak $L(x, y, \sigma)$ fonksiyonu hesaplanır. Uygulamanın hızını arttırma amaçlı SIFT algoritmasında Gauss işlemi yerine GF işlemi kullanılmıştır. Evrilmiş görüntü $D(x, y, \sigma)$ Denklem 3.1 ile hesaplanır. Bu işlemler sonucunda aday noktalar GF işleminin minimumu ve maksimumu olarak seçilir.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{3.1}$$

Burada k sabit faktör ve $*$ ise konvolüsyon işlemidir.

Adım-2: Bu aşamada kabaca yerleştirilmiş aday noktalar tespit edilerek, noktanın ölçek uzayında nerde olduğuna bağlı olarak konum güncellemesi yapılmaktadır. Ayrıca her bir aday nokta için konum bilgisi komşu piksellerin renk değerlerine göre de güncellenmektedir.

Adım-3: Bu adımda keypoint'ler filtrelenmektedir. GF işlemi sonucunda kenar bölgelerde güçlü değerler elde edilmektedir. SIFT algoritmasının amacı imge eşleştirme olduğu için kenar bölgelerinin işleme dâhil edilmesi elde edilen sonucu bozmaktadır. Dolayısıyla, sonucun sağlam bir şekilde elde edilmesi için kenar bölgelerinden elde edilen adaylar temizlenmektedir.

Adım-4: Keypoint yönelimlerinin hesaplanmasıdır. Yönelimlerin hesaplanması için HOG işlemi uygulanmaktadır. Daha sonra 128 boyutlu normalleştirilmiş vektörden oluşan tanımlayıcı oluşturulur. Algoritmada bu aşamada konum, ölçek ve yönelim özelliklerin bir listesi bulunmaktadır. Böylece bu değerler ile şekil bozulmalarında ve ortam ışığının kısmi değişikliğinde çok yararlı olmaktadır.

SIFT algoritması ile çok büyük sayıda önemli nokta tespit edilebilmektedir. SIFT algoritması çok sayıda özellik noktası bulmasının yanı sıra imge deformasyonlarına da dayanıklıdır. Gerçek zamanlı uygulamalar için SIFT yöntemi biraz yavaş kalmaktadır. Çözüm olarak Bay, H., Tuytelaars, T., ve Van Gool L. tarafından SURF algoritması önerilmiştir. SURF algoritması SIFT algoritmasında etkilenmiş ve nesne tanıma, sınıflandırma ve imge çakıştırmada kullanılmaktadır.

3.2.2. SURF Algoritması

SIFT ve SURF algoritmaları benzer adımlara sahiptir. Fakat her bir aşamanın gerçekleştirim detayları birbirinden farklıdır. SURF algoritması SIFT algoritmasına göre daha etkilidir [134]. SURF yöntemi ile daha fazla imge işlemek de mümkündür. Böylece gerçek zamanlı uygulamalar için daha uygundur. SURF algoritmasında ayırt edici nokta çıkarım işlemi için aşağıdaki adımlar takip edilmektedir [135].

Gri seviye Çevirimi: SURF algoritması 2 boyutlu gri seviye imgelerde çalıştığı için renkli imgeler gri seviyeye çevrilmelidir.

İmge İntegralinin Hesaplanması: İmge integralinin hesaplanmasının amacı gerekli olan kutu filtrenin hesaplanmasına kolaylık sağlamasıdır. İmge integrali piksel yoğunluk değerlerinin kümülatif bir şekilde toplanması ile elde edilmektedir.

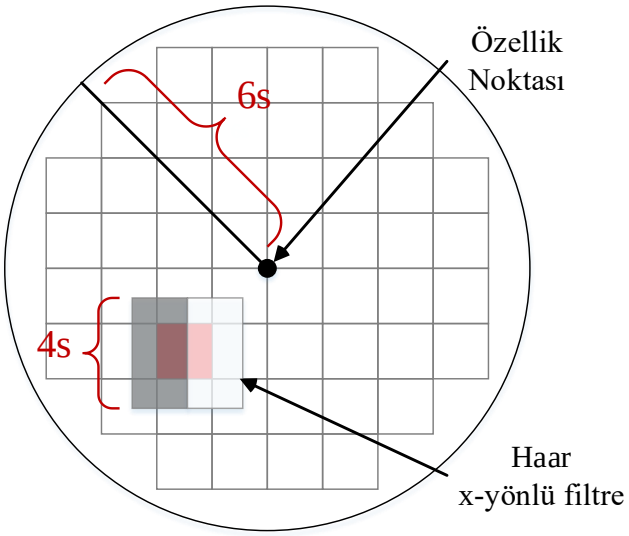
Kutu Filtreleme: Verilen bir piksel değeri için Hessian matrise yaklaşmanın etkili bir yolu kutu filtreleme kullanmaktır. SURF algoritmasında Hessian hesaplaması özellik noktasının bulunması amaçlı kullanılmaktadır.

Ölçek Uzay Üretimi: SURF algoritmasına ölçekten bağımsızlık kazandıran işlemdir. SIFT algoritması ile işlem gerçekleştirilmektedir, fakat SURF algoritmasında hesaplamalar kısmen farklıdır. SIFT yönteminde, Gauss imgeyi elde etmek için, her bir oktava iteratif olarak Gauss filtresi uygulanmaktadır. Daha sonra gauss farkı, iki farklı komşu gauss imgesinin farkının alınması ile elde edilmektedir. SURF yönteminde ise aynı integral imgesine filtreler uygulanır. Bu işlem Gauss filtre oluşturma işleminin çok daha hızlı bir şekilde hesaplanmasını sağlamaktadır.

Özellik Noktası Arama: Arama işlemi aslında tek piksel noktasına odaklanmasına rağmen, pikselin lokal maksimum olup olmadığı tespit etmek için 3x3x3'lük komşuluk kullanılmaktadır. 3x3x3'lük komşuluk ve ölçek uzay gösterimi Şekil 3.2'de verilmiştir. Arama uzayındaki merkez nokta eğer en yüksek yoğunluk değerine sahipse, bu nokta lokal maksimum olarak işaretlenir. Daha sonra, kullanıcı tarafından belirlen eşik değeri ile karşılaştırılır. Eğer merkez nokta eşik değerinden büyükse, bu nokta özellik noktası olarak seçilir.

Yukarıda bahsedilen adımlardan sonra her bir özellik noktası için tanımlayıcı oluşturulmaktadır. SURF yönteminde tanımlayıcıların oluşturulması için aşağıda belirtilen adımlar takip edilmektedir.

Yönlendirme: Bu aşamada amaç her bir özellik noktası için yönlendirme bilgisinin sağlanmasıdır. SURF yönteminde özelliğin yönlendirme bilgisi, özelliği çevreleyen bölgenin hesaplamaya dâhil edilmesi ile hesaplanır. Bu hesaplama sonucunda SURF algoritması döndürmeden bağımsızlık özelliği kazanmış olur. Yön hesabı için özellik noktası merkez alınarak $6s$ yarıçaplı bir çember tanımlanır. Burada s değeri Fast Hessian Determinat değeridir. Ayrıca bu s değeri Haar filtreleme için de kullanılmaktadır. Haar filtreleme kare şeklindedir ve uzunluğu $4s$ boyutundadır. Dahası, özel piksel sonuçları elde etmek amaçlı x ve y yönü için sağ ve sol Haar filtresi de kullanılmaktadır. Haar filtresinin gösterimi Şekil 3.3'te verilmiştir [136].



Şekil 3.3. Haar Filtreleme İşlemi [136].

Tanımlayıcının Hesaplanması: SURF yönteminde, tanımlayıcı özellik noktasını çevreleyen özellikleri tanımlamaktadır. Bu bölge Hessian tabanlı bir algılayıcı ile tespit edilmektedir. Bir sonraki işlemde ise bu bölgeyi tanıtan karakteristiği tanımlamaktır. Bu tanımlama işlemi özellik noktasını çevreleyen bir kare çizmek ve yönelimi göstermek olarak açıklanabilir. Bu kare alan ve özellik noktasının yönelimi Şekil 3.4'te verilmiştir.

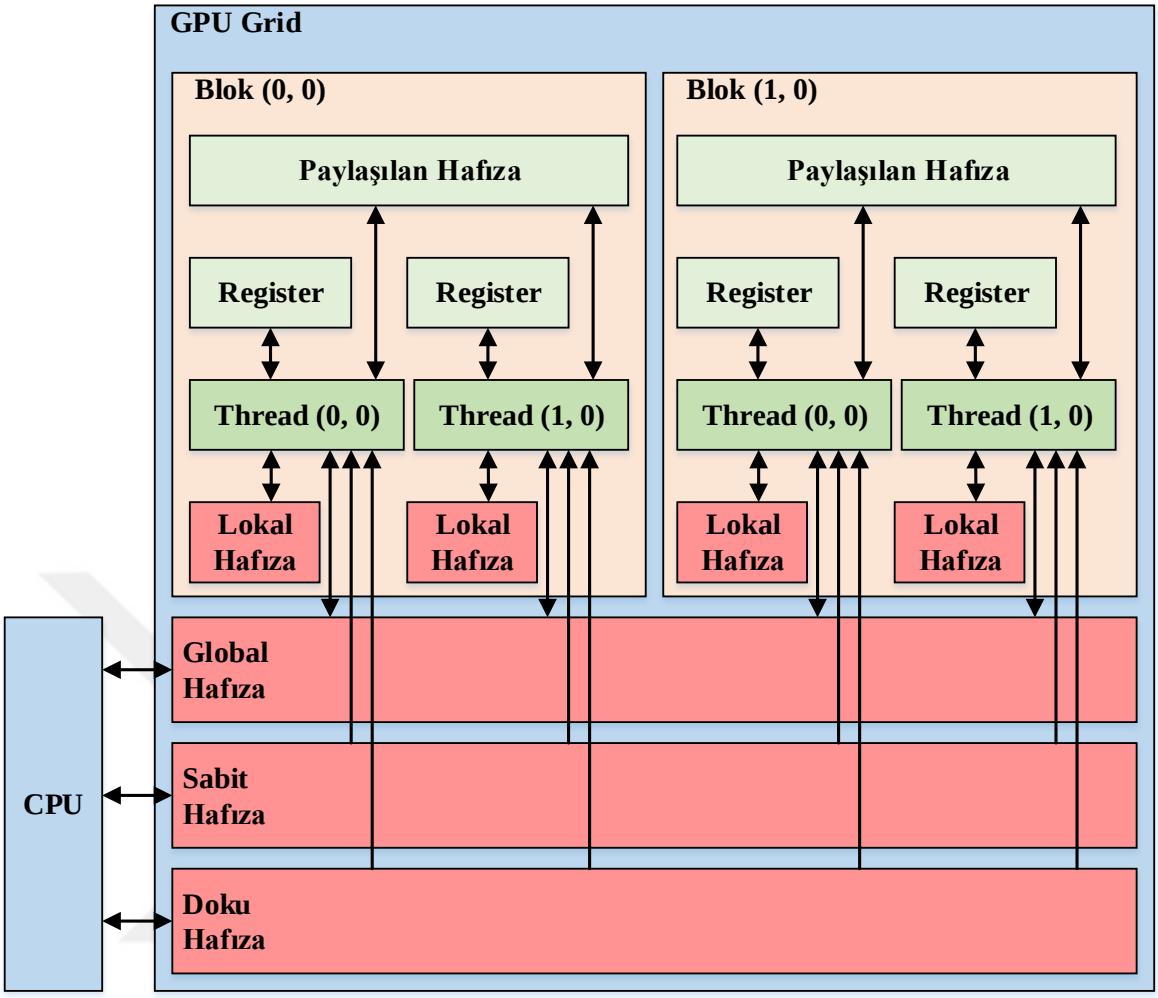
Şekil 3.4. Yönlendirme ve tanımlayıcı bilgileri ile birlikte örnek bir SURF algoritmasının çalışma sonucu [136].

3.2.3. GPU Mimarisi ve GPU-SURF Algoritması

Grafik kartlarının çok hızlı bir şekilde gelişmesi bu kartların genel amaçlı programlar için kullanımını yaygınlaştırmıştır. Grafik kartları uygulama performansını arttırmak amaçlı çok yaygın bir şekilde kullanılmaktadır. NVIDIA firması tarafından geliştirilen CUDA mimarisi uygun bir şekilde kullanımı sonucunda uygulama hızında çok büyük artışlar vadetmektedir. Bu performans artışı aslında paralel hesaplamalar için geliştirilen GPU mimarisinde gizlidir. GPU çok sayıda aritmetik mantık birimi içermektedir fakat önbellek kapasitesi düşüktür. CUDA mimarisi ile birlikte çok sayıda ipliği aynı veri üzerinde eşzamanlı çalıştırmak mümkündür dolayısıyla bu mimari sadece paralel hesaplamalar için uygundur. Örneğin, uygulama içerisinde çok sayıda akış kontrol mekanizması varsa, CUDA uygulama hızını arttıracağına belki de düşürebilir [4, 6].

CUDA C programlama dili, C programlama dilini temel almıştır. Kullanıcının kernel şeklinde isimlendirilen GPU tarafında çalıştırılacak fonksiyonları tanımlamasına izin verir. GPU üzerinde kernel kodu çalıştırıldığında, N tane fonksiyon tanımlanır ve paralel bir şekilde eşzamanlı çalıştırılır. Kernel çalıştırıldığında eşsiz değere sahip threadId ve blockId tanımlanmaktadır [137]. CUDA mimarisinde ızgaraların içinde bloklar, blokların içinde ise iplikler olacak şekilde tasarlanmıştır. Dolayısıyla büyük veri setlerinde threadId ve blockId değişkenleri ile o an hangi ipliğin aktif olduğu kolaylıkla bulunabilir. CUDA mimarisinde birçok hafıza tipi vardır. Şekil 3.5.'te hafıza tipleri ve her hafızanın nasıl erişilebildiği gösterilmiştir. CPU tarafından sadece global, sabit ve doku hafıza tiplerine erişilebilmektedir. Bu nedenle bu üç hafıza tipi CPU ve GPU veri transferi amaçlı kullanılabilmektedir. Lokal ve kayıt hafıza tipleri sadece iplikler tarafından erişilebilen hafıza tipleridir. Paylaşılan hafıza ise bir blok içerisindeki tüm hafıza tipleri tarafından erişilebilirler [3].

Tez çalışmasında OpenCV kütüphanesi bulunan SURF algoritmasının GPU versiyonu kullanılmıştır. GPU-SURF algoritması SURF algoritmasına yaklaşık 3-4 kat hızlı çalışmaktadır. Böylece gerçek zamanlı uygulamalar için gerekli olan işlem kapasitesine ulaşılmıştır.



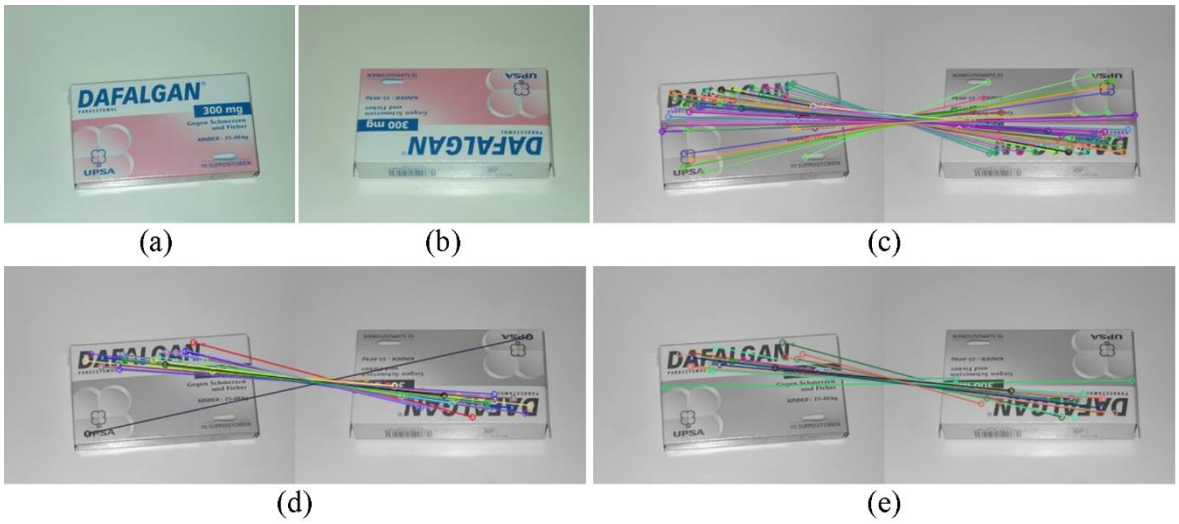
Şekil 3.5. GPU donanım mimarisi ve hafıza tipleri [6].

3.3. Uygulama Sonuçları

Bu bölümde deneysel sonuçlar verilmiştir. Yapılan tüm deneyler Intel Core i7-3770 CPU, 8GB ram bellek ve NVIDIA GeForge GTX 660 Ti ekran kartına sahip bilgisayar üzerinde test edilmiştir. Tez çalışmasında kullanılan GPU toplam 7 grup çok işlemci yapısına sahiptir. Her bir işlemci yapısı 192 CUDA işlemcisi içermektedir. Her çok işlemci 65536 register hafıza birimi içerir ve 32 bank içinde organize edilmiş 48KB paylaşılan hafıza bölgesine sahiptir. Toplam sabit hafıza kapasitesi 64KB olup toplam, global hafıza kapasitesi ise 2GB'dır ve bu hafıza bölgesine GGDR5 arayüzü ile erişilmektedir. Ayrıca GPU mimarisi çift duyarlıklı kayan nokta aritmetiğini desteklemektedir.

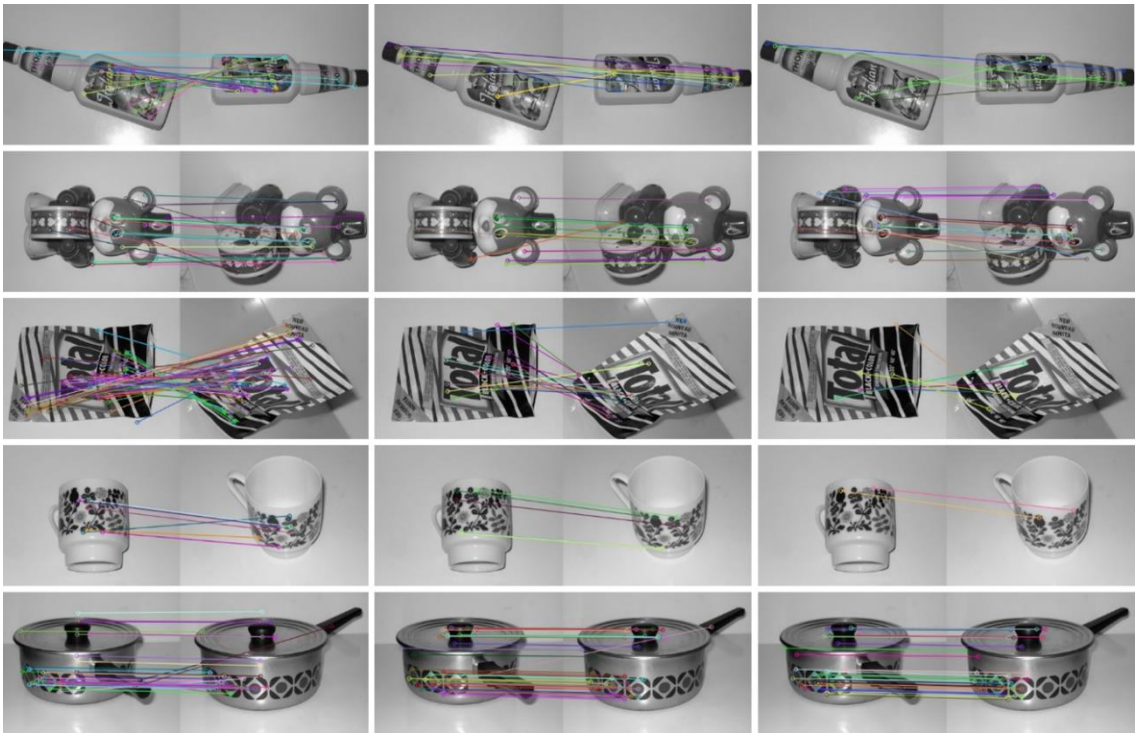
3.3.1. Nesne Eşleştirme için Sonuçlar

Bu bölümde imge eşleştirmek için ETH Zurich (ETHZ) nesne veri seti kullanılmıştır [138]. ETHZ nesne veri seti 53 farklı nesne ve her nesnenin 5 farklı imgesini içermektedir. Toplam 320x240 çözünürlüğünde 265 imge bulunmaktadır. Veri setinde farklı görünümlerde çeşitli nesne imgeleri bulunmaktadır. Geliştirilen uygulamanın performansını test etmek amaçlı, ilk imge temel imge olarak kaydedilmiş ve diğer dört imge test amaçlı kullanılmıştır. Temel imge diğer 4 test imgesi ile teker teker karşılaştırılarak sonuçlar elde edilmiştir. Yukarıda verilen işlemler SIFT, SURF ve GPU-SURF yöntemleri için ayrı ayrı tüm imgeler için tekrarlanmıştır. Şekil 3.6'da SIFT, SURF ve GPU-SURF yöntemleri ile örnek bir eşleştirme gösterilmiştir. Şekil 3.6 (a)'da orijinal imge, (b)'de test imgesi verilmiştir. (c), (d) ve (e)'de ise sırasıyla SIFT, SURF ve GPU-SURF yöntemlerinin özellik eşleştirme sonuçları gösterilmiştir. Şekil 3.6'da görüldüğü gibi SIFT yöntemi ile daha fazla özellik eşleştirmesi elde edilmiştir. Bunun nedeni SIFT algoritmasının imge üzerinde daha fazla özellik bulabilmesidir. SIFT algoritması ile orijinal imgede 216 özellik bulurken, SURF algoritması ile 86 ve GPU-SURF algoritması ile ise 71 özellik bulunabilmiştir. Test imgesinde SIFT, SURF ve GPU-SURF algoritmaları sırasıyla 228, 99 ve 77 özellik bulabilmiştir. Şekil 3.6 (c), (d), ve (e)'de SIFT, SURF ve GPU-SURF algoritmalarının eşleştirilen özellikleri gösterilmiştir. SIFT algoritması ile toplam 95 özellik eşleştirmesi elde edilirken, SURF algoritması ile 40 ve GPU-SURF algoritması ile de 23 özellik eşleştirmesi elde edilebilmiştir.



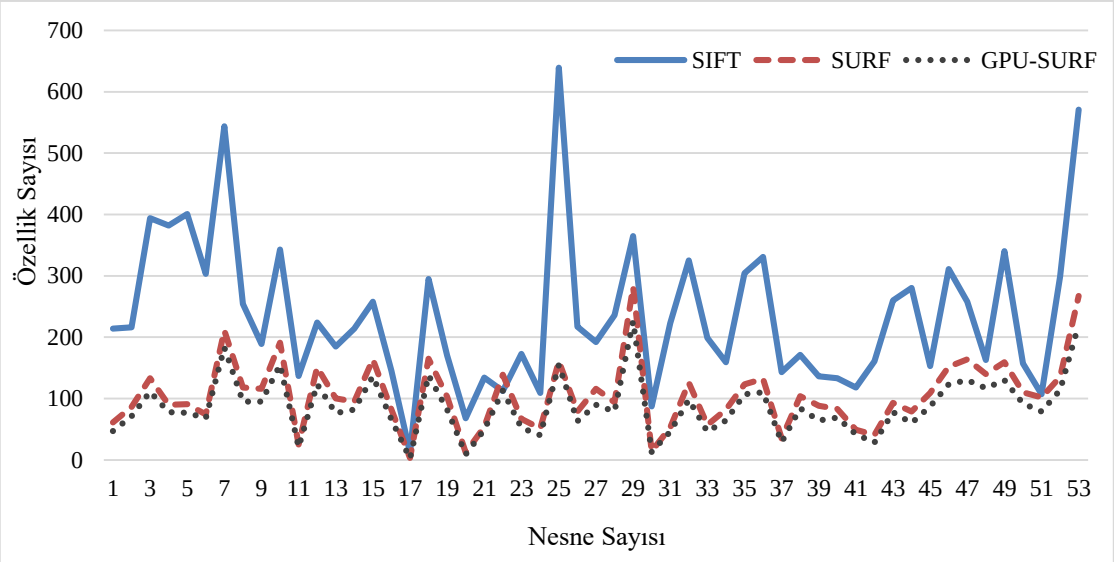
Şekil 3.6. SIFT, SURF ve GPU-SURF algoritmalarının özellik eşleştirme sonuçlarının karşılaştırılması

SIFT, SURF ve GPU-SURF algoritmaları ile bulunan keypoint sayıları ve eşleşen keypoint sayıları hesaplanmıştır. Veri setindeki tüm imgeler hesaplamaya dahil edilmiştir. Veri setinden elde edilen bazı eşleştirme sonuçları Şekil 3.7'de verilmiştir. Şekil 3.7'de ilk sütunda SIFT algoritmasının sonuçları, ikinci sütunda SURF algoritmasının sonuçları ve son sütunda da GPU-SURF algoritmasının sonuçları verilmiştir. Eşleştirme sonuçlarından görüldüğü gibi şeklin açısı, yönelimi değişmesine rağmen algoritmalar genel olarak başarılı eşleştirmeler elde edebilmiştir. Yine sonuçlardan görüldüğü gibi SIFT yöntemi ile diğer yöntemlere göre daha fazla sayıda eşleştirme elde edilmiştir.

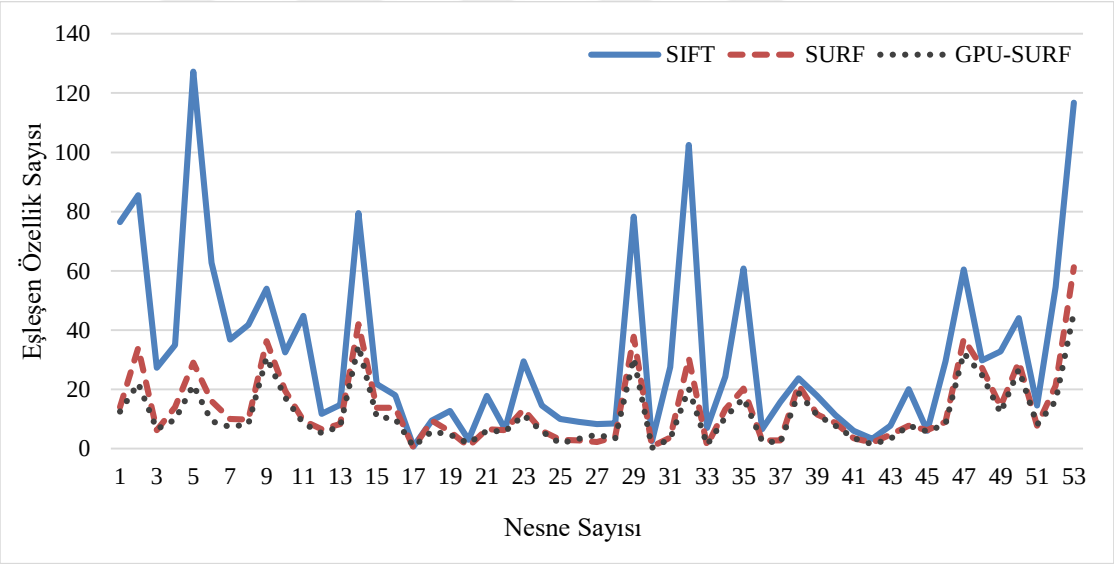


Şekil 3.7. SIFT, SURF, GPU-SURF yöntemleri ile nesne eşleştirme sonuçları

Nesneler üzerinde her bir algoritma için bulunan keypoint sayıları Şekil 3.8'de verilmiştir. Elde edilen sonuçlara göre SIFT yöntemi ile ortalama 236 keypoint bulunmuştur. Fakat SURF ve GPU-SURF algoritmaları ile sırasıyla ortalama 106 ve 87 özellik bulunabilmiştir. Bu sonuçlardan da görüldüğü gibi SIFT algoritması SURF ve GPU-SURF algoritmalarına göre daha fazla özellik bulmaktadır. Benzer sonuçlar eşleşen özellik sayılarında da ortaya çıkmaktadır. Şekil 3.9'da nesneler üzerinde her bir algoritma için eşleşen keypoint sayısı verilmiştir. SIFT yöntemi ile ortalama 32 özellik eşleştirmesi gerçekleşirken, SURF yöntemi ile 14 ve GPU-SURF yöntemi ile ise 11 özellik eşleştirmesi elde edilmiştir.



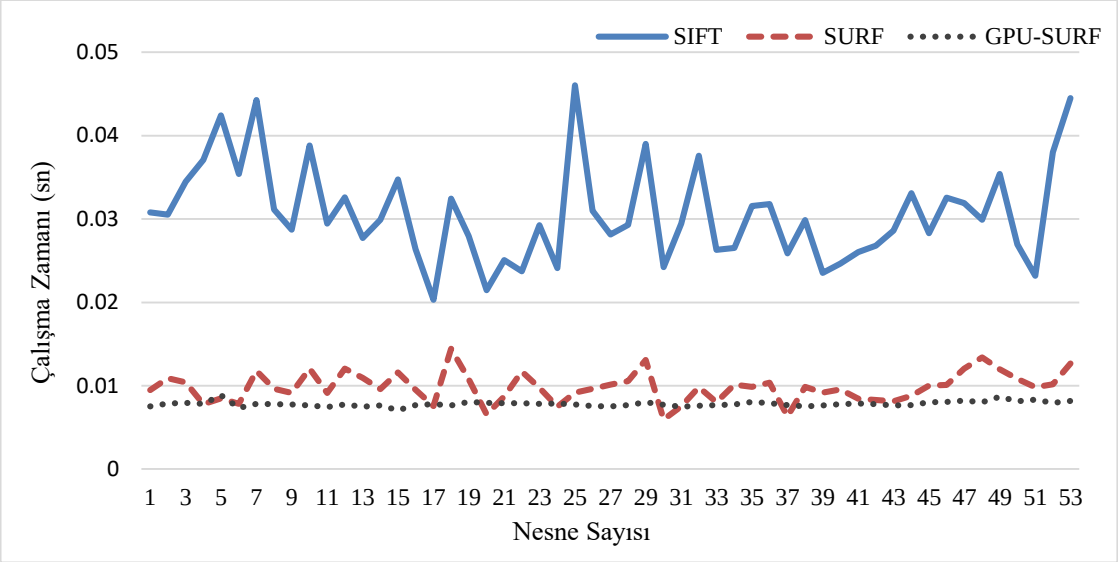
Şekil 3.8. Bulunan özellik sayısı bakımından SIFT, SURF ve GPU-SURF yöntemlerinin karşılaştırması



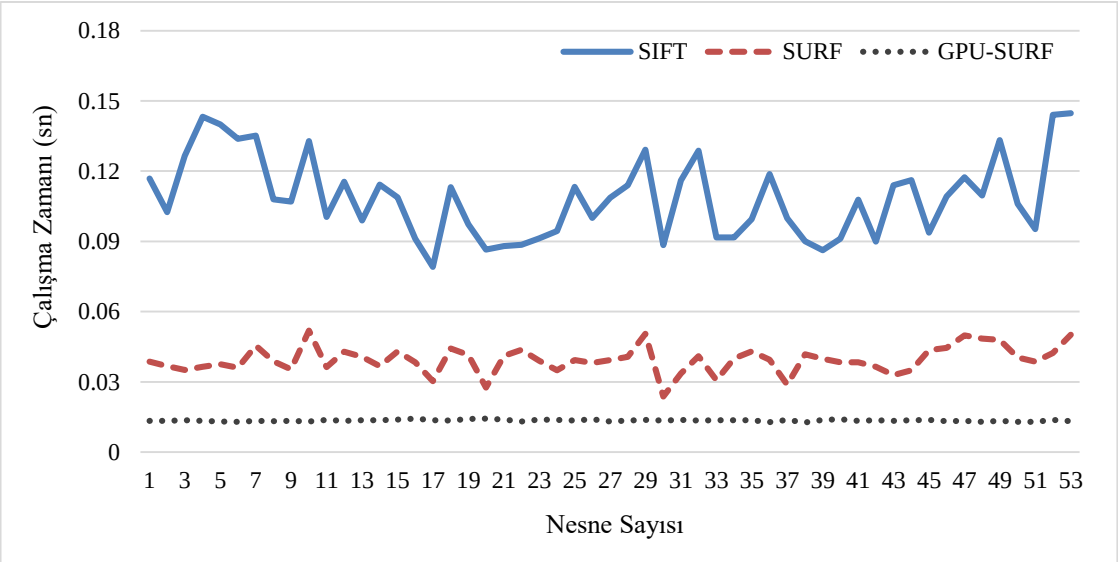
Şekil 3.9. Eşleştirilen özellik sayısı bakımından SIFT, SURF ve GPU-SURF yöntemlerinin karşılaştırması

Ayrıca yapılan çalışma her bir yöntem için çalışma zamanı bakımında da karşılaştırılmıştır. Çalışma zamanını karşılaştırmak amaçlı her bir imge farklı boyutlara çevrilmiş ve yöntemlerin çalışma hızları tespit edilmiştir. Orijinal imge çözünürlüğü 320x240 olmasına rağmen, imgeler 640x480, 960x720 ve 1280x960 çözünürlüklerine çevrilmiş ve her bir imge için çalışma süreleri hesaplanmıştır. 320x240 çözünürlüğünde imgeler için sonuçlar Şekil 3.10'da, 640x480 çözünürlüğünde imgeler için sonuçlar Şekil 3.11'de, 960x720 çözünürlüğünde imgeler için sonuçlar Şekil 3.12'de ve 1280x960 çözünürlüğünde imgeler için sonuçlar Şekil 3.13'te verilmiştir. Çalışma sonuçlarında

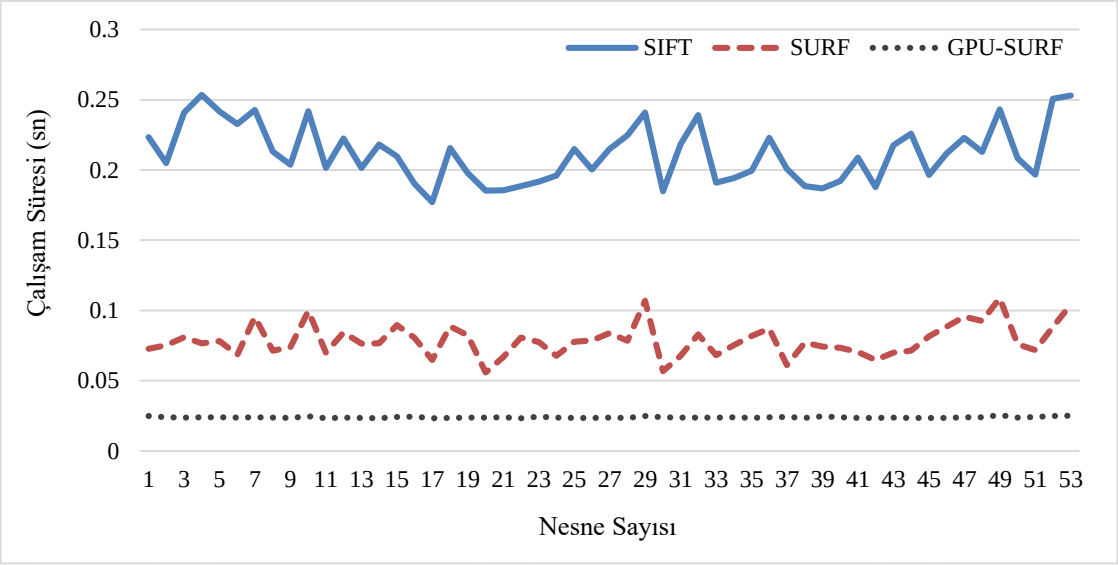
görüldüğü gibi imge çözünürlüğünün düşük olduğu imgelerde SURF ve GPU-SURF yöntemleri benzer çalışma zamanın sahiptir. Fakat imge çözünürlüğü büyüdükçe GPU avantajı ortaya çıkmakta ve CPU versiyonuna göre çok daha hızlı çalışmaktadır. SIFT algoritması ise tüm imge çözünürlüğünde en yavaş algoritma olmaktadır.



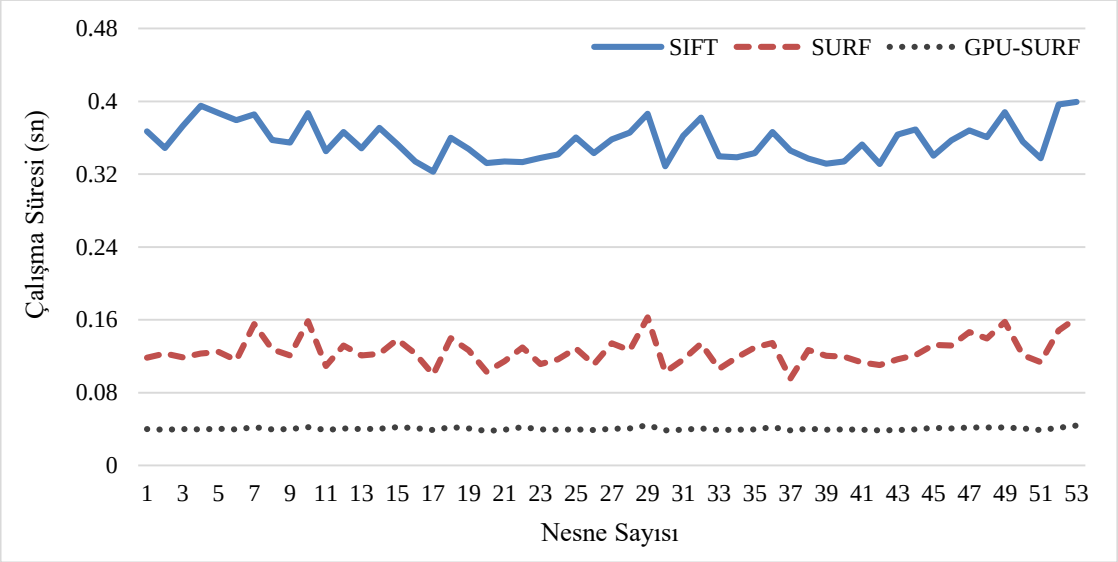
Şekil 3.10. 320x240 çözünürlüklü imgeler için SIFT, SURF, GPU-SURF yöntemlerinin çalışma zamanı



Şekil 3.11. 640x480 çözünürlüklü imgeler için SIFT, SURF, GPU-SURF yöntemlerinin çalışma zamanı



Şekil 3.12. 960x720 çözünürlüklü imgeler için SIFT, SURF, GPU-SURF yöntemlerinin çalışma zamanı



Şekil 3.13. 1280x960 çözünürlüklü imgeler için SIFT, SURF, GPU-SURF yöntemlerinin çalışma zamanı

Algoritmaların tüm nesneler için ortalama çalışma zamanı saniye cinsinden Tablo 3.1'de verilmiştir. Tabloda görüldüğü gibi SURF yöntemi SIFT yönteminden, GPU-SURF yöntemi ise SURF yönteminden yaklaşık 3 kat daha hızlıdır. GPU-SURF yöntemi için bu hız artışı özellikle 640x480 çözünürlüklü imge ile birlikte elde edilebilmektedir. Daha düşük çözünürlükte daha az hız artışı sağlanabilmektedir ama SURF algoritması tüm imge çözünürlüğü için benzer hızlanma sağlamıştır. Örneğin 320x240 çözünürlüklü imgede SURF yöntemi SIFT yöntemine göre 3,13 oranında daha hızlı çalışırken, GPU-SURF yöntemi SURF yöntemine göre sadece 1,26 oranında hızlanma sağlamıştır. Bunun nedeni

düşük çözünürlüklü imgelerde GPU paralel çalışma kapasitesinin tam olarak kullanılmamasıdır.

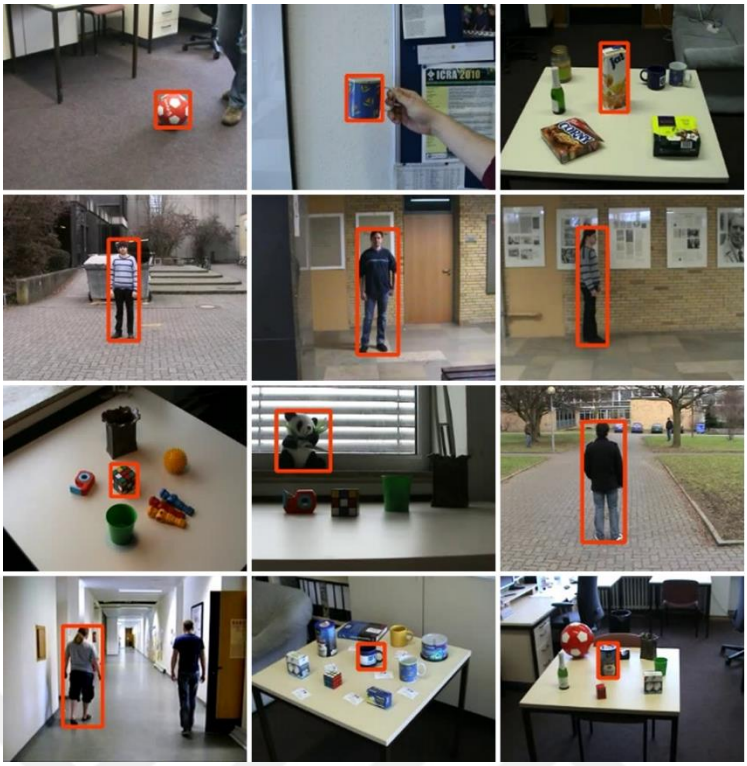
Tablo 3.1. SIFT, SURF ve GPU-SURF algoritmalarının farklı imge çözünürlüğünde ortalama çalışma zamanı

İmge Çözünürlüğü	SIFT		SURF		GPU-SURF		
	Zaman (sn)	Oran	Zaman (sn)	Oran	Zaman (sn)	Oran (SIFT)	Oran (SURF)
320x240	0,0307	1,00	0,0098	3,13	0,0078	3,94	1,26
640x480	0,1087	1,00	0,0394	2,76	0,0135	8,05	2,92
960x720	0,2119	1,00	0,0786	2,70	0,0239	8,87	3,29
1280x960	0,3567	1,00	0,1256	2,84	0,0403	8,85	3,12

3.3.2. Gerçek Zamanlı Nesne İzleme için Sonuçlar

Bu bölümde klasik SIFT, SURF ve GPU-SURF algoritmaları gerçek zamanlı nesne uygulaması üzerinde hem çalışma zamanı bakımından hem de nesne izleme başarışı bakımından test edilmiştir. Gerçek zamanlı test izleme uygulamasında Bonn Benchmark on Tracking (BoBoT) veri seti kullanılmıştır [139]. BoBoT veri seti içerisinde 320x240 çözünürlüğünde, avi formatında ve saniyede 25 çerçeve içeren toplam 12 kısa videodan oluşmaktadır. BoBoT veri setinde çerçeve sayısı 305 ile 1308 arasında değişmektedir. Ayrıca veri setinde her çerçeve için gerçek nesne konumu da verilmiştir. Veri setinde bulunan videoların ilk çerçeveleri ve izlenmesi amaçlanan nesneler Şekil 3.14'te verilmiştir.

BoBoT veri setindeki her bir videoda farklı özellikler test edilmektedir. Örneğin ilk videoda amaç hareket eden futbol topunu izlemektir. Bu videoda topun ani yer değişimlerinin yanı sıra kamera da hareketlidir. Üçüncü videoda ise ölçeklendirme, görünüm değişikliği, nesnesin ani hareketi ve kamera hareketi test edilmektedir. Beşinci videoda nesne kapalılığı ve kamera hareketi. Sekizinci videoda ise sadece ortam ışığının değişimi test edilmektedir. BoBoT veri setindeki hangi videonun hangi nesne izleme zorluğunu içerdiği Tablo 3.2'de verilmiştir [139]. Örneğin ilk videoda kamera hareketi, nesnenin ani hareketi, nesne yön değişimi ve nesne görünüm değişimi test edilmektedir. Videolarda test edilen özellikler "+" işareti ile test edilmeyen özellikler ise "-" işareti ile tabloda işaretleniştir. Videolarda birçok durum test edildiği için SIFT ve SURF algoritmaları bazı videolarda başarılı sonuçlar elde ederken bazı videolarda kısmen başarılı bazı videolarda ise başarısız olmuştur.



Şekil 3.14. BoBoT veri setindeki videoların ilk çerçeveleri ve izlenmesi amaçlanan nesneler

Tablo 3.2. BoBoT veri setindeki videolar ve her bir videoda test edilen zorluklar (Z1: Ölçeklendirme, Z2: Kapalılık, Z3: Kamera hareketi, Z4: Ani hareket, Z5: Benzer görünüm, Z6: Sahne karmaşıklığı, Z7: Yön değişimi, Z8: Görünüm değişimi, Z9: Ortadan kaybolma, Z10: Işık değişimi) [139].

Video No	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10
1	-	-	+	+	-	-	+	+	-	-
2	+	-	+	-	-	+	-	+	-	-
3	+	-	+	+	-	-	+	+	-	-
4	-	-	+	-	-	-	-	+	-	-
5	-	+	+	-	-	-	-	-	-	-
6	-	+	+	-	+	-	-	+	+	-
7	-	-	+	-	-	-	-	+	-	-
8	-	-	-	-	-	-	-	+	-	+
9	-	+	+	-	+	-	-	+	+	-
10	-	+	+	-	-	-	-	+	+	-
11	+	-	+	-	+	-	-	+	-	-
12	+	-	+	-	-	-	-	+	-	-

Yöntemlerin karşılaştırılması için gerçekleştirilen uygulamada, ilk çerçevedeki nesne arka plandan çıkarılmıştır. Elde edilen nesne diğer çerçevedeki imgeler ile karşılaştırılarak nesne konumunun doğru bir şekilde tespit edilmesine çalışılmıştır. Bu amaçla, SIFT, SURF

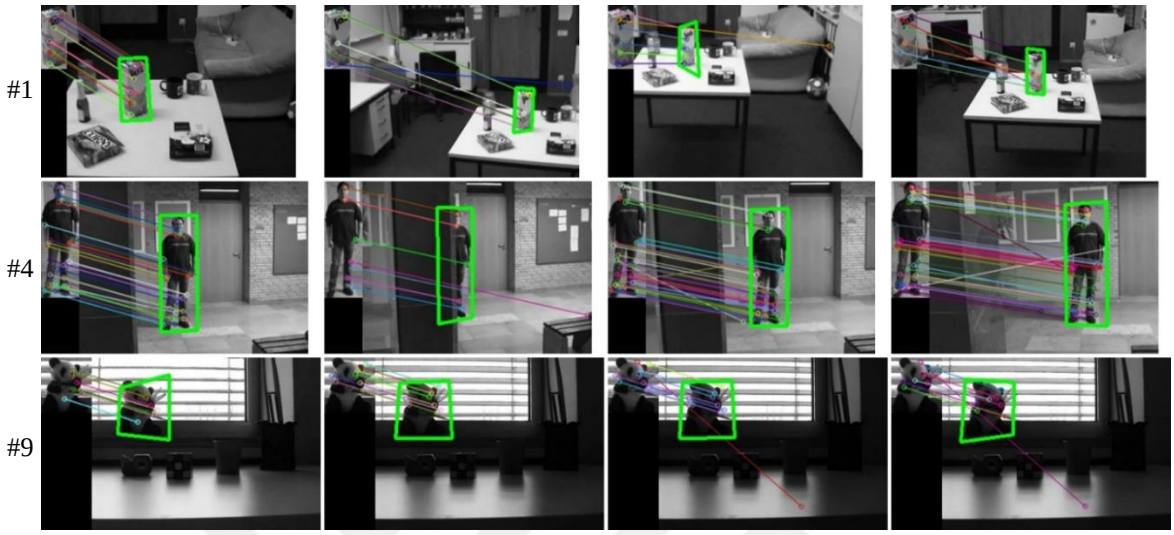
ve GPU-SURF yöntemleri kullanılarak gerçek zamanlı uygumlalar için yöntemler test edilmiştir.

SIFT, SURF ve GPU-SURF yöntemleri kullanılarak veri setindeki tüm videolar üzerinde test yapılmıştır. Test sonuçlarına göre SURF ve GPU-SURF yöntemleri benzer başarı sonuçları elde edilmiştir. Bu yöntemler arsındaki fark GPU-SURF yönteminin daha hızlı çalışmasıdır. Dolayısıyla GPU-SURF algoritması sadece çalışma zamanı karşılaştırmasında kullanılmıştır. Kısım 3.2.1'de bahsedildiği gibi SIFT yöntemi SURF yöntemine göre daha fazla özellik çıkartmaktadır. SIFT algoritması ile daha fazla özellik bulunabildiği için SIFT yöntemi ile nesne takip uygulaması daha başarılıdır. Ayrıca, bazı videolarda nesne boyutu çok küçük kaldığından bulunan özellik sayısı ciddi oranda düşüktür. Örneğin 11. videoda nesne çözünürlüğü 34x32 boyutundadır. Dolayısıyla bu tip videolarda hem SIFT yöntemi hem de SURF yöntemi başarısız olmuştur. Birinci videoda ise nesne boyutu 45x45 çözünürlüğündedir. Bu videoda SIFT algoritması kısmen başarılı olurken SURF algoritması nesne çözünürlüğünün düşük olduğundan başarısız olmuştur. Sonuç olarak SIFT algoritması düşük nesne çözünürlüğünde kısmen de olsa başarılı olabilmektedir. Fakat düşük nesne çözünürlüğü özellikle SURF algoritmasının başarı oranını ciddi derecede düşürmektedir.

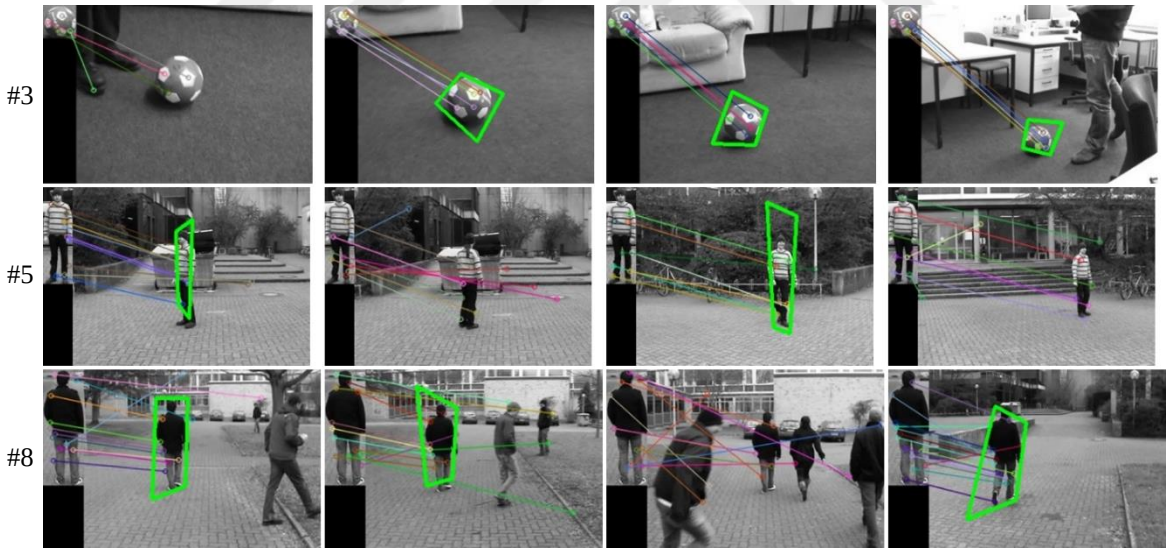
İlk olarak SIFT algoritması ile nesne izleme uygulamasının sonuçları elde edilmiştir. Şekil 3.15'te 3., 5. ve 8. videolar için SIFT algoritması ile nesne izleme sonuçları verilmiştir. Üçüncü video için sonuçlar ilk satırda verilmiştir. Bu videoda özellikle ani değişimler ve ölçeklendirme test edilmiştir. İkinci satırda beşinci video sonuçları verilmiştir. Bu videoda özellikle nesne kapalılığı test edilmiştir. Son satırında ise sekizinci video verilmiştir. Bu videoda sadece ortam ışık değişimi test edilmiştir. Sonuçlardan da görüldüğü gibi bu videolarda nesne izleme uygulaması başarılı bir şekilde çalışmaktadır. Sonuç olarak bu videolarda doğru tahmin edilen nesne oranı bakımından %70 ve üzeri başarı oranı yakalanabilmiştir.

Şekil 3.16'da 1., 4. ve 9. videolardan örnek çerçeve sonuçları verilmiştir. Gerçekleştirilen uygulamada nesne pozisyonu belirlendikten sonra nesne kutu gösterimi ile belirtilmiştir. Eğer nesne konumu tespit edilememişse veya yanlış tespit edildiği zaman sadece özellik eşleşmeleri gösterilmiştir. Videolar üzerinde yapılan testlerde, 1., 2., 4., 6. ve 9. videolar nesne izleme uygulamasında kısmen başarılı olmuştur. Örneğin nesne görünümünün değişmesi gibi bazı durumlarda nesne izleme başarılı olamamaktadır. Özellikle birinci ve dördüncü videoda nesne görünümü değişmektedir. Bu durumda nesne

izlemesi uygulaması nesneyi kaybetmekte, fakat nesne tekrar görünür olduğunda nesne izlemesi tekrar başlamaktadır. Sonuç olarak bu videolarda doğru tahmin edilen nesne oranı bakımından %30 ile %70 arası başarı oranı yakalanabilmiştir.



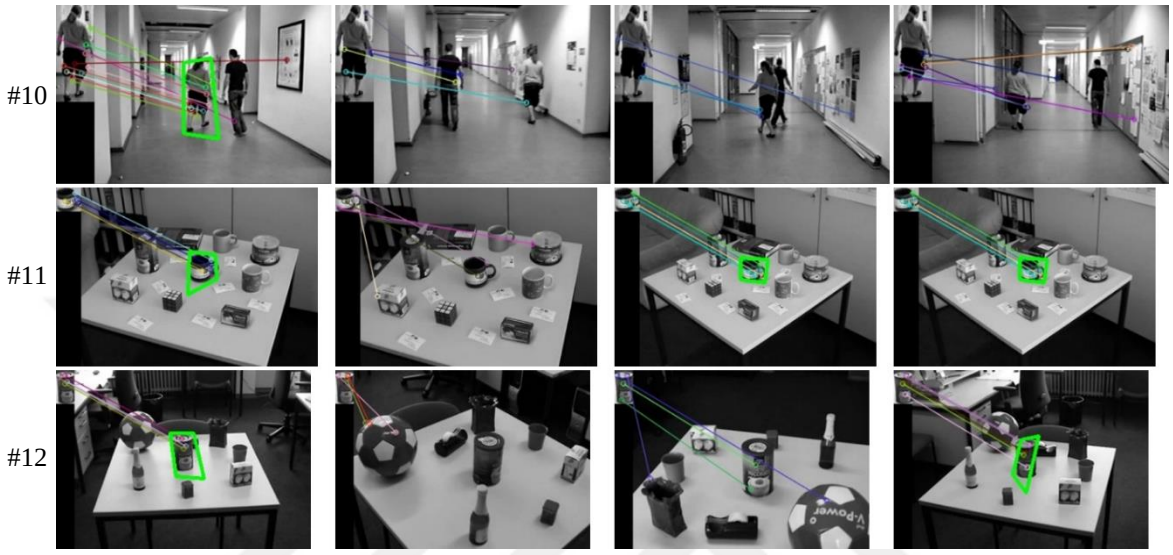
Şekil 3.15. BoBoT veri setinde başarılı sonuç veren videolardan örnekler



Şekil 3.16. BoBoT veri setinde kısmen başarılı sonuç veren videolardan örnekler

Şekil 3.17'de ise 10., 11., ve 12. videolar sırasıyla gösterilmiştir. 7., 10., 11. ve 12. videolarda genellikle nesne takibi başarılı bir şekilde gerçekleşmemiştir. Örnek olarak 11. videoda nesne görünümü değişmesinde dolayı 414. ile 752. video çerçevesi arasında nesne izleme uygulaması sonuç elde edememiştir. Aynı şekilde 12. videoda 590. ile 1233. video çerçeveleri arasında nesne izleme uygulaması ile çok az sonuç elde edilebilmiştir. 10.

videoda ise nesne izleme uygulaması eşleştirmeler bulmasına rağmen başarı oranı çok düşüktür. Dolayısıyla bu videolarda nesne izleme uygulamasının başarı oranı çok düşük kalmıştır. Elde edilen sonuçlara göre bu videolarda doğru tahmin edilen nesne oranı bakımından %30'un altındadır.



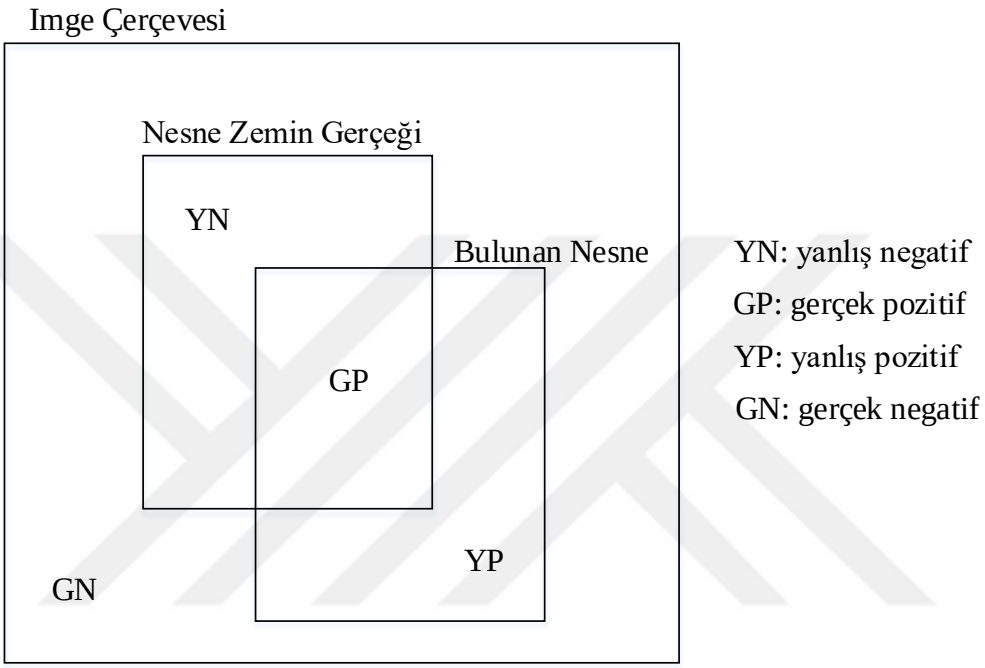
Şekil 3.17. BoBoT veri setinde başarısız sonuç veren videolardan örnekler

Nesne izleme uygulaması SURF yöntemi ile de çalıştırılmış ve başarı oranları elde edilmiştir. Nesne boyutunun çok küçük olması SURF algoritmasının başarı oranını direk etkilediği için 1., 2., 3., 7., 11. ve 12. videolarda sonuç elde edilememiştir.

Sonuç olarak nesne görünümünün dramatik olarak değiştiği videolarda hem SIFT algoritması hem de SURF algoritması nesne izleme işleminde başarılı olamamaktadır. Buna ek olarak SURF algoritması nesne boyutunun küçük olduğu durumlarda da başarılı olmamaktadır. Algoritmaların başarılı olduğu durumlar ise ortam ışık değişimi, nesnenin ani hareketleri, ölçeklendirme, nesne veya kamera hareketi ve nenenin kaybolması veya tekrar görünmesi durumlarıdır.

Nesne izleme uygulamalarının başarımını ölçmek literatürde çok sayıda yöntem kullanılmasına rağmen standart bir ölçüm sistemi yoktur. En temel yöntem nesne zemin gerçeği ile bulunan nesne konumunun karşılaştırmasıdır. Literatürde önerilen algoritmalardan birisi nesne ile bulunan konumun merkez noktalarının karşılaştırılmasıdır. Bu algoritmada gerçek merkez nokta ile bulunan merkez noktanın karşılaştırması yapılmaktadır. Diğer bir yöntemde ise bulunan nesne konumunun, gerçek nesne konumunu ne kadar kapsadığının ölçülmesi ile bulunur. İkinci algoritmada kullanılan yapı Şekil

3.18'de verilmiştir. Bu yapıda dört temek bölge vardır. Bunlar Yanlış Negatif (YN), Gerçek Pozitif (GP), Yanlış Pozitif (YP) ve Gerçek Negatiftir (GN). Nesne izleme uygulaması için düşünecek olursak YN ifadesi tahmin edilemeyen nesne nokta sayısını, GP ifadesi doğru tahmin edilen nesne nokta sayısını, YP yanlış tahmin edilen nesne nokta sayısını ve GN ise doğru tahmin edilen nesne olmayan nokta sayısını ifade etmektedir.



Şekil 3.18. Geliştirilen uygulamanın başarım ölçümü için kullanılan yapı

Gerçek negatif bölgesi geliştirilen uygulama için uygun değildir. Bu dört bölge ile doğruluk, seçicilik gibi birçok formül geliştirilmesine rağmen nesne izleme uygulamaları için bu yöntemler genel olarak kullanılmaz. Bunun nedeni bir video çerçevesinde gerçek negatif bölgesinin diğer bölgelere göre çok yüksek oranda olmasıdır. Dolayısıyla hatalı nesne tahminlerinde bile bu bölge için %90 yakın doğruluk sonuçları elde edilebilmektedir.

Tez projesinde kesişim tabanlı ölçüm araçları kullanılmıştır. Bu ölçümde tahmin edilen nesne bölgesi ile gerçek nesne bölgesinin kesişimi hesaplanmaktadır. Piksel tabanlı gerçek nesne ile tahmin edilen nesnenin kesişimi Denklem 3.2 ile hesaplanmaktadır.

$$Kesişim = \frac{GP}{GP + YN + YP} \tag{3.2}$$

Ayrıca tez çalışmasında kesinlik ve hassasiyet ölçümleri de hesaplanmıştır. Kesinlik formülü Denklem 3.3'te, hassasiyet formülü ise Denklem 3.4'te verilmiştir. Burada kesinlik ifadesi, doğruları tahmin etme oranını gösterirken, hassasiyet ifadesi ise tahminlerdeki doğru oranını göstermektedir.

$$Kesinlik = \frac{GP}{GP + YN} \tag{3.3}$$

$$Hassasiyet = \frac{GP}{GP + YP} \tag{3.4}$$

Kesinlik ve hassasiyet denklemleri ile F-skor ölçümü de hesaplanabilmektedir. F-skor ölçümü Denklem 3.5 ile hesaplanır.

$$F - Skor = 2 * \frac{Kesinlik * Hassasiyet}{Kesinlik + Hassasiyet} \tag{3.5}$$

Tez çalışmasında kullanılan bir başka ölçüm ise doğru tahmin edilen nesne oranıdır. Doğru tahmin edilen nesne oranı kesişim tabanlı ölçümlerdendir ve Denklem 3.6 ile hesaplanır.

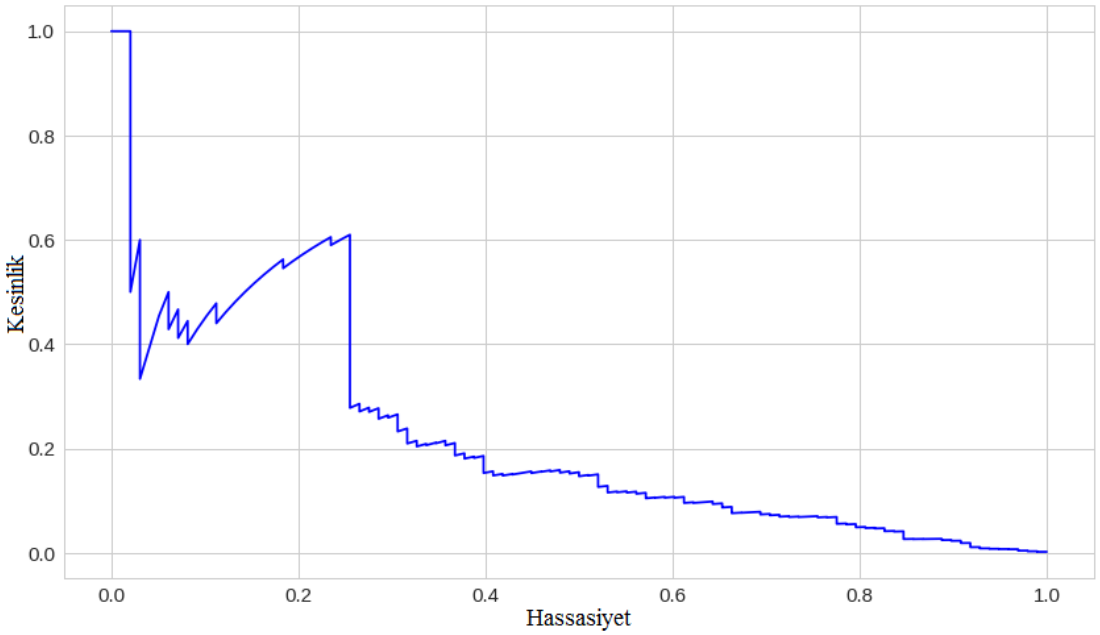
$$Kesişim\ Oranı = \frac{\sum_t \left\| \{t | \emptyset_c > T\}_{c=1}^{N} \right\|}{N} \tag{3.6}$$

Burada N videodaki toplam çerçeve sayısını $\emptyset_t$ ise c çerçevesindeki hesaplanan kesişim değerini ve T ise eşik değerini göstermektedir. Tahmin edilen bir nesnenin doğru olarak kabul edilmesi için gerekli olan T eşik değeri 0,5 olarak seçilmiştir.

Nesne tespiti ve izleme uygulamalarının değerlendirilmesi için literatürde yaygın olarak kullanılan bir başka değerlendirme ölçütü ortalama kesinlik (AP – average precision) değeridir. AP değerinin hesaplanmasında farklı yaklaşımlar getirilmiştir. Yaygın olarak kullanılan hesaplamalardan ilki, kesinlik / hassasiyet eğrisinin altında kalan alanın hesaplanması ile bulunmaktadır. Diğer bir yaklaşımda ise COCO veri setinin ana değerlendirme kriteri olan AP hesaplamasıdır. AP hesaplaması bazı kaynaklarda mAP olarak da geçmektedir.

Kesinlik hassasiyet eğrisinin hesaplaması için basit bir video üzerinde elma nesnesinin izleneceğini düşünelim. Kesimlik değeri o ana kadarki çerçeveler arasında bulunan nesne

sayısının bulunması gereken nesne sayısına oranı şeklinde hesaplanırken, hassasiyet o ana kadar bulunan nesne sayısının tüm video boyunca bulunması gereken nesne sayısına oranı şeklinde hesaplanmaktadır. Hesaplamalar yapılırken $\emptyset_c$ kesişim değeri için eşik değeri olarak 0,5 seçilmiştir. Örnek bir kesinlik hassasiyet eğirişi Şekil 3.19'da verilmiştir [140].



Şekil 3.19. Örnek bir kesinlik hassasiyet eğrisi

Daha sonra hassasiyet değeri 0 ile 1,0 arasında 0,1'lik adımlarla 11 noktaya bölünerek maksimum kesinlik değerleri hesaplanır. Hesaplanan 11 değerin ortalaması ile AP değeri elde edilmektedir. Burada hesaplanan değer AP olarak ifade edilirken diğer tüm nesne sınıflardan elde edilen sonuçların ortalaması ise ortalama yaklaşık doğruluk (mAP) olarak isimlendirilmektedir. Geliştirilen uygulamada tek nesne olduğu için AP ve mAP benzer sonuçlar elde edilmektedir.

mAP hesaplaması için kullanılan COCO mAP yaklaşımında ise kesişim değeri, farklı eşik değerleri ile hesaplanan kesişim değerlerinin ortalaması kullanılarak hesaplanmaktadır. Eşik değeri 0,5'ten başlayarak 0,95 değerine kadar 0,05 aralıklarla arttırılır. Hesaplanan başarı değerlerinin ortalaması kullanılarak mAP değeri hesaplanmaktadır.

Başarı sonuçları verilirken kesişim için LoU, kesinlik için KE, hassasiyet için HA, F skor için FS, doğru pozitif oran için TPR kısaltmaları kullanılmıştır. Ayrıca AP kısaltması

kesinlik hassasiyet eğri yaklaşımını mAP ise COCO mAP yaklaşımı ile elde edilen sonuçları göstermektedir.

SIFT yönteminin video izleme başarım sonuçları Tablo 3.3'te sunulmuştur. Nesne izleme uygulamasının her bir video için başarısız olma nedenleri tabloda verilmiştir. Başarısız olma nedenleri Görünüm Değişikliği (GD), Benzer Görünüm (BG), Kamera veya Nesne hareketi (KN) ve Nesne Çözünürlüğü (NÇ) olarak kategorize edilmiştir. Ayrıca Tablo 3.3'te videoların yüzdesel olarak başarısı ortalama olarak verilmiştir. Başarılı oranı yüksek olan videolar için başarısızlık nedeni hücresi boş bırakılmıştır. Tabloda görüldüğü gibi genel nesne izleme uygulaması için başarı oranları düşük kalmaktadır. Sadece 3., 5. ve 8. videolarda başarı oranı kabul edilebilir düzeydedir.

Tablo 3.3. Nesne izleme uygulamasında her bir video için SIFT algoritmasının başarı sonuçları (GD: Görünüm Değişikliği, BG: Benzer Görünüm, KN: Kamera veya Nesne hareketi, NÇ: Nesne Çözünürlüğü, TBN: Test Başarısızlık Nedeni)

Video No	HA	KE	FS	LoU	TPS	AP	mAP	TBN
1	0.37	0.45	0.32	0.30	0.30	0.55	0.26	GD
2	0.43	0.50	0.38	0.32	0.35	0.55	0.20	NÇ
3	0.75	0.78	0.76	0.63	0.83	0.94	0.35	
4	0.37	0.52	0.36	0.31	0.33	0.56	0.21	GD
5	0.97	0.89	0.92	0.87	0.95	0.98	0.79	
6	0.34	0.52	0.36	0.31	0.32	0.60	0.21	BG, KN
7	0.28	0.36	0.28	0.25	0.27	0.62	0.20	GD, NÇ
8	0.87	0.89	0.86	0.80	0.86	0.91	0.70	
9	0.46	0.51	0.42	0.33	0.34	0.57	0.16	BG, GD, KN
10	0.26	0.30	0.22	0.19	0.19	0.45	0.14	KN, GD
11	0.35	0.39	0.34	0.29	0.29	0.56	0.19	GD, NÇ
12	0.22	0.23	0.21	0.19	0.21	0.46	0.17	GD
Ortalama	**0.47**	**0.53**	**0.45**	**0.40**	**0.44**	**0.65**	**0.30**	

SURF yöntemi ile nesne izleme uygulamasının sonuçları Tablo 3.4'te verilmiştir. Tablodan görülebileceği gibi SURF algoritması ile gerçekleştirilen nesne izleme uygulaması başarım oranları, SIFT algoritması ile gerçekleştirilen uygulamaya göre yaklaşık %8 daha az başarılıdır. Bunun temel nedeni SURF yöntemi küçük boyutlu nesne durumunda daha az özellik noktası ürettir, dolayısıyla nesne konumu tahmininde daha az başarılı olmaktadır.

Tablo 3.4. Nesne izleme uygulamasında her bir video için SURF algoritmasının başarı sonuçları (GD: Görünüm Değişikliği, BG: Benzer Görünüm, KN: Kamera veya Nesne hareketi, NÇ: Nesne Çözünürlüğü, TBN: Test Başarısızlık Nedeni)

Video No	HA	KE	FS	LoU	TPS	AP	mAP	TBN
1	0.26	0.32	0.26	0.25	0.26	0.51	0.24	GD, NÇ
2	0.36	0.44	0.33	0.28	0.32	0.54	0.19	NÇ
3	0.70	0.72	0.69	0.57	0.73	0.90	0.31	NÇ
4	0.30	0.44	0.29	0.25	0.27	0.49	0.18	GD
5	0.94	0.88	0.90	0.85	0.94	0.98	0.78	
6	0.34	0.51	0.35	0.30	0.31	0.63	0.20	BG, KN
7	0.26	0.27	0.26	0.23	0.27	0.62	0.20	GD, NÇ
8	0.78	0.80	0.76	0.70	0.82	0.90	0.58	
9	0.35	0.46	0.30	0.24	0.22	0.43	0.12	BG, GD, KN
10	0.26	0.30	0.21	0.18	0.17	0.42	0.14	KN, GD
11	0.30	0.31	0.28	0.24	0.29	0.49	0.17	GD, NÇ
12	0.17	0.19	0.16	0.14	0.14	0.26	0.11	GD
Ortalama	**0.42**	**0.47**	**0.40**	**0.35**	**0.39**	**0.60**	**0.27**	

GPU üzerinde gerçekleştirilen SURF yöntemi ile nesne izleme başarı sonuçları Tablo 3.5'te verilmiştir. GPU-SURF yöntemi, SURF yöntemine göre yine daha az özellik bulduğu için başarı oranları SURF yöntemine göre daha düşük çıkmıştır. Sonuçlardan görüldüğü gibi nesne izleme başarı oranları oldukça düşüktür. Dolayısıyla SIFT, SURF ve GPU-SURF algoritmaları klasik versiyonları ile nesne izleme uygulamaları için uygun değildir.

Ayrıca SIFT, SURF ve GPU-SURF algoritmaları çalışma zamanı bakımından da karşılaştırılmıştır. Nesne izleme uygulamalarında önemli olan işlemin gerçek zamanlı bir şekilde yapılabilmesidir. Bunun için de literatürde kabul görmüş değer saniyede 30 video çerçevesinin işlenmesidir. Yapılan testlerde elde edilen ortalama sonuçlara bakıldığında SIFT algoritması ile saniyede 14 video çerçevesi işlenebilmektedir. SURF algoritması ile 25 video çerçevesi işlenebilirken GPU-SURF algoritması ile de ortalama 42 video çerçevesi işlenebilmektedir. Sonuçlarda görüldüğü gibi SIFT algoritması ile daha başarılı sonuçlar elde edilmesine rağmen, gerçek zamanlı uygulamalar için çok yavaş kalmaktadır. Ayrıca daha yüksek video çözünürlüğünde SIFT algoritmasının çalışma hızı daha da düşecektir. GPU-SURF algoritmasına bakıldığında gerçek zamanlı uygulamalar için en uygun algoritma olduğu söylenebilir. Ayrıca daha yüksek video çözünürlüğünde GPU kullanımının avantajı daha fazla ortaya çıkacaktır.

Tablo 3.5. Nesne izleme uygulamasında her bir video için GPU-SURF algoritmasının başarı sonuçları (GD: Görünüm Değişikliği, BG: Benzer Görünüm, KN: Kamera veya Nesne hareketi, NÇ: Nesne Çözünürlüğü, TBN: Test Başarısızlık Nedeni)

Video No	HA	KE	FS	LoU	TPS	AP	mAP	TBN
1	0.27	0.33	0.27	0.26	0.26	0.52	0.25	GD, NÇ
2	0.30	0.38	0.27	0.24	0.26	0.48	0.16	NÇ
3	0.64	0.70	0.63	0.53	0.68	0.88	0.29	NÇ
4	0.31	0.44	0.29	0.25	0.27	0.49	0.17	GD
5	0.94	0.88	0.89	0.85	0.93	0.97	0.78	
6	0.34	0.49	0.34	0.29	0.33	0.61	0.20	BG, KN
7	0.26	0.29	0.25	0.23	0.26	0.61	0.20	GD, NÇ
8	0.71	0.70	0.68	0.60	0.69	0.83	0.44	
9	0.31	0.46	0.27	0.21	0.19	0.39	0.11	BG, GD, KN
10	0.25	0.29	0.20	0.17	0.15	0.37	0.13	KN, GD
11	0.24	0.26	0.22	0.19	0.21	0.43	0.12	GD, NÇ
12	0.17	0.19	0.16	0.14	0.14	0.26	0.11	GD
Ortalama	**0.39**	**0.45**	**0.37**	**0.33**	**0.37**	**0.57**	**0.25**	

3.4. Sonuç

Bu bölümde SIFT, SURF ve GPU-SURF yöntemleri iki farklı uygulama üzerinde test edilmiştir. İlk olarak her bir algoritma için bulunan özellik sayısı ve çalışma zamanı karşılaştırılmıştır. Elde edilen sonuçlara göre GPU tabanlı SURF yöntemi SURF yöntemine göre 3 kat, SIFT yöntemine göre ise yaklaşık 9 kat daha hızlı çalışmaktadır. Testlerde elde edilen bir başka sonuç ise SIFT yöntemi SURF ve GPU-SURF yöntemlerine göre imge üzerinde özellik bulma ve eşleştirme bakımından daha başarılıdır. İkinci olarak, SIFT, SURF ve GPU-SURF yöntemleri gerçek zamanlı nesne takip uygulamasında test edilmiştir. Testlerden elde edilen sonuçlara göre SIFT yöntemi diğer yöntemlere göre daha istikrarlı ve daha başarılıdır. SURF ve GPU-SURF yöntemlerinde, başarı oranı bakımından benzer sonuçlar elde edilmiştir fakat GPU-SURF algoritması ciddi derecede daha hızlı çalışmaktadır. Yaptığımız testler sonucunda HD videolarda bile GPU-SURF algoritmasının gerçek zamanlı uygulamalar için uygun olduğu görülmüştür. Nesne izleme uygulamasında BoBoT veri setindeki 12 farklı video ile test edilmiştir. SIFT algoritması 3 test senaryosunda başarılı olmuş, 5 test senaryosunda kısmen başarılı olmuş ve geri kalan 4 test senaryosunda ise başarısız olmuştur. SURF algoritması ise sadece 2 test senaryosunda başarılı olurken, 4 test senaryosunda kısmen başarılı oluş, gere kalan 6 test senaryosunda ise başarısız olmuştur.

4. ÖZELLİK ÇIKARTMA ALGORİTMALARI İÇİN DBSCAN VE GAUSS ALGORİTMALARI İLE YENİ BİR NESNE İZLEME UYGULAMASI

Bu bölümde Nokta Tabanlı Özellik Çıkartma (NTÖÇ) algoritmaları nesne izleme uygulamalarında kullanımı için yeni bir yaklaşım önerilmiştir. Genelde NTÖÇ algoritmaları nesne tespiti, nesne eşleştirme gibi uygulamalarda kullanılmaktadır. NTÖÇ algoritmaları nesne izleme uygulamalarında sıkça kullanılan yaygın özelliklerdendir.

Tez çalışmasının bu bölümünde gerçekleştirilen nesne takip sistemi tanıtılmaktadır. Kısım 4.2'de SIFT ve SIFT algoritmasından esinlenerek geliştirilen algoritmalara yer verilmiştir. Kısım 4.3'te geliştirilen uygulamanın gerçekleştirim detayları aktarılmıştır. Kısım 4.4'te ise deneysel sonuçlar verilmiştir. Deneyler için kullanılan veri seti tanıtılmış ve deney şartları açıklanmıştır. Son alarak deney sonuçlarından elde edilen sonuçlar verilmiştir.

4.1. Giriş

Nokta tabanlı özellik çıkarım algoritmaları için Moravec Operatörü [8], Harris Nokta Detektörü[9], KLT Detektörü [10] ve SIFT Detektörü [11] örnek olarak verilebilir. Moravec operatörü imge üzerindeki kenar bilgilerini kullanarak nesne özelliklerini çıkartmaktadır. Harris Nokta Detektörü ise imge üzerindeki köşe noktaları özellik noktası olarak bulmaktadır. KLT Detektörü, Harris Nokta Detektörü ile benzer yapıdadır fakat KLT yönteminde ayırt edici noktaların arasında belli bir mesafe olmasını zorunlu tutar. KLT ve Harris detektörleri öteleme ve döndürme değişimine karşı dayanıklıdır fakat afin dönüşümü ve nesne bakış açısının değişimine karşı dayanıksızdır. Bu probleme karşı SIFT yöntemi önerilmiştir. SIFT yöntemi, farklı çözünürlükte ve ölçekte özelliklerin hesaplanması ile birlikte bakış açısı ve afin dönüşümüne karşı da dayanıklılık kazanmıştır. SIFT yöntemi temel olarak üç adımdan oluşmaktadır. Bu adımlar ayırt edici nokta çıkarımı, tanımlayıcı oluşturulması ve özellik eşleştirme adımlarıdır. SIFT yöntemini temel olarak geliştirilen yöntemler bu üç adımından birisini, birkaçını veya hepsinin geliştirilmesi sonucu ortaya çıkmaktadır. SIFT yönteminin geliştirilmesine yönelik yapılan çalışmalar sonucunda PCA-SIFT [123], GSIFT [141], CSIFT [142], SURF [13], ASIFT [12] ve RIFF[14] yöntemleri geliştirilmiştir. Geliştirilen bu yöntemler SIFT uygulamasının üç temel adımında geliştirmeler sunmaktadırlar [143].

Bu bölümde geliştirilen nesne izle uygulaması için SIFT yöntemi ve türevleri özellik çıkartma amaçlı kullanılmıştır. SIFT yöntemi ve türevlerinin ilk iki adımında değişiklik yapılmamış, sadece özellik eşleştirme kısmında DBScan algoritması kullanılarak özelliklerin iyileştirilmesi sağlanmıştır. Daha sonra elde edilen özellikler ile nesne konum tespiti gerçekleştirilmiş ve nesne izleme uygulaması geliştirilmiştir. Nesne izleme uygulaması için Gauss yöntemi geçmiş nesnelerin konumlarını tutma ve sonraki video çerçevelerindeki nesne konumunu tahmin etmek için kullanılmıştır.

4.2. SIFT ve SURF Harici Diğer Nokta Tabanlı Özellik Çıkartma Algoritmalarının İncelenmesi

SIFT ve SURF algoritmasının yanı sıra literatürde birçok nokta tabanlı özellik çıkarma algoritması vardır. Bunlardan bazıları PCA-SIFT, CSIFT ve ASIFT algoritmalarıdır. Bu algoritmalar incelendiğinde her durumda başarılı bir şekilde çalışan bir yöntem bulunmamaktadır. Tez çalışmasında kullanılan SIFT, SURF ve GPU-SURF yöntemleri bölüm 3.2'de ayrıntılı olarak anlatılmıştır. Dolayısıyla bu yöntemlerin nasıl çalıştığıyla ilgili bilgi bu bölümde yer verilmemiştir. Bu bölümde geriye kalan PCA-SIFT, GSFT, CSIFT ve ASIFT yöntemleri kısaca anlatılacaktır.

SIFT algoritmasında tanımlayıcı oluşturulması aşamasında her bir ayırt edici noktayı tanımlamak için 128 boyutlu bir vektör kullanır. Bu yüksek boyut yöntemin yavaş çalışmasına neden olmaktadır. Her bir ayırt edici noktanın boyutunu düşürmek için Y. Ke tarafından [123], SIFT yöntemindeki histogram kullanımı yerine temel bileşen analizi kullanılması önerilmiştir. Geliştirilen yöntem PCA-SFT alarak anılmaktadır. Temel bileşen analizi veri boyutunu düşürmede yaygın olarak kullanılan etkili bir yöntemdir. Temel bileşen analizi veri boyutunu küçültürken, veri setinde bulunan bilgilerin birçoğunu barındırmayı sürdürür. Temel bileşen yöntemi SIFT yönteminde kullanılan HOG işleminin yerini alır. Projeksiyon matrisi oluşturularak tanımlayıcı oluşturulur böylece vektör boyutu önemli derecede küçülmesi sağlanır. PCA-SIT yönteminde ilk olarak ayırt edici nokta çevresinde 41x41 boyutunda bölge alınarak, 2x39x39=3042 elemanlı bir vektöre dönüştürür. Daha sonra bu vektör ile koveryans matrisi oluşturulur ve temel bileşen analizi yöntemi uygulanır. Elde edilen özellikler ile PCA-SIFT yöntemi için gerekli olan projeksiyon matrisi oluşturulmaktadır. Böylece, her ayırt edici nokta için SIFT yönteminde tanımlanan 128

boyutlu vektör yerine PCA-SIFT yöntemi yaklaşık 20 boyutlu vektör tanımlamaktadır [123, 143].

GSIFT [141] yöntemi ise global bilgileri SIFT yöntemi ile birleştirir. GSIFT yönteminde temel fikir, global doku bilgisinin eklenmesi sonucu, geniş yelpazede eğri şekil bilgisinin elde edilmesidir. Her tespit edilen ayırt edici nokta için 2 parçalı bir vektör tanımlanır. Vektörün ilk parçası yerel özellikler ile SIFT tanımlayıcısıdır. İkinci parça ise, benzer yerel özellikleri ayırt etmek için kullanılacak olan global doku vektörüdür. GSIFT tarafından üretilen iki parçalı özellik vektörü Denklem 4.1'de verilmiştir [141].

$$F = \begin{bmatrix} \omega SS \\ (1-\omega)GS \end{bmatrix} \tag{4.1}$$

Burada S 128 boyutlu SIFT vektörü, G ise 60 boyutlu global vektörü ve ω ise göreceli ağırlık faktörüdür. Global vektör de SIFT vektörünün oluşturulduğu gibi histogram hesaplanması ile elde edilir. GSIFT yönteminde Hessian Matrisi de kullanılmaktadır. Bu matris her pikselin maksimum eğriliğini hesaplar. Her bir piksel için dairesel bir bölge ayrılarak, her bir ayırt edici noktanın açısal ve radyal değerleri hesaplamaktadır. Böylece eğrilik değerleri hesaplanır [141].

SIFT yöntemi sadece gri seviye imgelerde çalışmaktadır. Dolayısıyla renkli imgelerdeki birçok renk bilgisi göz ardı edilir. A. A. Frag [142] önerdiği CSIFT yöntemi ile bu eksikliğe değinmiştir. CSIFT yöntemi temel olarak SIFT yöntemine renk bilgisini dahil etmektedir. CSIFT algoritmasında renk değişmezliği Kubelka-Munk [144] yöntemi ile sağlanmaktadır. Bu yöntem renkli cisimlerin yansıyan spektrumunu Denklem 4.2 ile modellemektedir [142].

$$E(\lambda, \vec{x}) = e(\lambda, \vec{x})(1 - p_f(\vec{x}))^2 R_\infty(\lambda, \vec{x}) + e(\lambda, \vec{x}) p_f(\vec{x}) \tag{4.2}$$

Burada λ dalga boyu, $\vec{x}$ imge pozisyonunu temsil eden 2 boyutlu vektör, $e(\lambda, \vec{x})$ ışıklandırma spektrumunu, $p_f(\vec{x})$ $\vec{x}$ noktasının Fresnel yansıması katsayısını, $R_\infty(\lambda, \vec{x})$ materyal yansımasını ve $E(\lambda, \vec{x})$ ise gözlem pozisyonundan yansıyan spektrumu temsil etmektedir. Denklem 4.2 kullanımı ile CSIFT algoritması $E(\lambda, \vec{x})$ ifadesinin birinci ve ikinci türevlerini hesaplar ve H renk değişmezliğini Denklem 4.3'te görüldüğü gibi bu iki türevin oranı olarak hesaplanmaktadır [142].

$$H = \frac{E_\lambda}{E_{\lambda\lambda}} \tag{4.3}$$

CSIFT, RGB renk modeli ile $(E, E_\lambda, E_{\lambda\lambda})$ ifadesini eşleşme modeli olarak tanımlamaktadır. Bu ifadenin ışık durumu, yansıma katsayısı, gözlem pozisyonu ve yüzey yönü gibi bilgiler ile ilgi yoktur. Böylece ayırt edici noktaları tespit edeceğimiz GF piramidi için, SIFT yöntemindeki $I(x, y)$ ile CSIFT yöntemindeki $H(x, y)$ değiştirerek mümkün olmaktadır [142, 143].

SIFT yöntemi afin dönüşümünde çok başarılı değildir. Bu probleme karşı J.M. Morel [12] ise ASIFT modelini tanıtmıştır. Bu modelde imgeyi düzeltmek ve afin değişmezliği sağlamak için afin dönüşümü parametresi önerilmiştir. Afin dönüşüm performansını arttırmak için, ASIFT yöntemi kameranın optik eksen etrafından dönüşünü modellemektedir. Bakış acısının değişmesi sonucu oluşan afin dönüşümünü modellemek için Denklem 4.4 kullanılır [12, 143].

$$u(x, y) \rightarrow u(ax + by + e, cx + dy + f) \tag{4.4}$$

Denklem 4.4'te verilen afin modelinde kameranın nesneden uzak olduğu ve kamera hareketlerinin ölçülen nesne görüntüsünde bozulmalara yol açtığı varsayılmaktadır. Ölçülen nesnenin normal düzleminden üretilen açı ve kamera optik ekseninin eşleme düzlemi, boylam açısı olarak tanımlanmaktadır. ASIFT yönteminde ilk önce görüntüye döndürme dönüşümü eklenir. Daha sonra x yönünde görüntü yatırma $u(x, y) \rightarrow u(tx, y)$ işlemi uygulanarak seri halinde afin imgesi elde edilir. Hem döndürme dönüşümü hem de yatırma dönüşümü, belirli bir aralıktaki boylam açısını ve enlem açısını değiştirerek sağlanır. Bu işlemlerden sonra, ASIFT ayırt edici noktaları tespit eder ve afin görüntüsünden tanımlayıcıyı oluşturur. Eşleştirme işleminde, ASIFT yöntemi ile daha fazla ayırt edici nokta tespit edilmektedir. Ayrıca ASIFT yöntemi SIFT yöntemine göre daha az hatalı eşleştirme yapmaktadır [12].

SIFT yöntemi ve SIFT yönteminden esinlenerek geliştirilen yöntemler 5 farklı durum ile değerlendirilebilir. Bu durumlar ölçeklendirme, döndürme, ışık değişimi, bulanıklık ve afin dönüşümüdür [143]. Ayrıca yöntemlerin çalışma hızları da önemlidir. Tek tek algoritmaların bu beş durum için nasıl sonuçlar verdiğini inceleyecek olursak, SIFT yöntemi ölçeklendirme ve döndürme durumlarında başarılı sonuçlar vermektedir. Diğer durumlarda ise bu beş algoritma söz konusu olduğunda ortalama değerler elde edilmektedir. CSIFT

algoritmasında ise bulanıklık durumunda SIFT yöntemine göre daha başarılıdır. Ayrıca ölçeklendirme ve döndürme durumlarında SIFT algoritması ile benzer sonuçlar elde edilmiştir. Fakat CSIFT yöntemi SIFT yöntemine göre daha yavaş çalışmaktadır. ASIFT algoritması özellikle afin durumunda iyi sonuçlar vermektedir. Diğer durumlarda ise ortalama değerler yakalanmıştır, fakat bu 5 yöntem içerisinde en yavaş çalışan yöntemdir. PCA-SIFT yöntemi ölçeklendirme, döndürme, ışık değişimi ve bulanıklık durumlarında en başarılı sonuçları vermese bile ortalama üstü sonuçlar elde edilmiştir. Afin durumu ve çalışma hızı bakımından ise ortalama sonuçlar elde edilmiştir. Son olarak SURF algoritmasında genel olarak ortalama değerler elde edilirken, çalışma hızı en yüksek olan yöntemdir. Yöntemler ve hangi durumda nasıl çalıştıkları 4 farklı derecelendirme ile Tablo 4.1.'de verilmiştir. Kullanılan derecelendirmeler en iyi, çok iyi, iyi ve orta olarak belirlenmiştir [143].

Tablo 4.1. SIFT ve varyantlarının belli video zorlukları için karşılaştırma sonuçları [143].

Algoritma	Ölçeklendirme	Döndürme	Işık Değişimi	Bulanıklık	Afin Dönüşümü	Çalışma Hızı
SIFT	En iyi	En iyi	Çok iyi	İyi	İyi	Çok iyi
SURF	Orta	Orta	Orta	Orta	Orta	En iyi
PCA-SIFT	Çok iyi	Çok iyi	Çok iyi	Çok iyi	İyi	Çok iyi
GSIFT	İyi	İyi	En iyi	En iyi	İyi	Çok iyi
CSIFT	En iyi	En iyi	İyi	Çok iyi	Çok iyi	İyi
ASIFT	İyi	İyi	Orta	Orta	En iyi	Orta

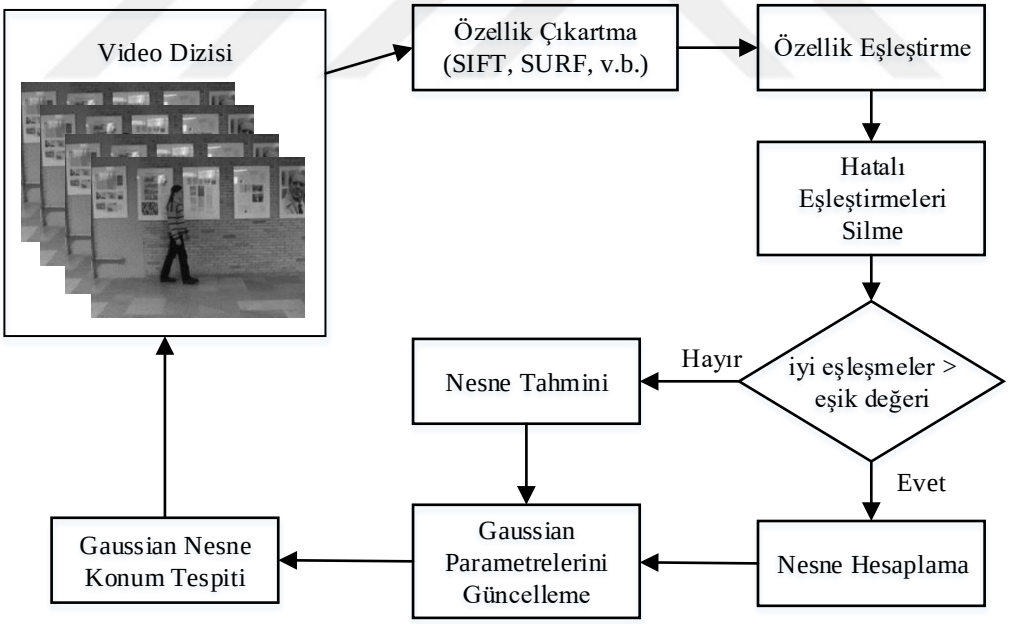
Tablo incelendiğinde her algoritmanın avantajlı ve dezavantajlı olduğu durumlar bulunmaktadır. Özellik çıkarım algoritmaları nesne görünümü tamamen değişmediği sürece, nesnenin bir sonraki imge veya video çerçevesinde bulunup bulunmadığını tespit etmek amaçlı kullanılabilir. Dolayısıyla nesne takip uygulamalarında da sıkça tercih edilmektedir.

Nesne izleme uygulamalarında en büyük problem nesnenin video üzerinde kaybolup tekrar göründüğünde aynı nesne olup olmadığını tespit etmektir. Özellik çıkarım algoritmaları bu amaç doğrultusunda kullanılan en yaygın yöntemlerdendir. Ayrıca nesne izleme uygulamalarında bir başka önemli nokta ise saniyede işlenen video çerçeve sayısıdır. Yapılan çalışmalarda elde edilen sonuçlara göre bu altı yöntemden en hızlı çalışan yöntem SURF yöntemidir. SIFT algoritması ise en hızlı çalışan ikinci algoritmadır. Dolayısıyla bu bölümde SIFT ve SURF algoritmaları çalışma hızlarının yüksek olmasından dolayı seçilmiştir. Ayrıca SURF algoritmasının GPU versiyonu gerçekleştirilmiştir. Böylece SURF algoritmasının daha hızlı çalışması sağlanmıştır.

Bu bölümde SIFT, SURF ve SURF algoritmasının GPU versiyonu (GPU-SURF) Bölüm 4.3'te anlatılan algoritma ile entegre edilmiştir. Algoritmaların geleneksel versiyonları ile önerilen algoritmanın entegre edilmiş halleri nesne izlemede doğruluk ve çalışma hızı bakımından karşılaştırılmıştır.

4.3. Önerilen Algoritma

Bu bölümde gerçekleştirilen Nokta Tabanlı Nesne İzleme uygulaması (NTNİ) detaylı bir şekilde anlatılacaktır. NTNİ uygulaması ile kullanılan özellik çıkartma algoritmasından bağımsız bir şekilde nesne izleme gerçekleştirilebilmektedir. Kullanılan yapı özellik çıkartma algoritmasına ve özellik eşleştirme aşamasına karışmamaktadır. Bunun yerine 3 aşamada nesne izleme gerçekleşmektedir. Bu aşamalar hatalı özelliklerin bulunması, nesne modeli ve nesne izleme aşamalarıdır. NTNİ uygulamasının genel yapısı Şekil 4.1'de verilmiştir.



Şekil 4.1. NTNİ Uygulamasının genel akış şeması

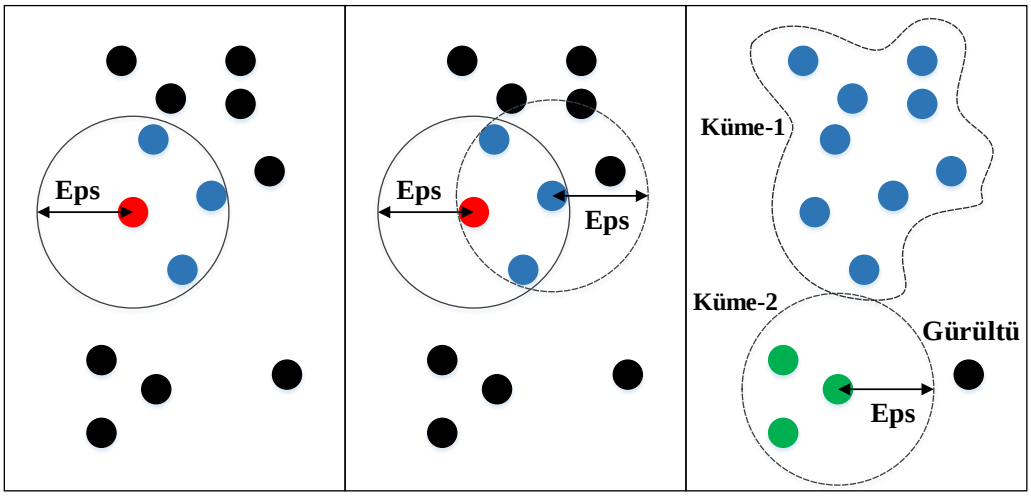
4.3.1. DBSCAN Algoritması ile Hatalı Özellik Eşleşmelerinin Bulunması

NTNİ uygulamasında özellik eşleştirme işlemi özellik çıkarım algoritması tarafından gerçekleştirilir. Nesne pozisyonunun doğru tahmin edilebilmesi için, özellik eşleştirmenin doğruluğu kritik bir öneme sahiptir. Fakat özellikleri doğru bir şekilde bulan algoritma halen gerçekleştirilememiştir. Bu durumda hatalı eşleştirmelerin bulunması bir başka önemli konu haline gelmektedir.

Özellik bulma algoritması ile özellikler bulunduğu zaman, genel doğru bulunan özelliklerin yanlış bulunan özelliklerden sayıca fazla olduğu gözlemlenmektedir. Ayrıca, yanlış özellik noktalarının nesneden uzak bölgelerde dağınık bir şekilde bulunduğu görülmektedir. Dolayısıyla, burada hatalı eşleştirmelerin bulunması problemi, koordinat sisteminde ayrık noktaların bulunması şeklinde düşünülebilir.

İstatistiksel bir sınıflandırma yöntemi olan DBScan [145] algoritması bu hatalı(ayrık) noktaların bulunması amaçlı kullanılmıştır. DBScan algoritması yoğunluk tabanlı bir sınıflandırma algoritmasıdır. Farklı şekillerde ve farklı eleman sayısı içeren kümeleri kolaylıkla sınıflandırabilmektedir. Ayrıca kullanımı ve gerçekleştirimi kolaydır. Aşağıda DBScan algoritmasının genel çalışma prensibi verilmiş ve Şekil 4.2'de gösterilmiştir [145].

- Veri setinde istenilen eleman ile başlanır. (Şekil 4.2, kırmızı nokta)
- Bu nokta ile diğer noktalar arasındaki uzaklıklar hesaplanır. Hesaplanan uzaklıklar eşik değerinden(eps) küçükse ilgili nokta kümeye dâhil edilir. (Şekil 4.2, mavi noktalar) Bu işlem her kümeye dahil edilen eleman için tekrarlanır. Bu işlem için Derin Öncelikli Arama algoritması kullanılır.
- Eğer taranmayan noktalar varsa, Yukarıdaki işlemler her taranmayan nokta için tekrarlanır. (Şekil 4.2, yeşil noktalar)
- Kümeleme işlemi bittiğinde kümeler ve kümelerin içindeki elemanlar belirlenmiş olur. Eğer küme eleman sayısı belirlenen minimum eleman sayısından(MinPts) az ise o kümenin elemanları gürültü olarak işaretlenir. Bu işlem tüm kümeler için tekrarlanır. (Şekil 4.2, kalan siyah nokta)



Şekil 4.2. DBScan algoritmasının genel çalışma yapısı

Bu projede, DBScan algoritması hatalı özellik eşleşmelerini bulmak için kullanılmıştır. DBScan algoritmasında eps ve MinPts parametreleri kullanılmaktadır. Bu iki parametre klasik DBScan algoritmasından farklı seçilmiştir. Normalde eps değeri noktaların birbirine uzaklıklarının ortalaması olarak hesaplanırken, biz bu projede sabit bir değer belirledik. Belirlenen parametreler eps değeri için 50 olarak alınmıştır. Verilen değer, iki özellik noktası arasındaki Öklid uzaklığından büyük veya küçük olmasına göre işlem yapılmaktadır. $A(x_1, y_1)$ ve $B(x_2, y_2)$ noktası arasındaki Öklid uzaklığı Denklem 4.5 ile hesaplanmaktadır.

$$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.5}$$

DBScan algoritması ile gürültü tespit algoritmasının sözde kodu Algoritma-1'de verilmiştir. Algortima-1'de görüldüğü gibi MinPts parametresi kümelerdeki maksimum eleman sayısına göre değişmektedir. Algoritma-2'de ise DFS yordamının sözde kodu verilmiştir. "eps" parametresi bu algoritmada kullanılmakta olup sınır içerisinde kalan noktaları aynı kümeye almayı sağlar. "uzaklikBul" yordamında ise iki nokta arasındaki uzaklık Denklem 4.5 kullanılarak bulunmaktadır. Ayrıca nesne izleme uygulamasında DBScan algoritmasının çalışma prensibi Şekil 4.3.'de gösterilmiştir. Şeklin sol tarafında turkuaz renk ile bulunan tüm eşleştirmeler gösterilmiştir. Şeklin sağ tarafında ise DBScan algoritması sonucu gürültü olarak bulunan noktalar kırmızı renk ile diğer doğru noktalar ise yeşil renk ile gösterilmiştir. Toplam bulunan 15 eşleştirmeden hatalı olan 5 eşleştirme başarılı bir şekilde bulunmuş nesne konum hesabından çıkarılmıştır. Böylece nesne tespit işleminin doğruluğu önemli derecede iyileştirilmiştir.

Algoritma 1 DBScan Algoritması ile Gürültü Tespiti

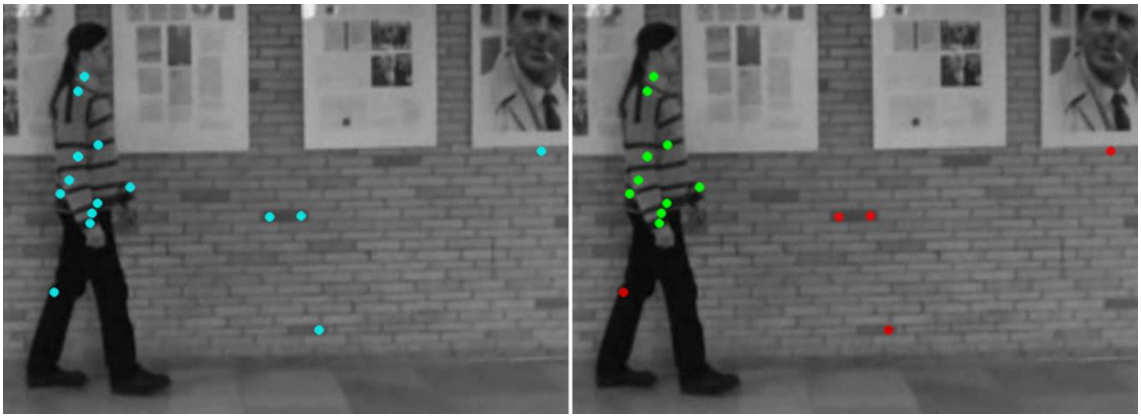
Giriş: kp= özellik noktası, mt= eşleşmeler
Çıkış: kp= doğru özellik noktası mt: eşleşme sonuçları

```
1:  for all keypoints in dataset
2:      kume_no = 0
3:      read next_point in keypoints
4:      if (next_point not visited)
5:          DFS (next_point, kume_no)
6:      endIf
7:      kume_no = kume_no + 1
8:  endFor
9:
10: max = kümelerdeki maksimum eleman sayısı
11:
12: if (1<max<6) minPts = 1
13: else if (6<max<12) minPts = 2
14: else if (12<max<18) minPts = 3
15: else if (18<max<24) minPts = 4
16: else minPts = 5
17: minPts'den kücük eleman sayısına sahip kümeleri sil
```

Algoritma 2 DFS yordamı

Giriş: gelen_nokta, kume_no

```
1:  gelen_nokta gezildi olarak işaretle
2:  for all keypoints in dataset
3:      read sonraki_nokta in keypoints
4:      uzaklik = uzaklikBul(gelen_nokta, sonraki_nokta)
5:      if (uzaklik <eps ve nokta gezilmediyse)
6:          DFS (sonraki_nokta, sayac)
7:      end
8:          gelen_nokta_kume_no = kume_no
9:      endIf
10: endFor
```



Şekil 4.3. DBScan algoritmasının SIFT algoritması ile bulunan özellikler üzerinde örnek çalışma sonucu. SIFT yöntemi ile bulunan özellikler turkuaz renk ile gösterilmiştir. DBScan algoritması sonucu doğru özellikler yeşil renk ile hatalı özellikler kırmızı renk ile gösterilmiştir.

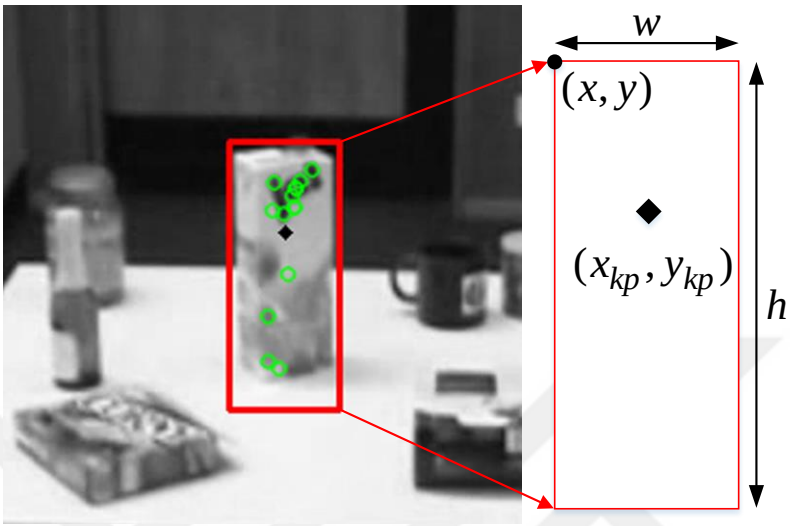
4.3.2. Nesne Modeli

Hatalı eşleştirmeler temizlendikten sonra, nesne pozisyonu ortalama olarak tahmin edilebilir. Nesne pozisyonunun tahmin etmek için nesne ile video çerçevesi arasında homografi hesabının yapılması gerekmektedir. Homografi, bir görüntüdeki özellik noktalarını diğer bir görüntüdeki özellik noktalarına eşleyen 3x3'lük bir matris dönüşümüdür [146]. Fakat homografinin doğru bir şekilde hesaplanabilmesi için en az 4 eşleşen noktaya ihtiyaç duymaktadır. Homografi hesabı için DLT (Direct Linear Transformation) [147] ve RANSAC (Random Sample Consensus) [148] algoritmaları en yaygın kullanılan algoritmalardır. RANSAC algoritması Fischler ve Bolles tarafından 1981 yılında geliştirilmemiştir. Bu algoritma ile imge modelleri sağlan bir şekilde yerine oturmaktadır.

RANSAC ile yapılan deneylerde eşleşen özellik sayısının çok olduğu durumlarda hatalı eşleştirmelere rağmen başarılı sonuçlar elde edilmektedir. Fakat özellikle eşleşen özellik sayısı 10 ve altına düştüğünde algoritmanın gürültülere duyarlılığı artmaktadır. Dolayısıyla gerçekleştirilen uygulamada en az 10 eşleşen özellik noktası aranmaktadır. Yeterli sayıda eşleşen özellik bulunursa, uygulama RANSAC sonuçlarının doğruluğunu kontrol edip direk kullanmaktadır. Fakat yeterli sayıda eşleşen özellik bulunamaz ve RANSAC hesaplaması yapılmaz ise bölümün geri kalanında anlatılacak algoritma kullanılır.

Bu çalışmada nesne kutu gösterimi ile temsil edilmektedir. Şekil 4.4'te görüldüğü gibi nesne modeli 6 değer ile temsil edilmektedir. Eşleşen özelliklerin hesaplanması sonucu özellik noktaları ortaya çıkmaktadır. x_{kp} ve y_{kp} değerleri özellik noktalarının x ve y koordinat bilgilerinin ortalaması ile, x, y, w ve h değerleri ise eşleşen özelliklerin

kullanılması sonucu hesaplanan homograpi ile elde edilmektedir. Kısaca homografi iki düzlem arasındaki dönüşümü ifade etmektedir. Homografi matrisi 3x3'lük bir matristir ve 8 serbestlik derecesine sahiptir.



Şekil 4.4. 6 değerli nesne modeli

Bu 6 değer için Gauss modeli kullanılmıştır. Her bir nokta kendi modeline sahiptir. Nesne modelini geçmişteki t nesne modeli kullanılarak başlatılmaktadır. Daha sonra, t zamanındaki olasılık fonksiyonu Denklem 4.6 ile hesaplanmaktadır.

$$P(X_t) = w_{i,t} n(X_t, \mu_i, \sum_i) \tag{4.6}$$

X_t : t zamanındaki nesne model değeri

$w_{i,t}$: t zamanındaki dağılımın ağırlık değeri

μ_i : Dağılımın ortalaması

$\sum_i$: Dağılımın standart sapması

Olasılık yoğunluk fonksiyonu $n(X_t, \mu_i, \sum_i)$, Denklem 4.7 ile hesaplanır.

$$n(X_t, \mu_i, \sum_i) = \frac{1}{\sqrt{(2\pi}\,.\,\sigma} e^{\frac{-X_t^2}{2\sigma^2}} \tag{4.7}$$

Burada σ standart sapma değerini ifade etmektedir. Tüm model başlatıldıktan sonra0 model, nesne tahmini veya nesne hesaplaması şeklinde ayrılmaktadır. x, y, w ve h değerleri

bir sonraki işlemin seçilmesi için kullanılır. Bir sonraki işlemin seçilmesi, Denklem 4.8 ile sağlanmaktadır.

$$n = \sum_{k=1}^{g} \begin{Bmatrix} 1 & eğer\ w_k > T \\ 0 & diğer\ durumlarda \end{Bmatrix} \tag{4.8}$$

Burada g Gaussian modellerini, T ise eşik değerini göstermektedir. Burada Gauss değerleri T eşik değerinden büyük olup olmadığı kontrol edilmektedir. Her bir Gauss modeli için eşik değerini geçen modeller sonucu 1 arttırır. Sonuç olarak, eğer $n > 2$ koşulu sağlanıyorsa nesne hesaplama işlemi uygulanmaktadır. Nesne hesaplama işleminde eğer Gauss modeli sonucu eşik değerini ulaşamayan modeller varsa, değer üretme işlemi uygulanır. Böylece nesne modeli için 4 değer de elde edilmiş olur. Eğer $n > 2$ koşulu sağlanmıyorsa nesne tahmini işlemi uygulanmaktadır. Nesne tahmini işleminde x_{kp} ve y_{kp} değerleri Denklem 4.8 ile kontrol edilmektedir. Eğer her iki model de eşik değerini geçerse, nesne merkezi x_{kp} ve y_{kp} değerleri olarak atanır. Eğer en az birisi eşik değerini geçemezse Gauss modeli sonucu oluşan değer nesne merkezi olarak kullanılmaktadır. Son olarak, nesne için w ve h değerleri Gauss dağılımı yardımıyla hesaplanır ve nesnenin x ve y değerleri Denklem 4.9 ve Denklem 4.10 ile hesaplanmaktadır.

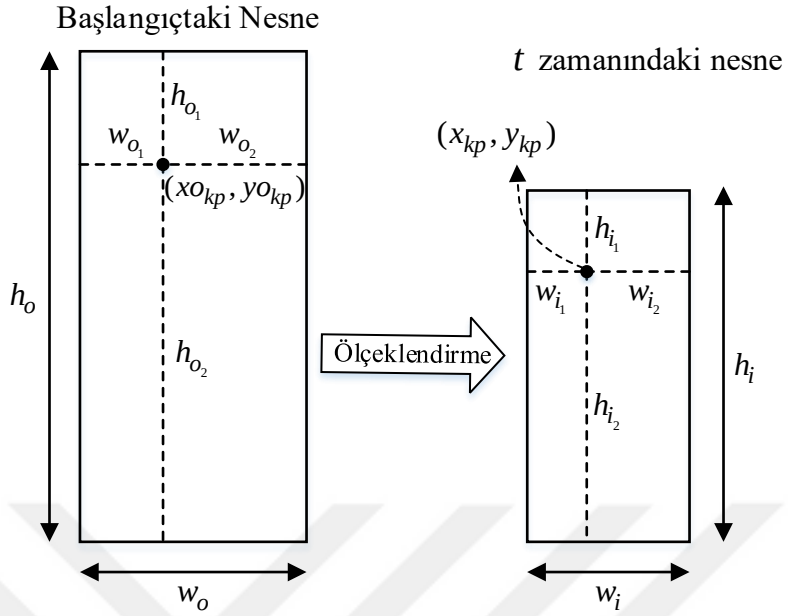
$$x = xi_{kp} - w_i . \frac{xo_{kp}}{w_o} \tag{4.9}$$

$$y = yi_{kp} - h_i . \frac{yo_{kp}}{h_o} \tag{4.10}$$

Nesne boyutu video içerisinde değişebileceği için Denklem 4.9 ve 4.10'da verilen formüller ile ölçeklendirme işlemi uygulanmaktadır. Dolayısıyla, Denklem 4.9 ve 4.10 ifadeleri Denklem 4.11 denklemini sağlar.

$$\frac{w_{o_1}}{w_{o_2}} \equiv \frac{w_{i_1}}{w_{i_2}} \quad and \quad \frac{h_{o_1}}{h_{o_2}} \equiv \frac{h_{i_1}}{h_{i_2}} \tag{4.11}$$

Denklemlerde geçen tüm parametreler Şekil 4.5'te verilmiştir. Şekil 4.5'te ve denklemlerde görüldüğü gibi nesne merkez noktası kullanılarak ölçeklendirme işlemi yapılmaktadır.



Şekil 4.5. Başlangıç nesnesi ve video çerçevesinde değişen nesne için nense modeli ve parametreleri

Tüm bu işlemler tamamlandıktan sonra, eğer nesne bulunduysa nesne izleme adımına geçilmektedir. Eğer nesne bulunamadıysa bir sonraki video çerçevesi için hesaplamalar tekrar yapılmaktadır.

4.3.3. Nesne İzleme

NTNİ uygulamasının son adımı, nesne izleme adımıdır. Bu kısımda 4.3.2 nolu kısımda elde edilen x, y, w ve h değerleri kullanılarak nesne kutu biçiminde işaretlenmiştir. NTNİ uygulaması bazı durumlarda kısa süreli nesne konumunu hatalı tahmin etmektedir. Bu durum genellikle geçici olmakta ve bir sonraki video çerçevesinde düzeltmektedir. Bu durum genel olarak kısım 4.3.1'de bahsedilen hatalı özellik noktalarından kaynaklanmaktadır. Kısım 4.3.1'de birçok hatalı özellik noktası elenmesine rağmen, bazı video çerçevelerinde halen hatalı noktalar olabilmektedir. Bu hatalı durumdan kurtulmak için Gauss yumuşatma işlemi uygulanmıştır. Böylece oluşması olası hata minimize edilmiş olur.

Gauss yumuşatma işlemi nesne komun bilgisi olan x, y, w ve h değerleri için uygulanmıştır. Bir önceki bölümde Gauss işlemi ile nesnenin geçmiş konumlarına göre nesne konumu tahmin edilip, hatalı konum bilgileri düzeltilmeye çalışılmıştır. Bu bölümde

ise gelen konum bilgileri doğru olarak varsayılmış, sadece son n video çerçeve bilgisi ile nesne konumu güncellenmiştir.

Günümüzde videolar genel olarak saniyede 30 video çerçevesi oynanmaktadır. Daha büyük veya daha küçük değerler seçilebilmesine rağmen insan gözünün durağanlığı anlamaması için saniyede en az 24 video çerçevesi gösterilmesi gerekmektedir. Bu bilgiler ışığında yapılan uygulama için n değeri 5 olarak seçilmiştir. 5 video çerçevesi yaklaşık 0,16 saniyeye denk gelmektedir ki bu kadar kısa süre içerisinde nesnenin çok büyük bir değişiminin olmaması gerekir.

Gauss yumuşatma işlemi ile son 5 video çerçevesindeki nesne konumuna göre nesnenin komunu belirlenmektedir. Yüzdesel olarak video çerçevelerinin ağırlığına bakacak olursak en son video çerçevesi %44 değere sahipken, diğer video çerçeveleri sırasıyla %34, %16, %5 ve %1 ağırlığa sahiptir. Dolayısıyla bir video çerçevesinde oluşan hata, ilk oluştuğu anda %56 oranında minimize edilmiştir. Daha sonraki video çerçevelerinde ise bu hata minimize oranı hızla %100 oranına yaklaşmaktadır. Tabi ki bu algoritmada akla gelen ilk problem nesnenin ani değişimlerinde bu algoritmanın nasıl cevap vereceğidir. 5 video çerçevesi çok kısa süre almasına rağmen, özellikle ani hareketin ve yön değişiminin olduğu durumlarda algoritmanın nesneyi geriden takip etmesi olasıdır fakat tüm video boyunca ani hareket olamayacağı için nesne çok kısa bir sürede yakalanır ve nesne takibi tekrar başarılı bir şekilde devam eder. Burada önemli olan konu nesne takibinde çok büyük kayıpları engellemek ve nispeten daha küçük kayıp vererek nesne takibini gerçekleştirmektir. Yapılan deneyler sonucunda sadece bu algoritma ile %2 ile %10 arasında nesne izleme başarımının arttığı görülmüştür.

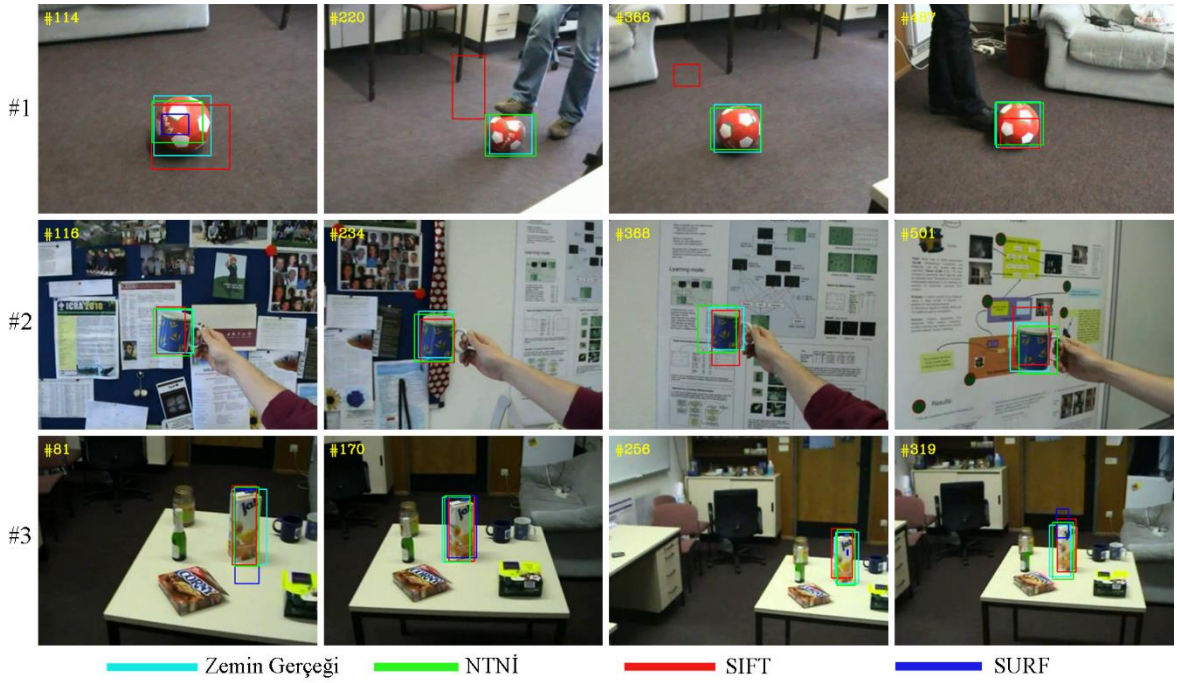
4.4. Uygulama Sonuçları

Bu bölümde geliştirilen uygulama BoBoT veri seti kullanılarak test edilmiştir [139]. BoBoT ver setinin tanıtımı kısım 3.3.2 de verilmiştir. Orijinal veri setinde video çözünürlüğü 320x240 boyutundadır. Fakat bazı videolarda çözünürlük çok düşük olduğundan nesne boyutu çok küçük kalmaktadır. Bu bölümde hem iş yükünü arttırmak hem de nesne boyutunu büyütmek için video çözünürlüğü 640x480 boyutuna dönüştürülmüştür. BoBoT video veri setinde birçok özellik test edilmektedir. Bu özelliklere ait açıklamalar yine kısım 3.3.2 ve Tablo 3.2'de verilmiştir.

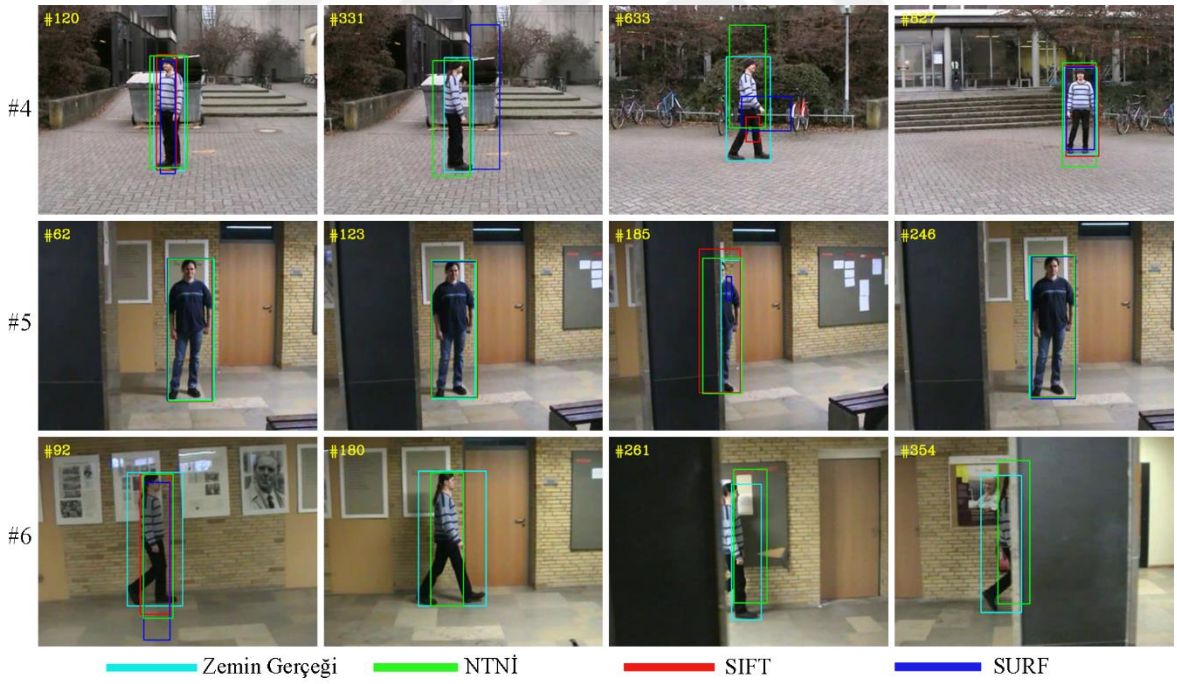
Geliştirilen uygulama veri setindeki videolar ile test edilmiştir. Geliştirilen nesne izleme uygulaması SIFT, SURF ve GPU-SURF yöntemleri ile entegre edilmiştir. Daha sonra standart SIFT, SURF ve GPU-SURF yöntemlerinin nesne izleme uygulaması ile geliştirilen uygulamaya entegre edilmiş versiyonları karşılaştırılmıştır. SURF ve GPU-SURF yöntemlerinde aynı uygulama farklı mimarilerde koşulduğu için nesne izleme başarım sonuçları benzerdir. Bu iki algoritmada temel fark SURF yönteminin merkezi işlemci biriminde çalıştırılması ve GPU-SURF yönteminin ise grafik işlemci biriminde çalıştırılmasıdır. Dolayısıyla GPU-SURF yöntemi daha hızlı çalışmaktadır.

Geliştirilen NTNİ uygulaması ile SIFT, SURF ve GPU-SURF yöntemlerinin karşılaştırma sonuçları Şekil 4.6, Şekil 4.7 ve Şekil 4.8'de verilmiştir. NTNİ uygulaması ile SIFT algoritmasının entegre edilmiş hali şekillerde NTNİ olarak isimlendirilmiştir. Şeklin sol tarafında hangi videonun test edildiği belirtilmiştir. Ayrıca her videoda hangi video çerçevesinin alındığı imgelerin sol üst köşesinde belirtilmiştir. Şekillerde her algoritma çalışma sonuçları farklı bir renk ile gösterilmiştir. Önerilen NTNİ uygulaması yeşil renk ile SIFT yöntemi kırmızı renk ile SURF yöntemi mavi renk ile son olarak zemin gerçeği sonucu turkuaz renk ile işaretlenmiştir. Geliştirilen NTNİ uygulaması veri seti üzerinde farklı senaryolar ile test edilmiş ve genel olarak tatmin edici sonuçlar elde edilmiştir. Şekillerden görüldüğü gibi NTNİ uygulaması diğer algoritmalara göre daha başarılı sonuçlar elde etmiştir. Bazı durumlarda SIFT ve SURF algoritmaları nesne izleme uygulaması için sonuç üretememektedir. Bu durumda bazen nesne konumu bulunamamış bazen ise nesne konumu yanlış belirtilmiştir. Örneğin ikinci videoda SURF algoritması genel olarak sonuç üretmemiştir. Benzer şekilde altıncı videoda SIFT ve SURF algoritmaları yine uygun sonuçlar elde edememiştir.

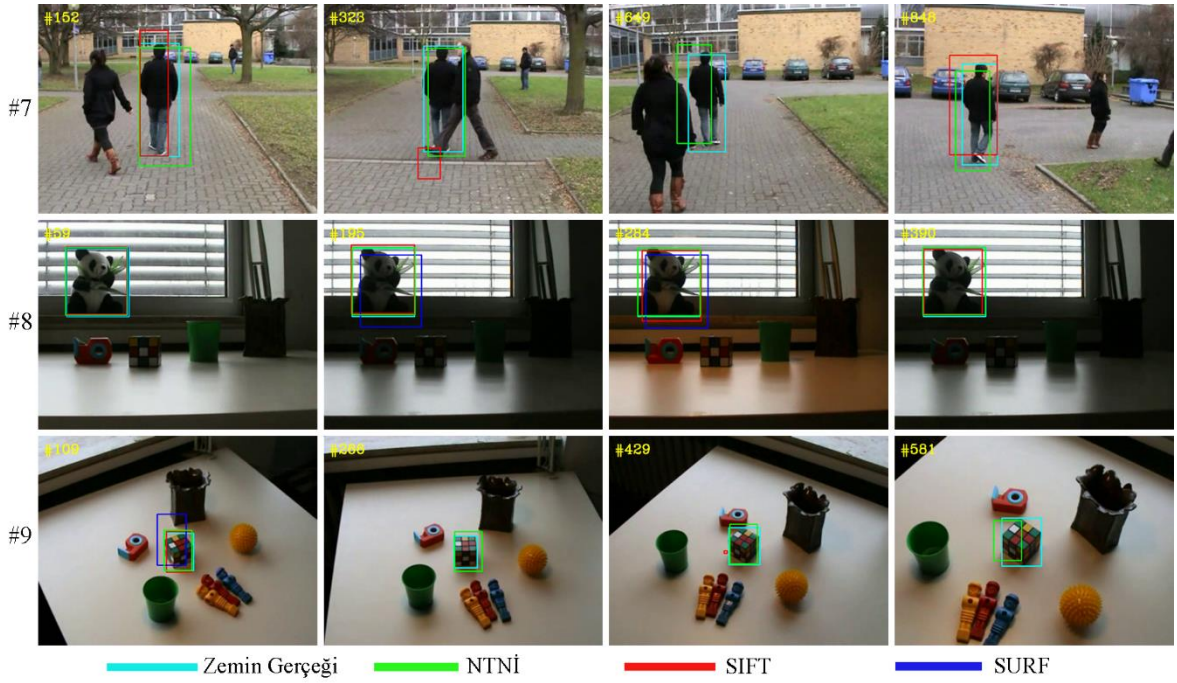
Bölüm 3'te bahsedildiği gibi SIFT yöntemi ile SURF ve GPU-SURF yöntemlerine göre daha fazla özellik bulunabilmektedir. Bu nedenle, geliştirilen nesne izleme uygulaması SIFT yöntemi ile daha başarılı sonuçlar vermektedir. Nesne izleme uygulamalarının başarımını ölçmek için en temel yöntem nesne zemin gerçeği ile bulunan nesne konumunun karşılaştırmasıdır. Bu amaçla HA, KE, FS, LoU, TPS, AP ve mAP ölçütleri uygulama sonuçlarını değerlendirmek amacıyla kullanılmıştır. Ölçütler ile ilgili hesaplamalara kısım 3.3.2'de detaylı olarak açıklanmıştır. Ayrıca HA ölçütü Denklem 3.4'te, KS ölçütü Denklem 3.3'te, FS ölçütü Denklem 3.5'te, LoU ölçütü Denklem 3.2'de ve TPS ölçütü Denklem 3.6'da verilmiştir.



Şekil 4.6. 1., 2. ve 3. videolar için algoritmaların nesne izleme sonuçları



Şekil 4.7. 4., 5. ve 6. videolar için algoritmaların nesne izleme sonuçları



Şekil 4.8. 7., 8. ve 9. videolar için algoritmaların nesne izleme sonuçları

Geliştirilen uygulama SIFT yöntemi ve varyantları için uygundur. Ayrıca diğer nokta tabanlı özellik bulan algoritmalar da dâhil edilebilir. Özellik tabanlı SIFT ve varyantları nesne görünümünün dramatik bir şekilde değiştiği durumlarda başarılı olamamaktadır. Fakat geliştirilen uygulama ile nesne izleme başarım oranı önemli derecede iyileştirilmiştir. Tablo 4.2'de verilen sonuçlarda görüldüğü gibi 1., 4. ve 7. videolarda SIFT ve varyantlarının başarım oranı düşüktür. Geliştirilen uygulama ile bu tür videolarda başarı oranları büyük oranda arttırılmıştır. Ayrıca diğer videolarda da büyük artışlar görülmektedir. Sadece 5. ve 8. videolarda zaten başarı oranları standart algoritmalarda da yüksek olduğu için başarı oranları çok fazla artmamıştır.

Nesne izleme uygulaması için geliştirilen uygulamanın klasik SIFT yöntemi ile karşılaştırma sonuçları Tablo 4.2'de verilmiştir. Sonuçlardan görüldüğü gibi geliştirilen uygulama ile tüm videolarda gelişim sağlanmıştır. Sadece 5. videoda çok küçük bir artış elde edilmiştir. Bunun nedeni ise zaten 5. videonun SIFT algoritmasında başarılı sonuçlar vermiş olmasıdır. Diğer videolarda ise başarı oranı ciddi derecede arttırılmıştır. Örneğin 3. video için LoU değeri SIFT yöntemi ile 0,63 değeri elde edilirken, SIFT+NTNİ yöntemi ile 0,85 değeri elde edilmiştir. Benzer şekilde 8. videoda mAP değeri SIFT yöntemi ile 0,7 değeri elde edilirken geliştirilen uygulama ile bu değer 0,96'ya ulaşmıştır. Ortalama sonuçlara bakacak olursak HA ölçütü 0,53'ten 0,81'e, KE ölçütü 0,6'dan 0,84'e, FS ölçütü 0,52'den

0,81'e çıkmıştır. Ayrıca LoU ölçütü 0,46'dan 0,72'ye, TPS ölçütü 0,5'ten 0,82'ye, AP ölçütü 0,7'den 0,91'e ve son olarak mAP ölçütü 0,34'ten 0,54 değerine ulaşmıştır.

Tablo 4.2. SIFT algoritmalarının NTNİ uygulaması ile karşılaştırmalı sonuçları

Video No	SIFT							SIFT + NTNİ						
	HA	KE	FS	LoU	TPS	AP	mAP	HA	KE	FS	LoU	TPS	AP	mAP
1	0,33	0,45	0,32	0,30	0,30	0,55	0,26	0,73	0,74	0,71	0,59	0,71	0,80	0,32
2	0,43	0,50	0,38	0,32	0,35	0,55	0,20	0,84	0,87	0,85	0,75	0,99	0,99	0,54
3	0,75	0,78	0,76	0,63	0,83	0,94	0,35	0,90	0,93	0,91	0,85	0,99	0,99	0,75
4	0,37	0,52	0,36	0,31	0,33	0,56	0,21	0,83	0,78	0,80	0,68	0,85	0,91	0,44
5	0,97	0,89	0,92	0,87	0,95	0,98	0,79	0,98	0,89	0,93	0,88	0,96	0,98	0,81
6	0,34	0,52	0,36	0,31	0,32	0,60	0,21	0,56	0,79	0,64	0,52	0,51	0,71	0,25
7	0,28	0,36	0,28	0,25	0,27	0,62	0,20	0,80	0,83	0,79	0,68	0,79	0,96	0,46
8	0,87	0,89	0,86	0,80	0,86	0,91	0,70	0,97	0,98	0,98	0,96	1,00	1,00	0,96
9	0,46	0,51	0,42	0,33	0,34	0,57	0,16	0,64	0,71	0,67	0,55	0,62	0,83	0,31
Ortalama	**0,53**	**0,60**	**0,52**	**0,46**	**0,50**	**0,70**	**0,34**	**0,81**	**0,84**	**0,81**	**0,72**	**0,82**	**0,91**	**0,54**

Nesne izleme uygulaması için geliştirilen uygulamanın klasik SURF yöntemi ile karşılaştırma sonuçları Tablo 4.3'te verilmiştir. SURF algoritmasının NTNİ yöntemi ile entegrasyonu sonucu başarı oranı ciddi derecede arttırılmıştır. Sadece 5. videoda SURF yönteminin başarı oranı zaten yüksek olduğu için başarı oranı çok fazla artmamıştır. Örneğin 5. video LoU ölçütü, 0,85'ten 0,88'e çıkmıştır. Diğer videolarda ise yüksek oranlı artışlar söz konusudur. Ortalama sonuçlara bakılacak olursa LoU ölçütü 0,41'den 0,66'ya, TPS ölçütü 0,46'dan 0,75'e ve mAP ölçütü ise 0,31'den 0,47'ye yükseltilmiştir.

Tablo 4.3. SURF algoritmalarının NTNİ uygulaması ile karşılaştırmalı sonuçları

	SURF							SURF + NTNİ						
Video No	**HA**	**KE**	**FS**	**LoU**	**TPS**	**AP**	**mAP**	**HA**	**KE**	**FS**	**LoU**	**TPS**	**AP**	**mAP**
1	0,26	0,32	0,26	0,25	0,26	0,51	0,24	0,52	0,69	0,57	0,46	0,45	0,62	0,20
2	0,36	0,44	0,33	0,28	0,32	0,54	0,19	0,86	0,86	0,85	0,75	0,98	0,99	0,54
3	0,70	0,72	0,69	0,57	0,73	0,90	0,31	0,85	0,90	0,87	0,80	0,93	0,98	0,67
4	0,30	0,44	0,29	0,25	0,27	0,49	0,18	0,70	0,62	0,65	0,54	0,56	0,74	0,30
5	0,94	0,88	0,90	0,85	0,94	0,98	0,78	0,98	0,89	0,93	0,88	0,95	0,98	0,81
6	0,34	0,51	0,35	0,30	0,31	0,63	0,20	0,56	0,77	0,63	0,52	0,51	0,77	0,26
7	0,26	0,27	0,26	0,23	0,27	0,62	0,20	0,72	0,81	0,74	0,61	0,68	0,88	0,33
8	0,78	0,80	0,76	0,70	0,82	0,90	0,58	0,93	0,95	0,94	0,88	1,00	1,00	0,82
9	0,35	0,46	0,30	0,24	0,22	0,43	0,12	0,70	0,57	0,63	0,53	0,67	0,87	0,33
Ortalama	**0,48**	**0,54**	**0,46**	**0,41**	**0,46**	**0,66**	**0,31**	**0,76**	**0,78**	**0,76**	**0,66**	**0,75**	**0,87**	**0,47**

Nesne izleme uygulaması için geliştirilen uygulamanın klasik GPU-SURF yöntemi ile karşılaştırma sonuçları Tablo 4.4'te verilmiştir. NTNİ yönteminin GPU-SURF yöntemi ile entegre edilmesi sonucu aynen SIFT ve SURF yöntemleri gibi GPU-SURF yönteminin de başarı oranlarını ciddi derecede geliştirmiştir. Ortalama sonuçlarda ise LoU değeri 0,38'den 0.66'ya ve mAP değeri ise 0.29'dan 0.44'e yükseltilmiştir.

Tablo 4.4. GPU-SURF algoritmalarının NTNİ uygulaması ile karşılaştırmalı sonuçları

	GPU-SURF							**GPU-SURF + IPBOT**						
Video No	**HA**	**KE**	**FS**	**LoU**	**TPS**	**AP**	**mAP**	**HA**	**KE**	**FS**	**LoU**	**TPS**	**AP**	**mAP**
1	0,27	0,33	0,27	0,26	0,26	0,52	0,25	0,53	0,73	0,60	0,47	0,45	0,57	0,22
2	0,30	0,38	0,27	0,24	0,26	0,48	0,16	0,85	0,80	0,82	0,72	0,91	0,93	0,50
3	0,64	0,70	0,63	0,53	0,68	0,88	0,29	0,83	0,89	0,86	0,78	0,90	0,96	0,64
4	0,31	0,44	0,29	0,25	0,27	0,49	0,17	0,68	0,64	0,65	0,54	0,60	0,70	0,31
5	0,94	0,88	0,89	0,85	0,93	0,97	0,78	0,97	0,89	0,92	0,87	0,93	0,97	0,80
6	0,34	0,49	0,34	0,29	0,33	0,61	0,20	0,59	0,84	0,68	0,56	0,59	0,72	0,24
7	0,26	0,29	0,25	0,23	0,26	0,61	0,20	0,73	0,80	0,74	0,61	0,67	0,88	0,31
8	0,71	0,70	0,68	0,60	0,69	0,83	0,44	0,94	0,93	0,94	0,89	1,00	1,00	0,80
9	0,31	0,46	0,27	0,21	0,19	0,39	0,11	0,80	0,51	0,61	0,48	0,62	0,81	0,18
Ortalama	**0,45**	**0,52**	**0,43**	**0,38**	**0,43**	**0,64**	**0,29**	**0,77**	**0,78**	**0,76**	**0,66**	**0,74**	**0,84**	**0,44**

Sonuçları genel olarak değerlendirirsek, SURF ve GPU-SURF algoritmasında başarı oranları benzer çıkmıştır. Klasik versiyonlarda SURF yöntemi GPU-SURF yönteminden yaklaşık %5-8 oranında daha başarılı iken, NTNİ yönteminin iki algoritmaya entegre edilmesinden sonra bu fark ortadan kalkmıştır. Örneğin SURF yönteminde LoU ölçütü 0,46 iken, GPU-SURF yönteminde bu oran 0,43 değerindedir. Fakat NTNİ algoritmasının uygulanması ile LoU ölçütü her iki algoritmada da 0,66 olarak hesaplanmıştır. Benzer şekilde TPS ölçütü için SURF yöntemi ile 0,46, GPU-SURF yöntemi ile ise 0,43 elde edilmiştir. NTNİ yönteminin uygulanması sonucu SURF yöntemi 0,75 değerine ulaşırken, GPU-SURF yöntemi 0,74 değerine ulaşmıştır.

SURF ve GPU-SURF algoritmaları benzer sonuçlar elde etmesine rağmen, GPU-SURF algoritması SURF algoritmasına göre daha hızlı çalışmaktadır. Tablo 4.5'te uygulamaların çalışma hızları saniyede işlenen video çerçeve sayısı (fps) olarak karşılaştırılmıştır. Ayrıca parantez içinde algoritmaların standart algoritmalara göre gelişimi veya yavaşlaması yüzde olarak verilmiştir. Tabloda görüldüğü gibi en yavaş algoritma SIFT algoritması iken, en hızlı çalışan algoritma ise GPU-SURF algoritmasıdır. Sonuçlardan görüldüğü gibi çalışma

zamanı bakımından standart algoritmalar ile benzer sonuçlar elde edilmiştir. Ayrıca, GPU-SURF + NTNİ uygulaması ortalama 39.15 fps oranı elde edilmiştir. Bu ise algoritmanın gerçek zamanlı uygulamalar için uygun olduğunu göstermektedir.

Tablo 4.5. SIFT, SURF, GPU-SURF algoritmalarının NTNİ uygulaması ile fps türünden karşılaştırmalı performans sonuçları

Video No	SIFT	SURF	GPU SURF	SIFT + NTNİ	SURF + NTNİ	GPU SURF + NTNİ
1	9,06	18,82	48,85	9,04 (%-0,22)	18,38 (%-2,36)	48,13 (%-1,48)
2	6,31	12,06	37,83	6,27 (%-0,67)	11,83 (%-1,91)	37,31 (%-1,37)
3	7,73	14,23	42,70	7,71 (%-0,21)	14,15 (%-0,57)	41,29 (%-3,31)
4	4,58	9,87	26,97	4,63 (%1,2)	9,96 (%0,92)	29,07 (%7,78)
5	6,85	13,17	37,60	6,77 (%-1,14)	13,16 (%-0,04)	36,52 (%-2,88)
6	7,36	12,66	34,39	7,24 (%-1,67)	12,76 (%0,77)	36,28 (%5,49)
7	8,83	18,43	51,74	8,62 (%-2,35)	18,16 (%-1,47)	51,02 (%-1,39)
8	8,01	14,83	39,17	7,91 (%-1,24)	14,82 (%-0,09)	41,36 (%5,58)
9	4,50	10,90	29,80	4,52 (%0,34)	10,99 (%0,79)	31,35 (%5,21)
Ortalama	7,03	13,89	38,78	6,97 (%-0,82)	13,8 (%-0,62)	39,15 (%0,94)

4.5. Sonuç

Bu bölümdeki amaç, nokta tabanlı özellik çıkarım algoritmaları için hatalı özellik noktalarının temizlenmesi ile yeni bir nesne izleme yapısının oluşturulmasıdır. Nokta tabanlı özellik çıkarım algoritmaları, nesnedeki birçok özellik noktasını başarılı bir şekilde çıkartabilmektedir. Fakat literatüre baktığımızda her bir algoritmanın avantajlı olduğu durumlar ve dezavantajlı olduğu durumlar bulunmaktadır. Önerilen bu nesne izleme yapısı ile birlikte, tüm bu algoritmalar nesne izleme uygulamaları için kullanılabilecektir.

Nokta tabanlı özellik çıkarım algoritmaları için özellik çıkarımı yapıldıktan sonra sırasıyla hatalı özellik tespiti, nesne modelleme ve nesne izleme adımları gerçekleştirilmektedir. Hatalı özellik tespit işlemi için DBScan algoritmasından yararlanılmıştır. DBScan algoritması ile nesne ve video çerçevesi arasında gerçekleşen özellik eşleştirme işlemi sonucu açığa çıkan hatalı eşleştirmeleri minimize etmesi amaçlanmaktadır. Böylece eşleşen özelik sayısı az olsa bile, nesne konumu doğru bir şekilde tahmin edilebilmektedir. Bir diğer gelişim ise nesne modelleme aşamasında yapılmıştır. Nesne 6 değer ile modellenmiş ve her bir değer için Gauss modeli uygulanmıştır. Normalde, nesne tespiti eşleşen özellik noktalarına dayanarak yapılmaktadır, fakat burada doğru nesne

tespiti için eşleşen özellik sayısı ve doğruluğu büyük önem arz etmektedir. Eğer sadece birkaç tane özellik eşleştirmesi varsa bu özelliklerin doğru olması gerekir. Yanlış eşleşmeler sadece birçok özellik noktasının olduğu durumlarda kabul edilebilir. Geliştirilen uygulama ile nesne konumu özellik noktasının az ve hatalı olduğu durumlarda bile nesne pozisyonu doğru bir şekilde tahmin edilebilmektedir. Bu başarılı tahmin DBScan algoritması ve 6 değerli nesne modeli ile gerçekleşmektedir.

NTNİ uygulaması SIFT, SURF ve GPU-SURF yöntemleri ile adapte edilmiş ve bu adapte edilmiş yöntemler standart SIFT, SURF ve GPU-SURF yöntemleri ile karşılaştırılmıştır. Sonuçlara göre NTNİ uygulaması çalışma hızı bakımından standart algoritmalar ile benzer sonuçlar elde etmiştir. Çalışma zamanı bakımından önemli bir fark olmamasına rağmen, NTNİ uygulaması ile nesne izleme başarımı yaklaşık %50 artmıştır.

5. SONUÇ

Nesne izleme uygulamaları günümüzde birçok alanda yaygın olarak kullanılmaktadır. Fakat yaygın kullanılmasına rağmen halen %100 doğruluk ile çalışan bir sistem geliştirilememiştir. Günümüzde nesne izleme uygulamaları ancak ışık değişimi, nesne görünümü ve hareketi gibi bazı belirli kısıtlar karşılandıktan sonra ancak tutarlı çalışmaktadır. Geliştirilen bir nesne izleme sistemi daha önce tanımlanmayan bir durumla karşılaştığında nesne izleme performansı düşebilir hatta nesne kaybedilebilir. Bu durumda sistemin nesne konumunu tahmin etmesi gerekebilir.

Nesne izleme sistemi genel olarak üç temel adımdan oluşmaktadır. Bu adımlar nesne tespiti, nesnenin modellenmesi ve nesne izlenmesidir. Nesne tespiti için nokta tabanlı özellik tespit yöntemleri en yaygın kullanılan yöntemlerdedir. Ayrıca nokta tabanlı nesne özellik tespit yöntemleri nesne izleme uygulamalarında sıkça kullanılmaktadır.

Nesne tespitinde önemli işlemlerden biri nesnenin sahnede olup olmadığı ve nesnenin konumunun doğru bir şekilde tespit edilerek izlenebilmesidir. Nokta tabanlı özellik tespit yöntemlerinden olan SIFT ve SURF yönteminden esinlenerek geliştirilen yöntemler bir nesnenin sahnede olup olmadığını ve nesnenin konumu hakkında bilgi verebilmektedir. Fakat video çerçevelerinde nesne görünüm değişikliği olabileceğinden dolayı başarılı sonuç vermemektedir.

Tez çalışmasında ilk olarak SIFT, SURF ve GPU-SURF yöntemleri ile bulunan ve eşleşen özellik sayısı bakımından test edilmiştir. Tezin bu aşamasında ETH Zurich (ETHZ) nesne veri seti kullanılmıştır. Ayrıca algoritmaların BoBoT veri seti kullanılarak izleme performansları değerlendirilmiştir. Yapılan testler sonucunda, SIFT algoritması diğer yöntemlere göre özellik bulma ve özellik eşleştirme uygulamalarında daha başarılı sonuçlar elde etmektedir. Ancak, çalışma hızı bakımından diğer yöntemlere göre oldukça yavaştır ve nesne izleme başarısı yeterli değildir. GPU-SURF yöntemi ise özellik çıkartma uygulamasında SURF algoritması ile benzer sonuçlar elde etmesinin yanı sıra en hızlı çalışan yöntemdir. Nesne izleme performansı olarak ise en iyi yöntem SIFT olmasına rağmen yeterli başarı oranı sağlanamamıştır. Sonuç olarak, algoritmaların nesne eşleştirme için ideal olduğu fakat nesne izleme uygulamaları için çok da başarılı olmadığı görülmüştür.

İkinci olarak tez çalışmasında geliştirilen algoritma ile SIFT ve SIFT yönteminden esinlenerek geliştirilen yöntemlerin nesne izleme uygulamalarında kullanılabilmesi

amaçlanmıştır. Geliştirilen sistem nesne özellik çıkarım algoritmasının yapısına karışmaz fakat üç adımda nesne izleme performansı sunmaktadır. Bu adımlar, hatalı özellik tespiti, nesne modelleme ve nesne izleme adımlarıdır. Hatalı özellik tespiti için istatistiksel bir yöntem olan DBScan algoritması kullanılmıştır. Nesne modelleme ve nesne izleme adımlarında ise Gauss yöntemi kullanılmıştır. Geliştirilen yöntem, SIFT, SURF ve GPU-SURF yöntemleri ile BoBoT veri seti üzerinde test edilmiş ve klasik algoritmalara göre oldukça başarılı sonuçlar elde edilmiştir. Ayrıca geliştirilen uygulama diğer nokta tabanlı özellik çıkarım algoritmaları için de uygundur. Kısa bir entegrasyon işleminden sonra kullanılması mümkündür.

6. KAYNAKLAR

[1] **Denman, S.P.** 2009. Improved detection and tracking of objects in surveillance video, *Doktora*, Queensland University of Technology, Faculty of Built Environment and Engineering,

[2] **Gupta, R.K.** 2014. Object detection and tracking in video image, *Yüksek Lisans,* National Institute of Technology Rourkela, Department of Computer Science and Engineering, Rourkela, India.

[3] **Cheng, J., Grossman, M., ve Mckercher, T.**, 2014. *Professional cuda c programming*. John Wiley & Sons.

[4] **Guler, Z. ve Cinar, A.**, 2013. Gpu-based image segmentation using level set method with scaling approach. *Computer Science & Information Technology (CS & IT)*. **3,** 81–92.

[5] Cuda c programming guide. https://docs.nvidia.com/cuda/archive/9.1/pdf/CUDA_C_Programming_Guide.pdf, 2019 25.05.2019.

[6] **Sanders, J. ve Kandrot, E.**, 2010. *Cuda by example: An introduction to general-purpose gpu programming*. Addison-Wesley Professional.

[7] **Yilmaz, A., Javed, O., ve Shah, M.**, 2006. Object tracking: A survey. *ACM Comput. Surv.* **38,** 13.

[8] **Moravec, H.P.**, 1979. Visual mapping by a robot rover, *Proceedings of the 6th international joint conference on Artificial intelligence - Volume 1*. Tokyo, Japan, 598-600.

[9] **Harris, C.G. ve Stephens, M.**, 1988. A combined corner and edge detector, *Alvey vision conference*. 10-5244.

[10] **Jianbo, S. ve Tomasi**, 1994. Good features to track, *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 21-23 June 1994, 593-600.

[11] **Lowe, D.G.**, 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*. **60,** 91-110.

[12] **Morel, J.-M. ve Yu, G.**, 2009. Asift: A new framework for fully affine invariant image comparison. *SIAM journal on imaging sciences*. **2,** 438-469.

[13] **Bay, H., Tuytelaars, T., ve Van Gool, L.**, 2006. Surf: Speeded up robust features, *European conference on computer vision*. 404-417.

[14] **Wu, S. ve Lew, M.S.**, 2014. Riff: Retina-inspired invariant fast feature descriptor, *Proceedings of the 22nd ACM international conference on Multimedia*. 1129-1132.

[15] **Dey, N., Nandi, P., Barman, N., Das, D., ve Chakraborty, S.**, 2012. A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique. *arXiv preprint arXiv:1209.1558*.

[16] **Jain, R. ve Nagel, H.-H.**, 1979. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE transactions on pattern analysis and machine intelligence*, 206-214.

[17] **Wren, C.R., Azarbayejani, A., Darrell, T., ve Pentland, A.P.**, 1997. Pfinder: Real-time tracking of the human body. *IEEE transactions on pattern analysis and machine intelligence*. **19,** 780-785.

[18] **Stauffer, C. ve Grimson, W.E.L.**, 2000. Learning patterns of activity using real-time tracking. *IEEE transactions on pattern analysis and machine intelligence*. **22,** 747-757.

[19] **Elgammal, A., Harwood, D., ve Davis, L.**, 2000. Non-parametric model for background subtraction, *European conference on computer vision*. 751-767.
[20] **Barnich, O. ve Van Droogenbroeck, M.**, 2010. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*. **20,** 1709-1724.
[21] **Qin, L., Sheng, B., Lin, W., Wu, W., ve Shen, R.**, 2015. Gpu-accelerated video background subtraction using gabor detector. *Journal of Visual Communication and Image Representation*. **32,** 1-9.
[22] **Dhanachandra, N., Manglem, K., ve Chanu, Y.J.**, 2015. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*. **54,** 764-771.
[23] **Purohit, P. ve Joshi, R.**, 2013. An efficient approach towards k-means clustering algorithm. *International Journal of Computer Science & Communication Networks*. **4,** 125-129.
[24] **Jose, A., Ravi, S., ve Sambath, M.**, 2014. Brain tumor segmentation using k-means clustering and fuzzy c-means algorithms and its area calculation. *International Journal of Innovative Research in Computer and Communication Engineering*. **2,** 3496-3501.
[25] **Comaniciu, D. ve Meer, P.**, 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 603-619.
[26] **Wu, Z. ve Leahy, R.**, 1993. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1101-1113.
[27] **Shi, J. ve Malik, J.**, 2000. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, 107.
[28] **Xu, N. ve Ahuja, N.**, 2002. Object contour tracking using graph cuts based active contours, *Proceedings. International Conference on Image Processing*. III-III.
[29] **Caselles, V., Kimmel, R., ve Sapiro, G.**, 1997. Geodesic active contours. *International journal of computer vision*. **22,** 61-79.
[30] **Zhu, S.C. ve Yuille, A.**, 1996. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 884-900.
[31] **Paragios, N. ve Deriche, R.**, 2002. Geodesic active regions and level set methods for supervised texture segmentation. *International journal of computer vision*. **46,** 223-247.
[32] **Sethian, J.A.**, 1996. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*. **93,** 1591-1595.
[33] **Hu, W., Zhou, X., Li, W., Luo, W., Zhang, X., ve Maybank, S.**, 2012. Active contour-based visual tracking by integrating colors, shapes, and motions. *IEEE Transactions on Image processing*. **22,** 1778-1792.
[34] **Modava, M. ve Akbarizadeh, G.**, 2017. A level set based method for coastline detection of sar images, *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*. 253-257.
[35] **Porikli, F. ve Yilmaz, A.**, 2012. *Object detection and tracking, Video analytics for business intelligence*, C. Shan, et al., Editors, Springer Berlin Heidelberg: Berlin, Heidelberg. 3-41.

[36] **Freund, Y. ve Schapire, R.E.**, 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*. **55,** 119-139.

[37] **Friedman, J.H.**, 2001. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189-1232.

[38] **Chen, T. ve Guestrin, C.**, 2016. Xgboost: A scalable tree boosting system, *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785-794.

[39] **Papageorgiou, C. ve Poggio, T.**, 2000. A trainable system for object detection. *International journal of computer vision*. **38,** 15-33.

[40] **Mohan, A., Papageorgiou, C., ve Poggio, T.**, 2001. Example-based object detection in images by components. *IEEE transactions on pattern analysis and machine intelligence*. **23,** 349-361.

[41] **Dalal, N. ve Triggs, B.**, 2005. Histograms of oriented gradients for human detection, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 20-25 June 2005, 886-893 vol. 1.

[42] **Zhu, Q., Yeh, M.-C., Cheng, K.-T., ve Avidan, S.**, 2006. Fast human detection using a cascade of histograms of oriented gradients, *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 1491-1498.

[43] **Tuzel, O., Porikli, F., ve Meer, P.**, 2006. Region covariance: A fast descriptor for detection and classification, *European conference on computer vision*. 589-600.

[44] **Mikolajczyk, K., Schmid, C., ve Zisserman, A.**, 2004. Human detection based on a probabilistic assembly of robust part detectors, *European Conference on Computer Vision (ECCV '04)*. Prague, Czech Republic, 2004-05-11, 69--82.

[45] **Leibe, B., Seemann, E., ve Schiele, B.**, 2005. Pedestrian detection in crowded scenes, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 878-885.

[46] **Blum, A. ve Mitchell, T.**, 1998. Combining labeled and unlabeled data with co-training, *Proceedings of the eleventh annual conference on Computational learning theory*. 92-100.

[47] **Qi, Z., Xu, Y., Wang, L., ve Song, Y.**, 2011. Online multiple instance boosting for object detection. *Neurocomputing*. **74,** 1769-1775.

[48] **Levin, A., Viola, P.A., ve Freund, Y.**, 2003. Unsupervised improvement of visual detectors using co-training, *ICCV*. 2.

[49] **Javed, O., Ali, S., ve Shah, M.**, 2005. Online detection and classification of moving objects using progressively improving detectors, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 20-25 June 2005, 696-701 vol. 1.

[50] **Sethian, J.A.**, 1999. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press.

[51] **Andriluka, M., Roth, S., ve Schiele, B.**, 2009. Pictorial structures revisited: People detection and articulated pose estimation, *2009 IEEE conference on computer vision and pattern recognition*. 1014-1021.

[52] **Ross, D.A., Tarlow, D., ve Zemel, R.S.**, 2010. Learning articulated structure and motion. *International journal of computer vision*. **88,** 214-237.

[53] **Sande, K.V.D., Gevers, T., ve Snoek, C.**, 2010. Evaluating color descriptors for object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*. **32,** 1582-1596.

[54] **Fagni, T., Falchi, F., ve Sebastiani, F.**, 2010. Image classification via adaptive ensembles of descriptor-specific classifiers. *Pattern Recognition and Image Analysis*. **20,** 21-28.

[55] **Avidan, S.**, 2007. Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence*. **29,** 261-271.

[56] **Bai, Q., Wu, Z., Sclaroff, S., Betke, M., ve Monnier, C.**, 2013. Randomized ensemble tracking, *2013 IEEE International Conference on Computer Vision*. 1-8 Dec. 2013, 2040-2047.

[57] **Edwards, G.J., Taylor, C.J., ve Cootes, T.F.**, 1998. Interpreting face images using active appearance models, *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. 14-16 April 1998, 300-305.

[58] **Canny, J.**, 1986. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*. **PAMI-8,** 679-698.

[59] **Horn, B.K. ve Schunck, B.G.**, 1981. Determining optical flow. *Artificial intelligence*. **17,** 185-203.

[60] **Lucas, B.D. ve Kanade, T.**, 1981. An iterative image registration technique with an application to stereo vision, *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. Vancouver, BC, Canada, 674-679.

[61] **Sun, D., Roth, S., ve Black, M.J.**, 2010. Secrets of optical flow estimation and their principles, *2010 IEEE computer society conference on computer vision and pattern recognition*. 2432-2439.

[62] **Vesdapunt, N. ve Sinha, U.** *Comparison of optimization methods in optical flow estimation*. arXiv e-prints, 2016.

[63] **Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., ve Pietikainen, M.** *From bow to cnn: Two decades of texture representation for texture classification*. arXiv e-prints, 2018.

[64] **Humeau-Heurtier, A.**, 2019. Texture feature extraction methods: A survey. *IEEE Access*. **7,** 8975-9000.

[65] **Blum, A.L. ve Langley, P.**, 1997. Selection of relevant features and examples in machine learning. *Artificial intelligence*. **97,** 245-271.

[66] **Sethi, I.K. ve Jain, R.**, 1987. Finding trajectories of feature points in a monocular image sequence. *IEEE transactions on pattern analysis and machine intelligence*, 56-73.

[67] **Salari, V. ve Sethi, I.K.**, 1990. Feature point correspondence in the presence of occlusion. *IEEE transactions on pattern analysis and machine intelligence*. **12,** 87-91.

[68] **Rangarajan, K. ve Shah, M.**, 1991. Establishing motion correspondence. *CVGIP: image understanding*. **54,** 56-73.

[69] **Intille, S.S., Davis, J.W., ve Bobick, A.F.**, 1997. Real-time closed-world tracking, *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. 697-703.

[70] **Veenman, C.J., Reinders, M.J., ve Backer, E.**, 2001. Resolving motion correspondence for densely moving points. *IEEE transactions on pattern analysis and machine intelligence*. **23,** 54-72.

[71] **Shafique, K. ve Shah, M.**, 2005. A noniterative greedy algorithm for multiframe point correspondence. *IEEE transactions on pattern analysis and machine intelligence*. **27,** 51-65.

[72] **Eom, K.-Y., Jung, J.-Y., ve Kim, M.-H.**, 2012. A heuristic search-based motion correspondence algorithm using fuzzy clustering. *International Journal of Control, Automation and Systems*. **10,** 594-602.

[73] **Saleemi, I. ve Shah, M.**, 2013. Multiframe many–many point correspondence for vehicle tracking in high density wide area aerial videos. *International journal of computer vision*. **104,** 198-219.

[74] **Yang, X., Qiao, H., ve Liu, Z.-Y.**, 2017. Point correspondence by a new third order graph matching algorithm. *Pattern Recognition*. **65,** 108-118.

[75] **Broida, T.J. ve Chellappa, R.**, 1986. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 90-99.

[76] **Rosales, R. ve Sclaroff, S.**, 2003. A framework for heading-guided recognition of human activity. *Computer Vision and Image Understanding*. **91,** 335-367.

[77] **Li, X., Wang, K., Wang, W., ve Li, Y.**, 2010. A multiple object tracking method using kalman filter, *The 2010 IEEE international conference on information and automation*. 1862-1866.

[78] **Jeong, J.-M., Yoon, T.-S., ve Park, J.-B.**, 2014. Kalman filter based multiple objects detection-tracking algorithm robust to occlusion, *2014 Proceedings of the SICE Annual Conference (SICE)*. 941-946.

[79] **Li, S.E., Li, G., Yu, J., Liu, C., Cheng, B., Wang, J., ve Li, K.**, 2018. Kalman filter-based tracking of moving objects using linear ultrasonic sensor array for road vehicles. *Mechanical Systems and Signal Processing*. **98,** 173-189.

[80] **Bouaynaya, N., Qu, W., ve Schonfeld, D.**, 2005. An online motion-based particle filter for head tracking applications, *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* ii/225-ii/228 Vol. 2.

[81] **Isard, M. ve Blake, A.**, 1998. Condensation—conditional density propagation for visual tracking. *International journal of computer vision*. **29,** 5-28.

[82] **Mackay, D.J.**, 1998. *Introduction to monte carlo methods, Learning in graphical models*, Springer. 175-204.

[83] **Bar-Shalom, Y.**, 1987. *Tracking and data association*. Academic Press Professional, Inc.

[84] **Reid, D.**, 1979. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*. **24,** 843-854.

[85] **Streit, R.L. ve Luginbuhl, T.E.**, 1994. Maximum likelihood method for probabilistic multihypothesis tracking, *Signal and Data Processing of Small Targets 1994*. 394-405.

[86] **Comaniciu, D., Ramesh, V., ve Meer, P.**, 2000. Real-time tracking of non-rigid objects using mean shift, *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*. 15-15 June 2000, 142-149 vol.2.

[87] **Tao, L. ve Xiao-Ping, C.**, 2010. Improved mean shift algorithm for moving object tracking, *2010 2nd International Conference on Computer Engineering and Technology*. 16-18 April 2010, V1-575-V1-578.

[88] **Zhou, H., Yuan, Y., ve Shi, C.**, 2009. Object tracking using sift features and mean shift. *Comput. Vis. Image Underst.* **113,** 345-352.

[89] **Rubner, Y., Tomasi, C., ve Guibas, L.J.**, 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*. **40,** 99-121.

[90] **Leichter, I.**, 2012. Mean shift trackers with cross-bin metrics. *IEEE transactions on pattern analysis and machine intelligence*. **34,** 695-706.

[91] **Schunck, B.G.**, 1986. The image flow constraint equation. *Computer Vision, Graphics, and Image Processing*. **35,** 20-46.
[92] **Shin, J., Kim, S., Kang, S., Lee, S.-W., Paik, J., Abidi, B., ve Abidi, M.**, 2005. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*. **11,** 204-218.
[93] **Zach, C., Gallup, D., ve Frahm, J.**, 2008. Fast gain-adaptive klt tracking on the gpu, *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 23-28 June 2008, 1-7.
[94] **Sinha, S.N., Frahm, J.-M., Pollefeys, M., ve Genc, Y.**, 2011. Feature tracking and matching in video using programmable graphics hardware. *Machine Vision and Applications*. **22,** 207-217.
[95] **Sepehr Aslani, H.M.-N.**, 2013. Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical and Computer Engineering*. **7,** 1251-1256.
[96] **Avidan, S.**, 2001. Support vector tracking, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. 8-14 Dec. 2001, I-I.
[97] **Zhang, S., Yu, X., Sui, Y., Zhao, S., ve Zhang, L.**, 2015. Object tracking with multi-view support vector machines. *IEEE Transactions on Multimedia*. **17,** 265-278.
[98] **Pang, S., Del Coz, J.J., Yu, Z., Luaces, O., ve Díez, J.**, 2017. Deep learning to frame objects for visual target tracking. *Engineering Applications of Artificial Intelligence*. **65,** 406-420.
[99] **Ma, C., Yang, X., Chongyang, Z., ve Yang, M.**, 2015. Long-term correlation tracking, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7-12 June 2015, 5388-5396.
[100] **Ciaparrone, G., Sánchez, F.L., Tabik, S., Troiano, L., Tagliaferri, R., ve Herrera, F.**, 2019. Deep learning in video multi-object tracking: A survey. *arXiv preprint arXiv:1907.12740*.
[101] **Bewley, A., Ge, Z., Ott, L., Ramos, F., ve Upcroft, B.**, 2016. Simple online and realtime tracking, *2016 IEEE International Conference on Image Processing (ICIP)*. 3464-3468.
[102] **Wojke, N., Bewley, A., ve Paulus, D.**, 2017. Simple online and realtime tracking with a deep association metric, *2017 IEEE International Conference on Image Processing (ICIP)*. 3645-3649.
[103] **Kim, M., Alletto, S., ve Rigazio, L.**, 2016. Similarity mapping with enhanced siamese network for multi-object tracking. *arXiv preprint arXiv:1609.09156*.
[104] **Milan, A., Rezatofighi, S.H., Dick, A., Reid, I., ve Schindler, K.**, 2017. Online multi-target tracking using recurrent neural networks, *Thirty-First AAAI Conference on Artificial Intelligence*.
[105] **Huttenlocher, D.P., Noh, J.J., ve Rucklidge, W.J.**, 1993. Tracking non-rigid objects in complex scenes, *1993 (4th) International Conference on Computer Vision*. 93-101.
[106] **Hausdorff, F.**, 1962. Set theory, chelsea publ. *Co., New York*.
[107] **Kang, J., Cohen, I., ve Medioni, G.**, 2004. Object reacquisition using invariant appearance model, *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* 759-762.
[108] **Sato, K. ve Aggarwal, J.K.**, 2004. Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding*. **96,** 100-128.

[109] **Li, B., Chellappa, R., Zheng, Q., ve Der, S.Z.**, 2001. Model-based temporal object verification using video. *IEEE Transactions on Image processing*. **10,** 897-908.
[110] **Terzopoulos, D. ve Szeliski, R.**, 1992. Tracking with kalman snakes. *Active vision*. **20,** 3-20.
[111] **Irani, M. ve Anandan, P.**, 1998. Video indexing based on mosaic representations. *Proceedings of the IEEE*. **86,** 905-921.
[112] **Maccormick, J. ve Blake, A.**, 2000. Probabilistic exclusion and partitioned sampling for multiple object tracking. *International journal of computer vision*. **39,** 57-71.
[113] **Chen, Y., Rui, Y., ve Huang, T.S.**, 2001. Jpdaf based hmm for real-time contour tracking, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. I-I.
[114] **Mansouri, A.-R.**, 2002. Region tracking via level set pdes without motion computation. *IEEE transactions on pattern analysis and machine intelligence*. **24,** 947-961.
[115] **Bertalmio, M., Sapiro, G., ve Randall, G.**, 2000. Morphing active contours. *IEEE transactions on pattern analysis and machine intelligence*. **22,** 733-737.
[116] **Yilmaz, A., Li, X., ve Shah, M.**, 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE transactions on pattern analysis and machine intelligence*. **26,** 1531-1536.
[117] **Shi, Y. ve Karl, W.C.**, 2005. Real-time tracking using level sets, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 20-25 June 2005, 34-41 vol. 2.
[118] **Dzyubachyk, O., Van Cappellen, W.A., Essers, J., Niessen, W.J., ve Meijering, E.**, 2010. Advanced level-set-based cell tracking in time-lapse fluorescence microscopy. *IEEE transactions on medical imaging*. **29,** 852-867.
[119] **Peña, M.G.** 2011. A comparative study of three image matching algorithms: Sift, surf, and fast, *Master of Science*, Utah State University, Civil and Environmental Engineering,
[120] **Wang, H. ve Brady, M.**, 1994. A practical solution to corner detection, *Proceedings of 1st International Conference on Image Processing*. 919-923.
[121] **Smith, S.M. ve Brady, J.M.**, 1997. Susan—a new approach to low level image processing. *International journal of computer vision*. **23,** 45-78.
[122] **Trajković, M. ve Hedley, M.**, 1998. Fast corner detection. *Image and vision computing*. **16,** 75-87.
[123] **Yan, K. ve Sukthankar, R.**, 2004. Pca-sift: A more distinctive representation for local image descriptors, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 27 June-2 July 2004, 506-513.
[124] **Shuo, H., Na, W., ve Huajun, S.**, 2012. Object tracking method based on surf. *AASRI Procedia*. **3,** 351-356.
[125] **Gupta, M., Kejriwal, N., Behera, L., ve Venkatesh, K.**, 2014. Surf-based human tracking algorithm with on-line update of object model. *IFAC Proceedings Volumes*. **47,** 321-325.
[126] **Kass, M., Witkin, A., ve Terzopoulos, D.**, 1988. Snakes: Active contour models. *International journal of computer vision*. **1,** 321-331.
[127] **Stauffer, C. ve Grimson, W.E.L.**, 1999. Adaptive background mixture models for real-time tracking, *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. 246-252.

[128] **Russell, D. ve Gong, S.**, 2005. A highly efficient block-based dynamic background model, *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* 417-422.
[129] **Papageorgiou, C.P., Oren, M., ve Poggio, T.**, 1998. A general framework for object detection, *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. 555-562.
[130] **Rowley, H.A.**, *Neural network-based face detection*. 1999, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.
[131] **Cireşan, D., Meier, U., ve Schmidhuber, J.**, 2012. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.
[132] **Howse, J., Joshi, P., ve Beyeler, M.**, 2016. *Opencv: Computer vision projects with python*. Packt Publishing Ltd.
[133] **Chaoyang, Z.** 2011. Video object tracking using sift and mean shift, *Yüksek Lisans,* Chalmers University of Technology, Communication Engineering,
[134] **Khan, N.Y., Mccane, B., ve Wyvill, G.**, 2011. Sift and surf performance evaluation against various image deformations on benchmark dataset, *2011 International Conference on Digital Image Computing: Techniques and Applications*. 501-506.
[135] **Bay, H., Ess, A., Tuytelaars, T., ve Van Gool, L.**, 2008. Speeded-up robust features (surf). *Computer Vision and Image Understanding*. **110,** 346-359.
[136] **Jurgensen, S.M.** 2014. The rotated speeded-up robust features algorithm (r-surf), *Yüksek Lisans,* Naval Postgraduate School, Electrical and Computer Engineering, Monterey, CA.
[137] **Cook, S.**, 2012. *Cuda programming: A developer's guide to parallel computing with gpus*. Newnes.
[138] **Shao, H., Svoboda, T., Tuytelaars, T., ve Van Gool, L.**, 2003. Hpat indexing for fast object/scene recognition based on local appearance, *International Conference on Image and Video Retrieval*. 71-80.
[139] **Dubuisson, S. ve Gonzales, C.**, 2016. A survey of datasets for visual tracking. *Machine Vision and Applications*. **27,** 23-52.
[140] Valkov, V. Credit card fraud detection using autoencoders in keras | tensorflow for hackers (part vii). 07.08.2019.
[141] **Mortensen, E.N., Deng, H., ve Shapiro, L.**, 2005. A sift descriptor with global context, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 184-190.
[142] **Abdel-Hakim, A.E. ve Farag, A.A.**, 2006. Csift: A sift descriptor with color invariant characteristics, *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 1978-1983.
[143] **Wu, J., Cui, Z., Sheng, V.S., Zhao, P., Su, D., ve Gong, S.**, 2013. A comparative study of sift and its variants. *Measurement science review*. **13,** 122-131.
[144] **Kubelka, P.**, 1948. New contributions to the optics of intensely light-scattering materials. Part i. *Josa*. **38,** 448-457.
[145] **Ester, M., Kriegel, H.-P., Sander, J., ve Xu, X.**, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise, *Kdd*. 226-231.
[146] **Brown, M. ve Lowe, D.G.**, 2007. Automatic panoramic image stitching using invariant features. *International journal of computer vision*. **74,** 59-73.
[147] **Abdel-Aziz, Y.**, 1971. Direct linear transformation from comparator coordinates into object space in close-range photogrammetry, *Proceedings of the ASP Symposium on Close-Range Photogrammetry, 1971*. 1-18.

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

ÖZGEÇMİŞ

Zafer GÜLER	
KİŞİSEL BİLGİLER	
ADRES :	*Fırat Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendsiliği Bölümü* *23119 – Elazığ*
TELEFON :	*+90 (5449485377)*
EPOSTA :	*zaferguler@firat.edu.tr*
DOĞUM YERİ/TARİHİ:	*01/04/1987*
MEDENİ HALİ:	Evli
EĞİTİM BİLGİLERİ	
LİSANSÜSTÜ	***2013, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, Elazığ***
LİSANS :	***2010, Anadolu Üniversitesi, Mühendislik-Mimarlık Fakültesi, Bilgisayar Mühendisliği (İngilizce), Eskişehir***
LİSE :	***2005, Tokat Anadolu Lisesi, Tokat***
İŞ DENEYİMİ	
2010 **– 2010**	***Anadolu Üniversitesi – Dilkom, Eskişehir*** ***Yazılım geliştirici***
2010 **–**	***Fırat Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, Elazığ*** ***Araştırma Görevlisi***
BİLDİĞİ YABANCI DİLLER	
	İngilizce (B2)