SIFT İLE GERÇEK ZAMANLI NESNE TANIMLAMA PERFORMANSININ GELİŞTİRİLMESİ

2022
DOKTORA TEZİ
MAKİNE MÜHENDİSLİĞİ

Ali ÖZTÜRK

Tez Danışmanı
Doç. Dr. İbrahim ÇAYIROĞLU

SIFT İLE GERÇEK ZAMANLI NESNE TANIMLAMA PERFORMANSININ GELİŞTİRİLMESİ

Ali ÖZTÜRK

Tez Danışmanı
Doç. Dr. İbrahim ÇAYIROĞLU

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Makine Mühendisliği Anabilim Dalında
Doktora Tezi
Olarak Hazırlanmıştır

KARABÜK
Aralık 2022

Ali ÖZTÜRK tarafından hazırlanan “SIFT İLE GERÇEK ZAMANLI NESNE TANIMLAMA PERFORMANSININ GELİŞTİRİLMESİ ” başlıklı bu tezin Doktora Tezi olarak uygun olduğunu onaylarım.

Doç. Dr. İbrahim ÇAYIROĞLU .....
Tez Danışmanı, Mekatronik Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından Oy Birliği ile Makine Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir. 28/12/2022

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. İsmail ESEN (KBÜ)	.....
Üye : Prof. Dr. İsmail Hakkı TAYYAR (KBÜ)	.....
Üye : Doç. Dr. İbrahim ÇAYIROĞLU (KBÜ)	.....
Üye : Doç. Dr. Gökhan BAYAR (ZKÜ)	.....
Üye : Dr. Öğr. Üyesi Osman ÜLKİR (MAU)	.....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile Doktora derecesini onamıştır.

Doç. Dr. Müslüm KUZU .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Ali ÖZTÜRK

ÖZET

Doktora Tezi

SIFT İLE GERÇEK ZAMANLI NESNE TANIMLAMA PERFORMANSININ GELİŞTİRİLMESİ

Ali ÖZTÜRK

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Makine Mühendisliği Anabilim Dalı

Tez Danışmanı:
Doç. Dr. İbrahim ÇAYIROĞLU
Aralık 2022, 95 sayfa

Bu çalışmada nesne tanımlamada en çok çalışılan metotlardan biri olan Ölçek Bağımsız Öznicelik Dönüşümü (SIFT) algoritması gürbüzlük ve hesaplama zamanı açısından incelenmiştir. SIFT algoritmasında anahtar-noktalar, Gauss Ölçek Uzay Piramidinden (GSSP) elde edilen Gauss Farkları (DoG) uzayı üzerindeki ekstremum noktalardır. Öznicelik vektörü ise Yönlendiriliş Gradyanların Histogramı (HOG) metodu ile elde edilmektedir.

GSSP'nin oluşturulmasında kullanılan en klasik metot Kesilmiş Ayrık Gauss Konvolüsyonudur (TDGC). Ancak TDGC algoritmasının uygulanmasında, örneklem uzunluğu veya kernel uzunluğunun hesaplama zamanı ve gürbüzlük üzerindeki etkisi sıklıkla göz ardı edilmektedir. Klasik yaklaşımda SIFT algoritmasının giriş görüntüsünün, GSSP hesaplamasından önce boyutu iki katına çıkarılmaktadır. Ayrıca

giriş görüntüsünde olduğu farz edilen bir başlangıç bulanıklık seviyesi vardır. GSSP'nin her bir elemanının bulanıklık seviyesi, ısı denklemi ve Gauss fonksiyonunun yarı-grup özelliği göz önünde bulundurularak açıklanmıştır. Gauss dağılımının hata fonksiyonunu (Erf) temel alarak, TDGC algoritmasının örneklem uzunluğunun, hem gürbüzlük hem de hesaplama zamanı açısından etkili bir şekilde nasıl belirlenebileceği gösterilmiştir. SIFT algoritmasının performansı, algoritmada öngörülen şu değişikliklerle test edilmiştir; başlangıç bulanıklık seviyesi kabulünün iptal edilmesi, giriş görüntüsünün iki katına çıkarılıp çıkarılmaması ve Erf temelli bir denklem ile belirlenen örneklem uzunluğu ile TDGC algoritması.

Diğer taraftan SIFT algoritmasının öznicelik hesaplama algoritması ayrıntılı bir şekilde incelenmiştir. HOG metodu, klasik yaklaşımda algoritmanın gürbüzlüğünü artırmak için trilineer interpolasyon ile birlikte uygulanmaktadır. HOG algoritması ile birlikte kullanılabilecek çeşitli interpolasyon metotları, birlikte öznicelik hesaplama metodunun önemli bir kısmı olan orijinal 'özel gruplama prosesi (SBP)' algoritması bu çalışmada tanımlanmıştır. Verilen orijinal SBP algoritması ile birlikte yama genişlik parametresi, SIFT algoritmasının performansı göz önünde bulundurularak belirlenmiş ve daha sonra da farklı interpolasyon metotları ile birlikte HOG algoritmasının SIFT algoritmasının performansı üzerindeki etkisi incelenmiştir.

Diğer taraftan, özel olarak kaydedilen bir videodaki nesne takibi uygulamasının gürbüzlüğünün arttırılması için SIFT algoritması ile birlikte Tekil Spektrum Analizi (SSA) filtresi bir arada kullanılmıştır. SSA'nın geri kazanım algoritması gerçek zamanlı bir uygulamaya yönelik olarak önemli ölçüde sadeleştirilmiştir.

SIFT algoritması üzerinde yapılan değişikliklerin performans üzerindeki etkisi Oxford, HPatches veri kümeleri ve kendi kaydettiğimiz bir videodaki nesne takibi uygulaması üzerinde test edilmiştir. Veri kümeleri üzerinde performansı ölçmek için Kesişimin Birleşime Oranı (IoU) ve Doğru Eşleşme Oranı (CMR) metrikleri kullanılmıştır. Videodaki nesne takibinin gürbüzlüğünü ölçmek için ise, kesin referans homografi matrisleri mevcut olmadığından dolayı hız temelli yörünge düzgünlük metriği kullanılmıştır.

Sonuç olarak; GSSP hesaplamasından önce giriş görüntüsünün iki katına çıkarılmasının ve Gauss konvolüsyon örnekleminin boyutunun Erf temelli bir denklem ile belirlenmesinin performans üzerindeki etkisinin önemli olduğu görülmüştür. Ancak, öznicelik hesaplanmasında HOG metodu ile birlikte interpolasyon metotlarının kullanılmasının performans üzerinde önemli bir etkisinin olmadığı anlaşılmıştır. SIFT algoritmasının üzerinde yapılan modifikasyonlar ile, algoritmanın 640x480 çözünürlüğünde kaydedilmiş bir videodaki nesne takibi uygulamasında 30 FPS (Frame Per Second)'nin üzerinde bir hızla başarılı bir şekilde çalıştığı görülmüştür. Üstelik, bu videodaki nesne takip yörüngesindeki dalgalanmaların SSA filtresi ile etkili bir şekilde düzeltilebildiği gösterilmiştir.

Anahtar Sözcükler : Nesne tanımlama, şablon eşleştirme, nesne takibi, filtreleme, performans, Ölçek Bağımsız Öznicelik Değişimi (SIFT), Gauss Ölçek Uzayı Piramidi (GSSP), Hata Fonksiyonu (Erf), Gauss Konvolüsyonu, Yönlendirilmiş Gradyanların Histogramı (HOG), İnterpolasyon, Tekil Spektrum Analizi (SSA).

Bilim Kodu **:** 91423 (Mekatronik)

ABSTRACT

Ph. D. Thesis

IMPROVING THE PERFORMANCE of REAL-TIME OBJECT DETECTION with SIFT

Ali ÖZTÜRK

Karabük University
Institute of Graduate Programs
Department of Mechanical Enginneering

Thesis Advisor:
Doç. Dr. İbrahim ÇAYIROĞLU
December 2022, 95 pages

In this study, the Scale Invariant Feature Transforms (SIFT) algorithm, one of the most studied methods in object identification, has been examined in terms of robustness and computation time. In SIFT's algorithm, the keypoints are the extreme points on the Gaussian Differences (DoG) space obtained from the Gaussian Scale Space Pyramid (GSSP). The feature vector is constructed by the Histogram of Orientation Gradient (HOG) method.

The most classical method used in constructing the GSSP is the Truncated Discrete Gaussian Convolution (TDGC). However, the effect of sample or kernel length on computation time and robustness has been often neglected in the TDGC algorithm. The performance of the SIFT algorithm has been tested with the predicted changes in the algorithm; cancellation of the initial blur level assumption, whether to double the

input image befor GSSP computation, and TDGC algorithm with sample length determined by an Erf-based equation.

On the other hand, the feature calculation algorithm of the SIFT algorithm has been examined in detail. The HOG method is implemented with trilinear interpolation to increase the algorithm's robustness in the classical approach. Various interpolation methods that can be used with the HOG algorithm and the 'special binning process' algorithm, which is an essential part of the feature calculation, are given. The patch length parameter was determined by considering the performance with the original 'spatial binning process' algorithm. Then the effect of the HOG algorithm on the performance was explored with different interpolation methods.

On the other hand, to increase the robustness of the object-tracking application in a custom-recorded video, the SIFT algorithm and the Singular Spectrum Analysis (SSA) filter are used together. A significant change has been made to the SSA algorithm for its use in a real-time application.

The performance impact of changes made to the SIFT algorithm was tested on Oxford, HPatches datasets, and an object tracking application in a custom-recorded video. The Intersection Over Union (IoU) and Correct Match Percentage (CMR) metrics were used to measure the performance on the datasets with given ground truth homographies. A velocity-based trajectory smoothness metric was used to measure the robustness of the object tracking in the video since ground truth homographies are unavailable.

As a result, it has been seen that the effect of doubling the input image before GSSP calculation and determining the size of the Gaussian convolution sample with an Erf-based equation on the performance is significant. However, it has been understood that using interpolation methods with the HOG method in calculating the feature vectors of SIFT does not significantly affect the performance. With overall changes to the SIFT algorithm, it has been observed that the object tracking application in a video recorded at 640x480 resolution works successfully at a speed of over 30 FPS.

Moreover, it has been shown in this video that fluctuations in the object tracking trajectory can be effectively smoothed out with the SSA filter.

Key Word : Object detection, template matching, object tracking, filter, performance, Scale Invariant Feature Transform (SIFT), Gaussian Scale Space Pyramid (GSSP), Error Function (Erf), Gauss Convolution, Histogram of Oriented Gradients (HOG), Interpolation, Singular Spectrum Analysis (SSA).

Science Code : 91423 (Mechatronics)

TEŞEKKÜR

Bu tez çalışmasının oluşturulmasında değerli katkılarından dolayı başta danışman hocam Doç. Dr. İbrahim ÇAYIROĞLU'na,

Tez çalışması sürecince izleyicilik görevinde destek veren, bilgi görüşleriyle yönlendiren tez izleme komitesi üyeleri Prof. Dr. İsmail Hakkı TAYYAR ve Prof. Dr. İsmail ESEN hocalarıma;

Her konuda maddi ve manevi desteğini esirgemeyen sevgili eşim Doç. Dr. Esra DEMİR ÖZTÜRK'e sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

BRIEF : Binary Robust Independent Elementary Features (İkili Gürbüz Bağımsız Elementer Öznicelikler)

CMR :Correct Match Ratio (Doğru Eşleşme Oranı)

CPU :Central Process Unit (Merkezi İşlem Birimi)

DGC :Discrete Gauss Convolution (Ayrık Gauss Konvolüsyon)

DoG :Difference of Gaussian (Gauss Frakları)

EBOX :Extended Box Filter (Genişletilmiş Kutu Filtresi)

Erf :Eror Funtion (Hata Fonksiyonu)

FAST :Features from Accelerated Segment Test (Hızlandırılmış Segment Testinden Öznicelikler)

FLANN :Fast Library for Approximated Nearest Neighbors (Yaklaşık En Yakın Komşular İçin Hızlı Kütüphane)

FOP :First Order Perturbation (Birinci Dereceden Perturbasyon)

FPS :Frame Per Second (Saniyedeki Kare Sayısı)

GCC :The GNU Compiler Collections (GNU Derleyeci Koleksiyonu)

GPS :Global Positioning System (Küresel Konumlama Sistemi)

GPU :Graphics Process Unit (Grafik İşlemci Birimi)

GSSP :Gauss Scale Space Pyramid (Gauss Ölçek Uzay Pramiti)

GTB :Ground Truth Box (Kesin Referans Kutusu)

HOG :Histogram of Oriented Gradients (Yönlendirilmiş Gradyanların Histogramı)

JPEG :Joint Photographic Experts Group (Birleşik Fotoğraf Uzmanları Grubu)

IoU :Intersection over Union (Kesişimin Birleşime Oranı)

KLT :Kanady Lucas Tomasi Feature Tracker (Kanadi Lucas Tomasi Öznicelik İzleyicisi)

k-NN :k-Nearest Neighbor Algorithm (En Yakın Komşuluk Algoritması)

L1 Norm : Sum of Absolute Difference of Vektors (Vektörlerin Mutlak Farklarının Toplamı)

L2 Norm :The Square Root of the Sum of the Squares of Absolute Differences (Vektörlerin Mutlak Farklarının Karelerinin Toplamının Kare kökü)

LoG :Laplacian of Gaussian (Gauss Laplasyanı)

ORB :Oriented FAST and Rotated BRIEF (Yönlendirilmiş FAST ve Döndürülmüş BRIEF)

PB :Predicted Box (Tahmin Edilmiş Kutu)

PCA :Principal Component Analysis (Temel Bileşen Analizi)

RANSAC :Random Sample Consensus (Rastgele Örnek Mutabakatı)

RT-SSA :Real Time Singular Spectrum Analysis (Gerçek Zamanlı Tekil Spektrum Analizi)

SBP :Special Bining Process (Özel Gruplama Prosesi)

SIFT :Scale Invariant Feature Transform (Ölçek Bağımsız Öznicelik Dönüşümü)

SII :Stack Integral Image (Yığın Integral Görüntü)

SLAM :Simultaneous Localization and Mapping (Eşzamanlı Lokalizasyon ve Haritalama)

SSA :Singular Spectrum Analysis (Tekil Spektrum Analizi)

SURF :Speeded-up Robust Feature (Hızlandırılmış Gürbüz Öznicelik)

SVD :Singular Value Decomposition (Tekil Değer Ayrışımı)

SVM :Support Vector Machine (Destek Vektör Makinesi)

TDGC :Truncated Discrete Gauss Convolution (Kesilmiş Ayrık Gauss Konvolüsyonu)

BÖLÜM 1

GİRİŞ

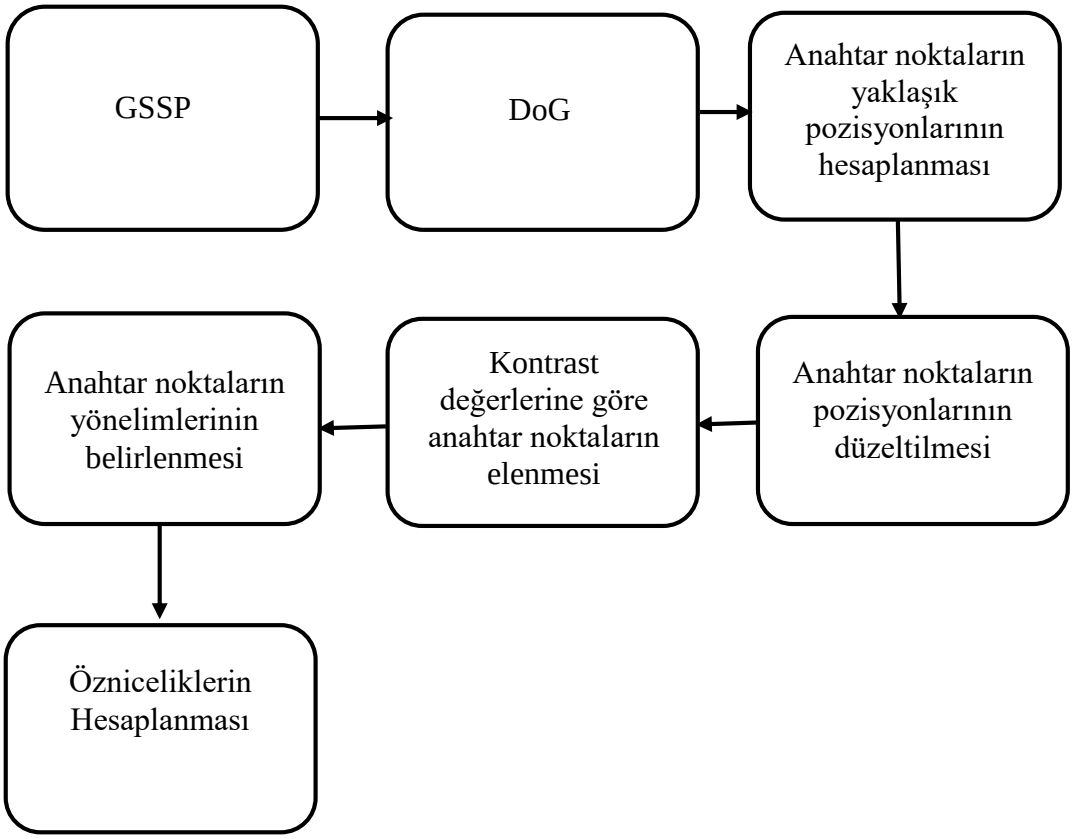
Nesne tanımlama algoritmalarının gürbüzlüğü, görüş açısındaki kısmi engellemeler (occlusion), ışıklandırmadaki ve bakış açılarındaki değişimler, afin ötelemeler, dönme ve yer değiştirme, bulanıklaşma ve ölçeklendirmedeki farklılıklar karşısında değişmezlikleri ile karakterize edilmektedir. Gürbüzlüğün yanı sıra, bir nesne tanımlama algoritmasının hesaplama zamanı da birçok uygulamanın gerçek zamanlı hız gereksinimlerini karşılamak açısından çok büyük önem taşımaktadır [1]. Ancak, genellikle gürbüzlük ile hesaplama zamanı arasında ters bir orantı vardır [2].

Literatürde, gürbüzlüğü düşürmeden hesaplama zamanını geliştirmeye yönelik yapılan çalışmalar, donanım-temelli ve metodoloji-temelli çalışmalar olmak üzere ikiye ayrılabilir. Donanım-temelli çalışmalar; CPU, GPU [3], FPGA [4] gibi donanımların hafıza hiyerarşisi ve çoklu-çekirdek yapısından faydalanabilmek için paralel algoritmalara başvurmaktadır. Metodoloji-temelli çalışmalar ise, benzer veya aynı fonksiyona sahip farklı algoritmalar arasından, gürbüzlük ve hesaplama zamanını göz önüne alarak en optimal olanını tespit etmeye çalışmaktadır [5].

SIFT [6, 7], üzerinde en çok çalışılan lokal tanımlayıcıdır. Gerçek dünyada hala en gürbüz algoritma olarak nitelendirilmektedir [8]. Robotikte SIFT'in kullanıldığı Eşzamanlı Lokalizasyon ve Haritalama (SLAM) metodu, nesne takibi, sahne anlamlandırma, nesne kategorisi belirleme, görüntü birleştirme, kamera kalibrasyonu, güvenlik amaçlı gözlem, yüz, iris (göz) ve parmak izi tespiti, kan grubu tespiti, akciğer kanseri tanısı, bir materyalin üzerinde meydana gelen gerilimin ölçülmesi, ürünlerin üzerindeki hatanın tespit edilmesi, tarım ürünlerinin gözlemlenmesi gibi çok geniş bir uygulama alanında kullanılmıştır. En önemli özelliği, ölçeklendirmeye (zoom out-zoom in) karşı yüksek derecede değişmezlik özelliğidir. SIFT algoritmasının anahtar-noktaların tespiti aşamasında GSSP kullanılır. GSSP'nin oluşturulmasında,

gri ölçekteki giriş görüntüsü Gauss konvolüsyonu ile birbiri ardı sıra filtrelenerek çoğaltılır. SIFT algoritmasının ölçek değişmezliği bu hesaplamadan kaynaklanmaktadır. Ancak GSSP'nin oluşturulması oldukça zaman almaktadır. Bu problemin üstesinden gelebilmek için, bazıları Gauss konvolüsyonu yerine integral görüntü algoritması ile hızlandırılmış kutu filtresi kullanmıştır [9]. Bazıları da GSSP hesaplamasındaki konvolüsyon işlemini tamamıyla ihmal etmiştir [10]. Ancak bu değişiklikler algoritmanın gürbüzlüğünü düşürmektedir. Dolayısıyla günümüzde SIFT hala en gürbüz nesne tanımlama algoritması olarak konumunu korumaktadır.

SIFT algoritmasının en önemli özelliği olan ölçek bağımsızlık, bu metodun anahtar-noktalarını hesaplama yönteminden kaynaklanmaktadır. Lokal bir tanımlayıcı olduğundan, ilk önce anahtar-noktalar tespit edilmekte ve daha sonra her bir anahtar-nokta için bir öznicelik vektörü tanımlanmaktadır. Anahtar-noktalar görüntü üzerinde göze çapan veya ayırt edici noktalardır. Anahtar-noktaların hesaplanması GSSP hesaplanması ile başlar (Şekil 1.1). Bu hesaplama iki temel fonksiyon ile gerçekleştirilir. Bunlar; interpolasyon ile yeniden boyutlandırma ve Gauss konvolüsyonu ile bulanıklaştırmadır. Anahtar-noktaların kaba pozisyonları DoG uzayı üzerindeki ekstremum noktalardır. Anahtar-noktaların bu şekilde hesaplanmasının arkasında yatan temel düşünce ısı denkleminden esinlenmiştir. Bir kere anahtar-noktaların yaklaşık pozisyonları elde edildiğinde bazıları bir piksel yoğunluk eşik değeri kullanılarak reddedilir ve kalanların da pozisyonları düzeltilir. Anahtar-noktaların pozisyonlarının düzeltilmesi, ikinci-dereceden Taylor açılımının bir yaklaşımı olarak kuadratik bir fonksiyonu temel alan lokal bir interpolasyon ile gerçekleştirilir. Üstelik, görüntüdeki kenar uçlarında konumlandırılmış anahtar-noktalar gürültüye karşı oldukça hassas oldukları için temel eğrilikleri kontrol edilerek bir eliminasyon daha gerçekleştirilir.



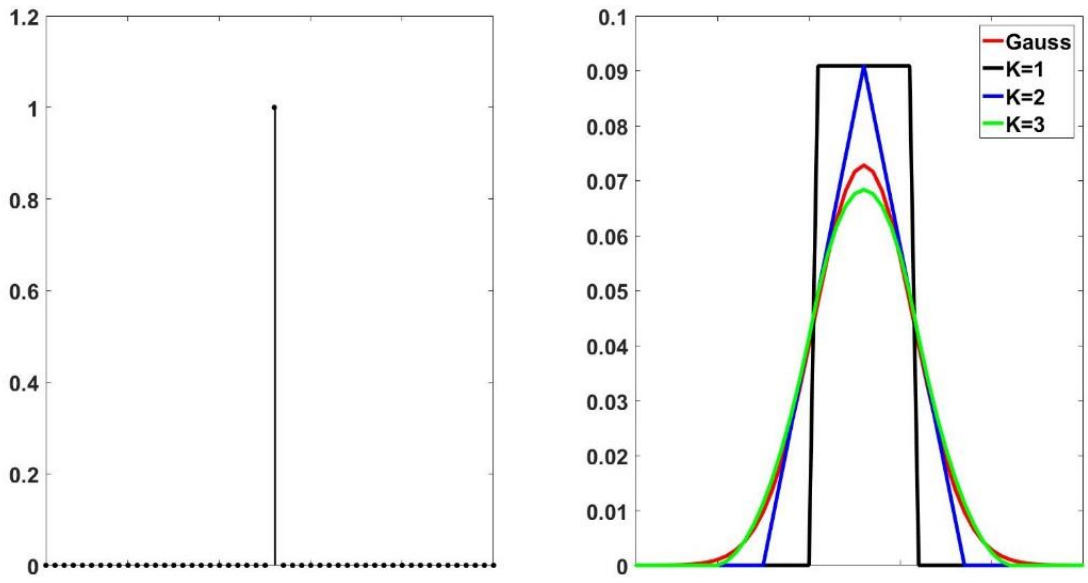
Şekil 1.1. SIFT algoritmasının genel hatları.

HOG, SIFT algoritmasında anahtar-noktalara yönelim (orientation) atamak için ve öznicelik vektörlerinin hesaplanmasında kullanılmaktadır. Ancak HOG ilk defa SIFT algoritmasında kullanılmamış olup, literatürde 1970'li yıllara kadar rastlanılmaktadır [11]. 2000'li yılların başından bu yana, genel olarak insan tanımlamada kullanılmış [12] ve SIFT algoritması ile popülaritesi artmıştır. Klasik yaklaşımda, HOG algoritmasının performansını artırmak için HOG algoritması trilinear interpolasyon ile birlikte uygulanmaktadır.

Diğer taraftan GSSP algoritması da ilk defa SIFT algoritmasında tanımlanmış bir yöntem değildir. 1980'li yılların başlarında, Witkin [13] ve Koenderink [14] bu metodun arkasındaki teorik temeli atmıştır. İlgi çekici bir şekilde, GSSP'nin görüntü elemanlarının ısı denkleminin birer çözümü olduğuna, Koenderink işaret etmiştir. Daha sonraları Lindeberg [15] GSSP teorisini geliştirmiştir. Lindeberg, Gauss

fonksiyonu ile konvolüsyonun ayrılabilirlik, yarı-grup ve yeni-lokal ekstremumlar oluşturmaması gibi özelliklerini sıralayarak öneminin altını çizmiştir.

Gauss konvolüsyonunun değerli ve önemli özellikleri sürekli (continues) Gauss konvolüsyonu için tanımlanmıştır. Ayrıklaştırılmış algoritmalar ise sürekli Gauss konvolüsyonunun bir yaklaşımıdır. Klasik yaklaşımda, Ayrık Gauss Konvolüsyon (DGC) algoritması olarak TDGC kullanılmaktır. Ancak, DGC için başka yaklaşım algoritmaları da mevcuttur. Getreur [16], TDGC, Fourier dönüşümü, tekrarlı kutu filtresi (Şekil 1.2), genişletilmiş kutu filtresi (EBOX) ve Yığın İntegral Görüntü (SII) gibi farklı DGC algoritmalarını hız ve doğruluk yönünden birbirleriyle karşılaştırmıştır. Hız ve doğruluğun her ikisi de göz önünde bulundurulduğunda öne çıkan bir algoritmanın olmadığı sonucuna ulaşmıştır. TDGC algoritmasının zaman kompleksliği, görüntünün boyutuna ve Gauss örnekleminin veya kernelin genişliğine bağlıdır. Getreur, Gauss kernelinin boyutunu, doğruluğu ve hesaplama zamanını göz önünde bulundurarak belirleyebilmek için, Gauss dağılımının hata fonksiyonu (Erf) [17] temelli bir denklemim kullanılabileceğini göstermiştir.



Şekil 1.2. Tekrarlı kutu filtresi, Gauss konvolüsyununun bir yaklaşımıdır. Solda dürtü sinyali, sağda ise dürtü sinyalinin Gauss kerneli ile ve tekrarlı kutu filtresi (tekrar sayısı: K=1, 2, 3) ile konvolüsyon cevapları verilmiştir

SIFT algoritmasının hızı ve gürbüzlüğüne tekrar dönersek, metodoloji-temelli bazı yaklaşımlar bu konuyu ele almıştır. Örneğin, PCA-SIFT [18], hesaplama zamanını ilerletmek için öznicelik vektör boyutunu düşürmeye çalışmıştır. HOG metodunun yerine temel bileşen analizini (PCA) kullanarak gradyanlardaki temel değişim elde edilmeye çalışılmıştır. Buradaki temel hedef öznicelik vektörünün boyutunu düşürmektir. Ancak, öznicelik vektörünün boyutundan ise anahtar-noktaların sayısının hesaplama zamanı üzerindeki etkisi daha fazladır. Bir karşılaştırma çalışmasında, Juan ve Gwun [19], ölçeklendirme, bulanıklaşma ve dönmeye karşı değişmezlik söz konusu olduğunda SIFT'in, PCA-SIFT'e göre daha ileri bir performans gösterdiğini rapor etmiştir. SIFT algoritmasının öznicelik vektör boyutunu, öznicelik vektörünün hesaplandığı bölgenin hücrelerinin (veya alt bölgelerinin) sayısını azaltarak da düşürülebilir.

Bazı çalışmalar ise SIFT algoritmasının parametrelerini optimize etmeye çalışmış ve algoritma üzerinde bazı modifikasyonlar yapmışlardır. Sima ve Buckley [20], hiperspektral görüntüler üzerinde eşleşen anahtar-noktaların sayısını artırmak için kontrast ve kenar eşik-değerleri (contrast and edge treshold), oktav ve ölçek sayısı, Gauss bulanıklaştırma parametresinin başlangıç değeri ve ayrıca eşleştirme algoritmasında kullanılan en yakın komşuluk oranı gibi parametrelerin en optimal olanını bulmaya çalışmıştır. Alonso-Fernandez vd. [21], aynı parametreleri iris (göz) tanımlama için incelemiştir. Fan vd. [22], optik ve SAR uydu görüntü çiftlerini birleştirmek için, GSSP'nin ilk oktavında konumlandırılmış anahtar-noktaları ve anahtar-noktalara yönelim atamayı ihmal etmiş ve ayrıca öznicelik vektör boyutunu düşürmüştür. Gajjar ve Patani [23], Destek Vektör Makinesi (SVM) temelli bir nesne sınıflandırma algoritmasının performansını artırabilmek için, HOG algoritmasındaki histogram gözlerinin ve hücre sayısını değiştirerek öznicelik vektörünün boyutunu artırmıştır.

Diğer taraftan, HOG algoritması SIFT algoritmasında kullanımının yanı sıra, güvenlik amaçlı gözetlemede de insan tanımlama için kullanılmıştır. HOG, görüntü piksellerinin gradyan açılarının bir istatistiğidir. Histogram değerleri bir görüntü üzerinde lokal bir bölgedeki ışıklandırma, renk değişimleri ve görüntü kontürlerindeki hafif kaymalara rağmen ana karakteristiklere karşı değişmez kalabilmektedir. HOG

algoritmasında bir gradyan açı histogramı, bu histogramın gözlerini gradyanın şiddeti (magnitude) ile oylayarak oluşturulur.

HOG'un insan tanımlamadaki kullanımı SIFT algoritmasındaki uygulamasından farklıdır [24]. İnsan tanımlamada, herhangi bir anahtar-nokta hesaplamasına gitmeden, verilen giriş görüntüsünün tek bir ölçeğinin tamamı üzerinde daha önceden tanımlanmış sabit bir boyutta kayan pencerelerin her biri için hesaplanmakta ve görüntünün tamamı için büyük bir öznicelik vektöründe toplanmaktadır. Ancak, SIFT algoritmasında GSSP uzayı kullanılarak belirlenmiş anahtar-noktaların her biri için ayrı ayrı hesaplanmaktadır. Öznicelikler GSSP üzerinde konumlandırılmış anahtar-noktaların porsiyonunu merkeze alan bir dikdörtgensel bölge içinde hesaplanırlar. Bu dikdörtgensel bölge sıklıkla yama (patch) olarak da isimlendirilir. Dikdörtgensel bölge alt bölgelere bölünür. Bu bölme işlemi Özel Gruplama İşlemi (special bining process-SBP) olarak adlandırılabilir. Gözler alt-bölgelerin iki-boyutlu koordinatlarıdır ve alt-bölgelere birer hücre denir. Her bir hücre için histogram hesaplanır ve tek bir vektörde hepsi toplanır. HOG algoritmasının gücü bu gözleme, açıların oylanması ve normalizasyon işlemlerinden kaynaklanmaktadır.

Görüntüleme şartlarındaki hafif bir değişim hücre kenarlarındaki histogram değerlerinde ciddi bir değişime neden olabilir. Dolayısı ile bu durum eşleşmesi gereken iki farklı görüntü bölgesinin histogram değerlerinin, şayet histogram gözleri basit bir şekilde oylanıyor ise birbirinden çok farklı olmasına neden olabilir. Dolayısı ile bu tür problemleri göz önüne alarak, Dalal [12], HOG algoritmasında trilinear interpolasyonun kullanılmasının, insan tanımlama performansını artıracağını ileri sürmüştür. Raxle Won ve Lien [25], HOG algoritmasında trilinear-interpolasyon ile Gauss ağırlıklaştırmanın (Gaussian weighting) kullanılmasının insan tanımlamada AdaBoost öğrenme algoritmasının performansını artırdığını ifade etmiştir. Kim ve Cho [26], yine bir insan tanımlama uygulamasında, HOG algoritmasının hücre içi lineer interpolasyon, hücreler arası trilinear interpolasyon ile kullanılmasının performans üzerindeki etkisini incelemiş ve ayrıca interpolasyonun hesaplama zamanını düşürmeye çalışmıştır.

Diğer taraftan Gerçek Zamanlı Nesne Takibi trafik gözetleme, insan sayma, insan hareketlerini izleme, hücre içindeki proteinleri takip etme gibi çok geniş bir yelpazede uygulama alanına sahiptir [27]. Nesne takibi, dinamik nesne tanımlama olarak tanımlanabilir [28]. Nesne takibinin temel problemleri nesne tanımlamayla aynıdır. Ancak, kendine özgü bazı problemleri de vardır. Nesne takibi için seçilecek optimal algoritma, kameranın pozisyonu, kullanılan kamera sayısı ve takip edilecek nesne sayısına göre değişebilmektedir. Şayet kamera sabit ise, hareketsiz pikseller elimine edilerek (background removel) işlem önemli ölçüde basitleştirilebilir [29]. Şayet hareket eden hedef nesnelerin sayısı artar ise tanımlama işlemi ve takip zorlaşır.

Nesne takibi iki temel adımda gerçekleşmektedir; hedef nesneyi tanımlama ve takip eden çerçevelerde tanımlamayı sürdürme [30]. Takibin temel amacı, model görüntü sahnede var olduğu müddetçe konumunu kaybetmemektir.

Bu tez çalışmasında nesne tanımlama bir şablon eşleştirme problemi olarak ele alınmaktadır. Şablon eşleştirmede, verilen bir model görüntünün pozisyonu herhangi bir sahne içerisinde tespit edilmeye çalışılmaktadır.

SIFT algoritması yüksek gürbüzlüğünden dolayı nesne takibinde sıklıkla başvurulan bir nesne tanımlama algoritması olmuştur. Zhou vd. [31] nesne takibi için SIFT ve Ortalama Shift algoritmasını bir araya getirmiştir. Jabar vd. [32], İHA temelli bir nesne takibi uygulamasında SIFT ile KLT algoritmasını birlikte kullanmıştır. Bu yaklaşımda nesne ilk olarak SIFT algoritması ile tanımlanmış ve daha sonra Kanadi Lucas Tomasi Öznicelik İzleyicisi (KLT) ile takip edilmiştir.

Nesne takibinde, nesne tanımlama algoritması tüm görüntü dönüşümleri ve deformasyonlarına karşı mükemmel olarak gürbüz olmadığı için, bazı çerçevelerde kaybedilebilir veya düşük doğrulukta tespit edilebilir. Bu durum takip edilen nesnesinin yörüngesinde gürültü oluşturarak dalgalanmalara neden olabilmektedir. Bu tür durumlarda, nesnenin pozisyonu birbirini takip eden çerçeveleri kullanarak filtrelenebilir veya yumuşatılabilir.

SSA, zaman serilerini analiz etmede kullanılan çok değişkenli bir istatistik metodudur. Zaman serilerinin temel bileşen analizi PCA olarak da bilinmektedir [33]. SSA parametrik olmayan (veya modelden bağımsız) olan bir filtreleme metodu olarak kullanılmaktadır [34]. Dinamik sistemler gibi bazı sistemlerin modellerini tahmin etmek amacıyla da kullanılmıştır. Literatürde genel olarak kaydedilmiş verilerin sonradan analizi için tercih edilmiştir [35, 36]. Ansari [37], bir GPS'den elde edilmiş verileri filtrelemek için Kalman ve SSA filtrelerini birlikte kullanmıştır. Ancak bu çalışmada SSA algoritmasının gerçek zamanlı uygulanmasına dair herhangi bir açıklama yoktur. Bhownik vd. [38], inşaat yapılarında gerçekleşen hasarları gerçek zamanlı tespit edebilmek için, SSA algoritmasının Henkel matrisini Birinci Dereceden Perturbasyon (FOP) metodunu kullanarak güncellemiştir.

Tüm SIFT algoritması birçok alt fonksiyondan ve bu fonksiyonlar için tanımlanmış birçok parametreden oluşmaktadır. Bu çalışmada, potansiyel olarak SIFT algoritmasının performansını etkileyebilecek iki ana fonksiyon üzerinde durulmuştur. Bu ana fonksiyonlar; anahtar-noktaların hesaplanmasında kullanılan GSSP ve özniceliklerin hesaplanmasında kullanılan HOG metotlarıdır. Son olarak da bir nesne takibi uygulamasında nesne takibi algoritmasının çıktısının (pozisyon verisinin), SSA algoritması ile etkili bir şekilde gerçek zamanlı olarak filtrelenmesi üzerinde durulmuştur.

BÖLÜM 2

GAUSS ÖLÇEK UZAY PRAMİDİ VE GAUSS FARKLARI UZAYI

SIFT algoritmasının tamamı birçok fonksiyon ile birlikte tanımlanmış bir hiper-parametre uzayı oluşturmaktadır. Ancak, SIFT algoritmasının performansı, hem gürbüzlük hem de hesaplama zamanı, Gauss Ölçek Uzay Piramit'i (GSSP) ve Gauss Frakları (DoG) uzaylarının hesaplanmasına bağlıdır. Dolayısıyla performans, GSSP uzayının hesaplanmasında kullanılacak Ayrık Gauss Konvolüsyon (DGC) algoritmasına bağlıdır. Bu çalışmada, DGC için klasik yaklaşım olan Kesilmiş Ayrık Gauss Konvolüsyonu (TDGC) algoritması kullanılmıştır. TDGC algoritmasında örneklem genişliği konvolüsyon işleminin doğruluğunu ve hesaplama zamanını etkilemektedir. Konvolüsyonun hem doğruluğu hem de hızı göz önünde bulundurularak, Gauss dağılımının hata fonksiyonu (Erf) yardımıyla, örneklem genişliğinin nasıl hesaplanacağı açıklanmıştır. Literatürde, SIFT algoritmasının performansı üzerinde örneklem genişliğinin etkisi henüz incelenmemiştir. Bu durum yapılan bu çalışmada iki veri kümesi ve özel olarak kaydedilmiş bir video üzerinde incelenmiştir.

Diğer taraftan, GSSP uzayının görüntü üyeleri ısı denkleminin çözümleri olduğu ve Gauss Laplasyanın (LoG) bir yaklaşımı olan DoG uzayı da GSSP uzayından elde edildiği için, GSSP uzayı üyelerinin nasıl elde edildiği çok önemlidir. Bu çalışmada, Gauss fonksiyonunun yarı grup özelliği ve ısı denkleminden faydalanılarak, GSSP uzayının üyelerinin bulanıklaşma seviyelerinin nasıl hesaplandığı açıklanmıştır.

Klasik yaklaşımda, GSSP uzayının giriş görüntüsü üzerinde olduğu farz edilen bir başlangıç değer bulanıklaşma seviyesi vardır. Bu varsayımın performans üzerindeki etkisi incelenmiştir. Ayrıca klasik yaklaşımda, GSSP uzayının hesaplanmasında giriş görüntüsü iki katına çıkarılmaktadır. Bu hesaplamanın da performans üzerindeki etkisi incelenmiştir.

Bu bölümün takip eden alt bölümlerinin içerikleri şu şekilde özetlenebilir; Bölüm 2.1'de, GSSP, LoG, DoG ve ısı denkleminin arasındaki ilişkiyi kuran teori açıklanmıştır. Daha sonra GSSP uzayının görüntü üyelerinin bulanıklaşma seviyelerinin hesaplanma metodu verilmiştir. Bölüm 2.2'de, TDGC algoritması ve örneklem genişliğinin Erf'yi temel alarak hesaplanması metodu açıklanmıştır.

2.1. GSSP, DoG, LoG ve ISI DENKLEMİ ARASINDAKİ İLİŞKİ

GSSP, herhangi bir görüntünün gittikçe artan derecede uzaktan fotoğraflanması sonucunda, görüntü detaylarının kaybolmasını taklit etmektedir. GSSP'de, anahtar noktaların pozisyonu yalnızca görüntü üzerinde belirlenmeyip, ayrıca ölçekler üzerinde de belirlenmektedir. Dolayısıyla algoritma ölçeklendirmeye karşı yüksek bir gürbüzlük kazanabilmektedir.

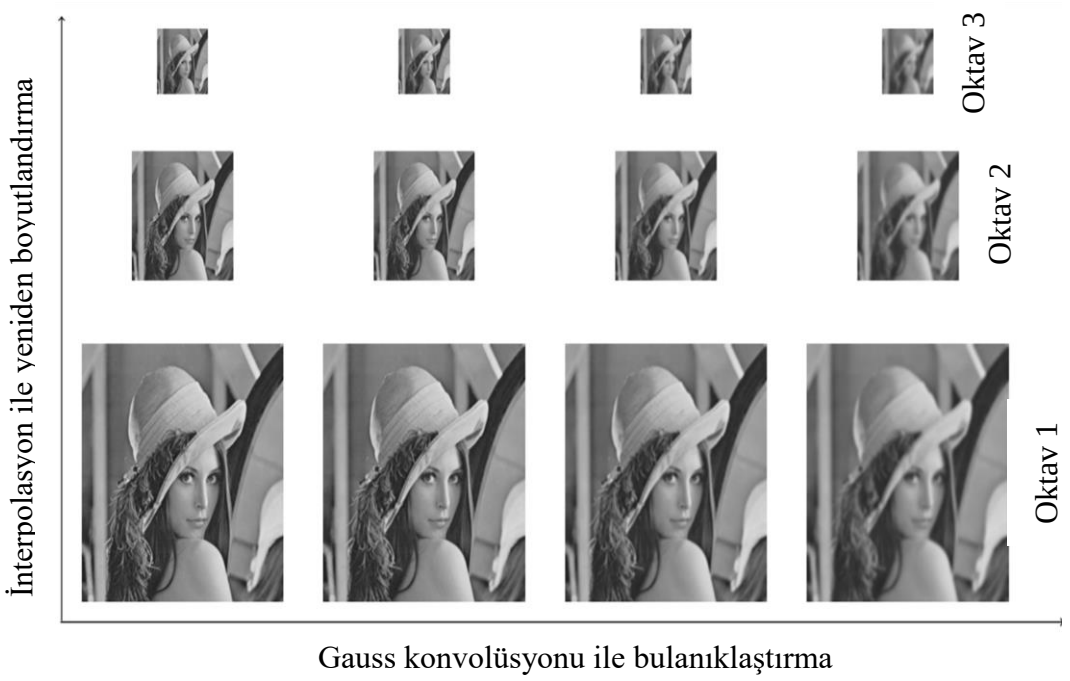
GSSP, ölçeklendirme ve yeniden boyutlandırma parametrelerine göre iki boyutlu bir görüntü uzayıdır (Şekil 2.1). Bu görüntü uzayı, yeniden boyutlandırma parametresine göre alt uzaylara ayrılmaktadır. Her bir alt uzayda, görüntü boyutu yarıya indiği için, bu alt uzaylara oktav denilir. Alt uzayların (oktavların) sayısı ve her bir oktavdaki ölçek sayısı kullanıcı tarafından önceden belirlenir. GSSP'nin hesaplanmasında, kullanıcı tarafından belirlenen ölçek sayısı üç artırılır. Bu fazla ölçeklerden biri DoG'un hesaplanmasında, diğer ikisi ise anahtar noktaların konumlandırılması sırasında kullanılır.

DoG uzayının elemanları, GSSP uzayının bir oktavındaki birbirini takip eden iki görüntünün farkını alarak oluşturulur (Şekil 2.2). Bu fark LoG'un bir yaklaşımıdır. Dolayısı ile DoG uzayının elemanları ikinci dereceden türevlenmiş görüntülerdir. DoG ile anahtar noktaların belirlenmesi düşüncesinin temelinde ısı denklemi yer almaktadır. GSSP uzayının elemanları ısı denkleminin çözümleridir. Dolayısıyla, Gauss dağılımı ile bir görüntüyü konvolve etmeyi bir tas çorbayı ısıtmaya benzetebiliriz.

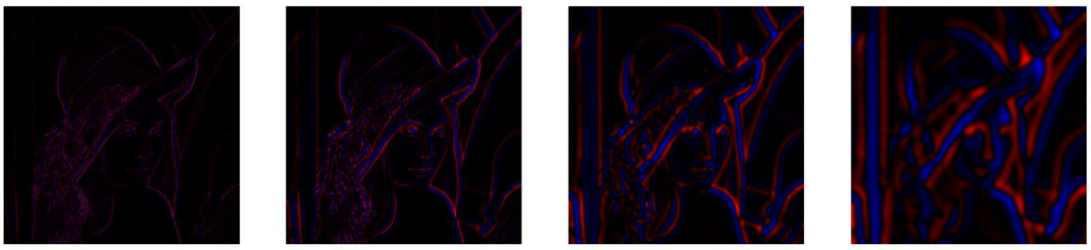
Gauss fonksiyonunun ayrılabilirlik özelliğinden dolayı, iki boyutlu bir görüntüyü tek boyutlu olarak ele alabiliriz. $f(x), \forall x \in R$ için bir boyutlu herhangi bir sinyal ve ayrıca

$\sigma > 0$ ölçeklendirme (veya bulanıklaştırma) parametresi olsun. Bu durumda Gauss konvolüsyonu aşağıdaki Eşitlik 2.1 ile ifade edilebilir.

$$U(x, \sigma) = f(x) * G_\sigma(x) \tag{2.1}$$



Şekil 2.1. GSSP uzayının bulanıklaşma ve yeniden boyutlandırma parametrelerine göre oluşturulan iki boyutlu görüntüsü.



Şekil 2.2. Oktav sayısı 4 ve ölçek sayısı 2 olduğunda, son oktavdaki Gauss Farklarının resim temsilleri. Pozitif ve negatif değerler sırası ile mavi ve kırmızı renk yoğunlukları ile gösterilmiştir.

Eşitlik 2.1, ölçeklendirme parametresine göre bir görüntü ailesini temsil etmektedir. Bu durumda $U(x, 0) = f(x)$ başlangıç koşuluna göre, $U(x, \sigma)$ görüntü ailesi ısı denklemi Eşitlik 2.2'nin birer çözümüdür.

$$\sigma \Delta U(x,\sigma) = \frac{\partial U(x,\sigma)}{\partial \sigma} \tag{2.2}$$

Eşitlik 2.2'de Δ ikinci dereceden diferansiyel operatör olan Laplas operatörünü temsil etmektedir. Eşitlik 2.2' nin ayrıklaştırılmış hali aşağıdaki Eşitlik 2.3 ile ifade edilebilir;

$$\sigma \Delta U(x,\sigma) = \lim_{k \to 1} \frac{U(x,k\sigma) - U(x,\sigma)}{\sigma(k-1)} \tag{2.3}$$

Eşitlik 2.3'de, $k-1$ değeri sıfıra yaklaşırken, Gauss Farkları, Gauss Laplasyanın bir yaklaşımıdır. Bu yaklaşım aşağıdaki Eşitlik 2.4 ile ifade edilebilir.

$$\sigma^2 \Delta U(x,\sigma) \approx U(x,k\sigma) - U(x,\sigma) \tag{2.4}$$

Eşitlik 2.4'ün sağ tarafı DoG, sol tarafı ise σ^2 ile LoG'un çarpımıdır. Ancak burada sağ tarafın $k-1$ ile bölümünün ihmal edildiğine dikkat edilmelidir. GSSP uzayı boyunca $k-1$ sabit olduğu için, Lowe [7] bu ihmalin algoritmanın kararlılığını etkilemediğini ifade etmiştir. Dolayısı ile DoG, σ^2 ile LoG'un çarpımının bir yaklaşımı olmaktadır.

Diğer taraftan, giriş görüntüsünün boyutu, GSSP hesaplamasına sokulmadan önce interpolasyon ile iki katına çıkarılmaktadır. Bu hesaplamanın arkasında yatan düşünce, görüntü çözünürlüğünü yükseltmektir. GSSP uzayının giriş görüntüsü çekirdek (veya temel) görüntü olarak isimlendirilmektedir.

Çizelge 2.1'de çekirdek görüntü hesaplama algoritması verilmiştir. Bu algoritmanın, giriş görüntüsü üzerinde var olduğu kabul edilen bir bulanıklaşma seviyesi (σ_i) bulunmaktadır. Ayrıca burada şayet giriş görüntüsünün boyutu iki katına çıkarılıyor ise, bulanıklaşma seviyesi (σ_i) de iki katına çıkarılmaktadır.

Çizelge 2.1'de σ, GSSP uzayının ilk oktavının ilk görüntüsü olan çekirdek görüntünün sahip olması istenilen bulanıklık seviyesini temsil etmektedir. Bu durumda, Gauss konvolüsyonunun yarı grup özelliğinden faydalanarak, aşağıdaki eşitliği yazabiliriz;

$$G_{\sigma_i}(x)G_{\sigma_d}(x) = G_{\sqrt{\sigma_i^2+\sigma_d^2}}(x) = G_{\sigma}(x) \tag{2.5}$$

Eşitlik 2.5'de, σ_d giriş görüntüsünün konvolve edilmesi gereken Gauss fonksiyonunun bulanıklaşma parametresidir. Böylece, çekirdek görüntünün bulanıklaşma seviyesi σ olacaktır. Eşitlik 2.5'den, Eşitlik 2.6 aşağıdaki gibi elde edilmiş olur;

$$\sigma_d = \sqrt{\sigma^2 - \sigma_i^2} \tag{2.6}$$

Son durumda çekirdek görüntünün (GSSP uzayının giriş görüntüsü) bulanıklaşma seviyesi σ'dır. Çekirdek görüntüyü takip eden ikinci görüntünün bulanıklaşma seviyesi, Eşitlik 2.4 deki denkleme göre DoG'un LoG'un bir yaklaşımı olabilmesi için $k\sigma$ olmalıdır. Bu durumda, tekrar yarı grup özelliğinden faydalanarak Eşitlik 2.7 yazılabilir.

$$G_{\sigma}(x)G_{\sigma_{n_2}}(x) = G_{\sqrt{\sigma^2+\sigma_{n_2}^2}}(x) = G_{k\sigma}(x) \tag{2.7}$$

Eşitlik 2.7'den de çekirdek görüntünün konvolve edilmesi gereken Gauss parametresi aşağıdaki Eşitlik 2.8 ile bulunur.

$$\sigma_{n_2} = \sqrt{(k\sigma)^2 - \sigma^2} \tag{2.8}$$

Bu durumda, oktavdaki ikinci görüntünün bulanıklık seviyesi $k\sigma$ dır. Dolayısıyla oktavdaki tüm görüntülerin bulanıklık seviyesi Çizelge 2.2'de verilen algoritma ile hesaplanabilir. Burada bulanıklık seviyesi çarpanı (k), oktavdaki ölçek sayısının bir fonksiyonu olarak hesaplanmaktadır (Eşitlik 2.9).

$$k = 2^{\frac{1}{n_{scl}}} \tag{2.9}$$

Örneğin, ölçek sayısı, $n_{scl} = 3$ ise bu takdirde $k = 1.2599$ olmaktadır. Burada belirtmeliyiz ki, algoritmanın kararlılığı yüksek derecede bu parametreye bağlıdır.

Çizelge 2.1. Çekirdek görüntünün hesaplanması.

Giriş	:	I : Giriş görüntüsü.
Giriş	:	*doubleFlag: Giriş* görüntüsünün boyutunun iki katına çıkarılıp çıkarılmayacağına göre 'true' veya 'false' değerlerini almaktadır.
Giriş	:	σ_i : Giriş görüntüsünde olduğu kabul edilen bulanıklaşma seviyesi.
Giriş	:	σ : Çekirdek görüntüsünün getirilmek istendiği bulanıklaşma seviyesi.
Çıkış	:	B : Çekirdek görüntüsü.
Adım 1	:	**If** *doubleFlag* **then**
Adım 2	:	$\sigma_d \leftarrow \sqrt{\sigma^2 - (2\sigma_i)^2}$
Adım 3	:	$I_{2x} \leftarrow$ Giriş Görüntüsünün İki Katına Çıkar (I)
Adım 4	:	$B \leftarrow$ Gauss Konvolüsyonunu Hesapla (I_{2x}, σ_d)
Adım 5	:	**else**
Adım 6	:	$\sigma_d \leftarrow \sqrt{\sigma^2 - \sigma_i^2}$
Adım 7		$B \leftarrow$ Gauss Konvolüsyonunu Hesapla (I, σ_d)

Çizelge 2.2. Her bir oktav için bulanıklaşma vektörünün hesaplanması.

Giriş	:	n_{scl} : Oktavdaki ölçek sayısı.
Giriş	:	σ : Çekirdek görüntüsünün bulanıklaşma seviyesi.
Çıkış	:	V_σ: Bulanıklaşma vektörü.
Adım 1	:	$V_\sigma[0] \leftarrow \sigma$
Adım 2	:	$k \leftarrow 2^{\frac{1}{n_{scl}}}$
Adım 3	:	**for** $i = 1; i < n_{scl} + 3; i++$ **do**
Adım 4	:	$\sigma_{prev} \leftarrow k^{i-1}\sigma$
Adım 5	:	$\sigma_{new} \leftarrow k\sigma_{prev}$

Adım 6	:	$V_\sigma[i] \leftarrow \sqrt{\sigma_{new}^2 - \sigma_{prev}^2}$

2.2. KESİLMİŞ AYRIK GAUSS KONVOLÜSYONU (TDGC)

$F(X)$'in R^2 üzerinde bir görüntüyü temsil eden sürekli bir fonksiyon, $U(X)$'in filtrelenmiş görüntü ve $\forall X = (x, y) \in R^2$ olduğunu kabul edelim. Bu takdirde Gauss konvolüsyonu aşağıdaki gibi tanımlanır;

$$U(X) = G_\sigma(X) * F(X) := \int_{R^2} G_\sigma(X')F(X - X')\, dX' \tag{2.10}$$

Eşitlik 2.10'da G_σ, standart sapma σ ve ortalama $\mu = 0$ parametreleri ile Gauss dağılım fonksiyonunu göstermektedir;

$$G_\sigma(X) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2.11}$$

Gauss fonksiyonunun ayrılabilirlik özelliğinden dolayı;

$$G_\sigma(X) = G_\sigma(x)G_\sigma(y) \tag{2.12}$$

Gauss konvolüsyonu tek boyutlu uygulanabilir; ilk önce x-ekseni yönünde ve daha sonra y-ekseni yönünde veya tam tersi olabilir.

Diğer taraftan Eşitlik 2.10, iki boyutlu sürekli bir fonksiyon için Gauss konvolüsyonunu tanımlamaktadır. Ancak gerçekte görüntü ayrıktır. f_i tek boyutlu ayrık bir giriş sinyali, u_i tek boyutlu ayrık çıkış sinyali ve $i \in Z$ olsun. Bu takdirde tek boyutlu DGC aşağıdaki gibi Eşitlik 2.13'de tanımlanır.

$$u_i = \sum_{j=-\infty}^{\infty} g_j f_{i-j} \tag{2.13}$$

Eşitlik 2.13'de, g_j ayrık Gauss fonksiyonunu göstermektedir;

$$g_j = \frac{1}{S(\infty)} e^{\frac{-j^2}{2\sigma^2}}, j \in Z \tag{2.14}$$

Dikkat edilirse Eşitlik 2.14'de Gauss dizisinin toplamının bire eşit olmasını garanti altına almak için normalleştirme işleminde, Eşitlik 2.11'deki $\sqrt{2\pi}\sigma$ yerine $S(\infty)$ kullanılmıştır. Gauss dağılımına bu sebepten dolayı, normal dağılım da denir. Bir olasılık yoğunluk fonksiyonu olarak, dağılımın toplamı bir olmalıdır. Burada $S(\infty)$ aşağıdaki gibi tanımlanmaktadır;

$$S(\infty) = \sum_{i=-\infty}^{\infty} e^{\frac{-i^2}{2\sigma^2}} \tag{2.15}$$

Diğer taraftan, konvolüsyon sonsuz bir aralıkta uygulanamayacağı için, bu sonsuz aralık $|i| \leq r$ veya eşdeğer olarak $i \in [-r, r]$ olacak şekilde kesilmelidir. Bu durumda Kesilmiş Ayrık Gauss Konvolüsyonu (TDGC) aşağıdaki gibi Eşitlik 2.16'da ifade edilir.

$$u_i^{tr} = \sum_{j=-r}^{r} g_j^{tr} f_{i-j} \tag{2.16}$$

Eşitlik 2.16'da ayrık Gauss örneklemi g_j^{tr} ise aşağıdaki gibi ifade edilir.

$$g_j^{tr} = \frac{1}{S(r)} e^{\frac{-j^2}{2\sigma^2}}, j \in [-r, r] \tag{2.17}$$

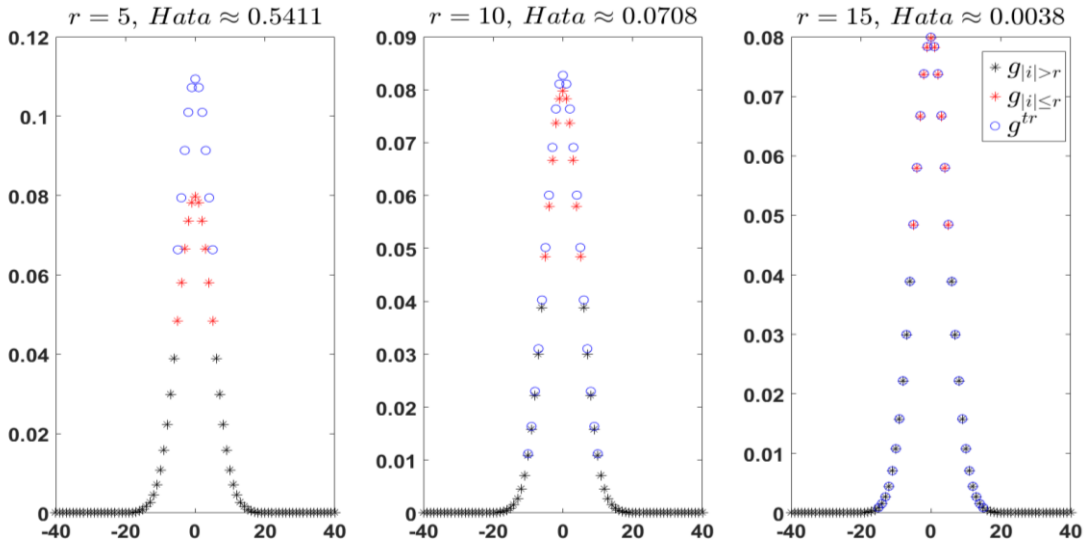
Eşitlik 2.17'de $S(r)$ ise aşağıdaki gibi ifade edilmektedir;

$$S(r) = \sum_{i=-r}^{r} e^{\frac{-i^2}{2\sigma^2}} \tag{2.18}$$

Eşitlik 2.17'de şayet r değeri uygun bir şekilde belirlenebilir ise, $S(r)$ yerine sabit $\sqrt{2\pi}\sigma$ değeri normalizasyon için kullanılabilir. Burada ayrıca $S(r)$ hesaplamasının kesikli ayrık Gauss örnekleminin hesaplama zamanını artıracağına dikkat edilmelidir.

Burada temel problem şu şekilde ifade edilebilir; hem konvolüsyonun hızı hem de doğruluğu göz önünde bulundurarak, uygun bir r örneklem uzunluk değeri belirlenmelidir (Şekil 2.3). Kabul edelim ki, ayrık ve kesikli ayrık Gauss örneklemleri arasındaki fark verildiği şekilde L1 norm kullanılarak aşağıdaki gibi ifade edilebilir.

$$Hata \approx |g_i - g_i^{tr}| \tag{2.19}$$



Şekil 2.3. $\sigma = 5$ için Ayrık Gauss örneklemi ile Kesilmiş Ayrık Gauss örneklemi arasındaki fark. Ayrık Gauss örneklemi için örneklem uzunluğu 10^4 alınmıştır.

Diğer taraftan, ayrık Gauss örneklem dizisi iki guruba ayrılabilir;

$$g_i = \{g_{|i|\leq r}, g_{|i|>r}\} \tag{2.20}$$

Eşitlik 2.20'de $g_{|i|\leq r}$ ve $g_{|i|>r}$ sırası ile g_i'nin $[-r, r]$ ve $(-\infty, -r) \cup (r, \infty)$ aralıklarında kalan bölgelerini temsil etmektedir. Bu durumda Eşitlik 2.19 aşağıdaki gibi tekrar yazılabilir;

$$Hata \approx \left| g_{|i|\leq r} - g_i^{tr} \right| + \left| g_{|i|>r} \right| \tag{2.21}$$

Bu takdirde, şayet $\sqrt{2\pi}\sigma$ normalizasyon için kullanılırsa, Eşitlik 2.21 aşağıdaki gibi tekrar yazılabilir.

$$Hata \approx \left| g_{|i|>r} \right| = \frac{1}{\sqrt{2\pi}\sigma} \sum_{|i|>r} e^{\frac{-i^2}{2\sigma^2}} \leq 1 - G(r) \tag{2.22}$$

Eşitlik 2.22'de, $G(r)$ aşağıdaki gibi tanımlanmaktadır.

$$G(r) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-r}^{r} e^{\frac{-x^2}{2\sigma^2}} dx \tag{2.23}$$

Eşitlik 2.23'de, şayet $t = x/(\sqrt{2}\sigma)$ ve $dt = dx/(\sqrt{2}\sigma)$ olacak şekilde değişken dönüşümü yapılır ise aşağıdaki Eşitlik 2.24 elde edilir.

$$G(r) = \frac{1}{\sqrt{\pi}} \int_{-r^*}^{r^*} e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \int_{0}^{r^*} e^{-t^2} dt = Erf(r^*) = Erf\left(\frac{r}{\sqrt{2}\sigma}\right) \tag{2.24}$$

Eşitlik 2.24'de, Erf hata fonksiyonunu temsil etmektedir. Eşitlik 2.22 ve 2.24 birlikte göz önünde bulundurulur ise aşağıdaki eşitsizlik elde edilebilir.

$$Hata \leq 1 - Erf\left(\frac{r}{\sqrt{2}\sigma}\right) \tag{2.25}$$

Sonuç olarak Eşitlik 2.25'ten r çekilirse aşağıdaki Eşitsizlik 2.26 elde edilir.

$$r \leq \sqrt{2}\sigma InvErf(1 - Hata) \tag{2.26}$$

Eşitsizlik 2.26'da $InvErf$ hata fonksiyonunun tersini ifade etmektedir. Dolayısıyla, r belirlenmiş bir hata toleransına göre aşağıdaki eşitlik ile belirlenebilir;

$$r = floor\left(\sqrt{2}\sigma InvErf(1 - Hata)\right) \tag{2.27}$$

Hata fonksiyonu analitik bir çözüme sahip olmayan bir integral ile tanımlandığı için, kendisi ve tersi nümerik bir yaklaşım ile hesaplanabilmektedir. Birçok yazılım paketi bu hesaplama fonksiyonuna sahiptir. Ancak bu çalışmada $InvErf$ aşağıdaki denklem ile hesaplanmıştır [39]

$$InvErf(x) = \sqrt{\sqrt{\left(\frac{2}{0.147\pi} + \frac{ln(1-x^2)}{2}\right)^2 - \frac{ln(1-x^2)}{0.147}} - \left(\frac{2}{0.147\pi} + \frac{ln(1-x^2)}{2}\right)} \tag{2.28}$$

BÖLÜM 3

ÖZNİCELİKLERİN HESAPLANMASI

Bu bölümün temel amacı, SIFT algoritmasının öznicelik hesaplanması kısmında HOG ile birlikte interpolayon tekniklerinin kullanılmasının etkisini araştırmaktır. Literatürde, SIFT algoritmasının performansı üzerinde HOG algoritması ile birlikte interpolasyon metotlarının kullanılmasının etkisini inceleyen bir çalışma henüz rapor edilmemiştir. Burada SIFT algoritması çerçevesinde HOG algoritmasının kullanımının, insan tanımlamadaki kullanımından farklı olduğunu göz önünde bulundurmalıyız. Dolayısı ile de performans üzerindeki etkisinin aynı olmasını bekleyemeyiz. Bu bölümde HOG hesaplamasında kullanılabilecek farklı interpolasyon metotları verilmiştir. Üstelik HOG algoritmasının çok önemli bir kısmı olan 'özel gruplama prosesi' algoritması açıklanmıştır. İnterpolasyon metotlarının performans üzerindeki etkisi iki veri kümesi üzerinde test edilmiştir.

GSSP uzayına ait görüntü üzerindeki piksel pozisyonu (x, y), GSSP üzerindeki ölçek pozisyonu σ, yeniden boyutlandırma parametresi δ ve temel yönelimi $\hat{\theta}$ olmak üzere, herhangi bir anahtar noktanın Key(x, y, σ, δ, θ̂) ile gösterildiğini varsayalım. Bu durumda, bu anahtar-nokta için öznicelik vektörü, piksel koordinatlarını merkeze alan karesel bir bölge içinde hesaplanır. Bu karesel bölgenin bir kenarının yarısı veya çapı aşağıdaki Eşitlik 3.1 ile bulunur;

$$\text{çap} = \lambda_{\text{decr}} * \text{scl} * \text{r} * \sqrt{2} \tag{3.1}$$

Eşitlik 3.1'de λ_{descr} sabit bir parametre değeridir. λ_{descr} için varsayılan değer 3'dür. scl parametresi, anahtar noktanın ölçeği veya GSSP uzayındaki konuma göre aşağıda verilen Eşitlik 3.2 ile değişmektedir.

$$scl = \frac{1}{2}.\sigma.\delta \tag{3.2}$$

Eşitlik 3.1'deki r parametresi, çap uzunluğunu modifiye ederek SIFT algoritmasının performansı üzerindeki etkisini incelememize olanak sağlayan bir değişkendir. Ayrıca Eşitlik 3.1'e, karesel bölge döndürüldüğünde istenilen sınırda kalabilmesi için $\sqrt{2}$ çarpanı eklenmiştir.

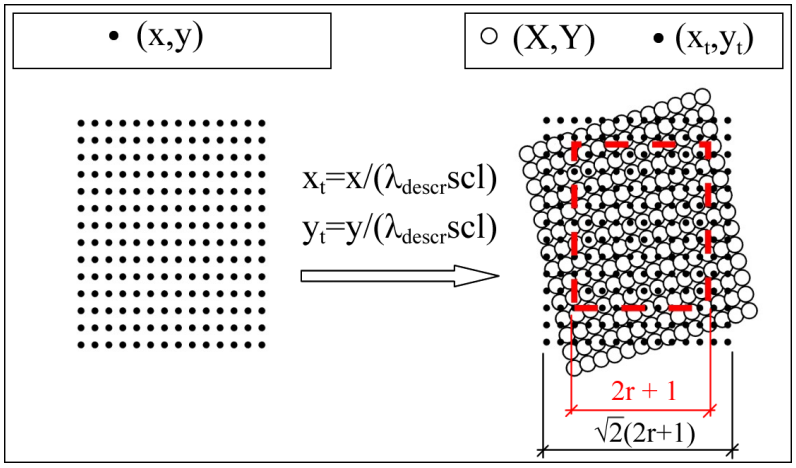
Karesel bölgedeki tüm piksel koordinatları, anahtar noktanın temel yönelimi $\hat{\theta}$ ile aşağıdaki eşitlik kullanılarak döndürülür.

$$\begin{aligned} X &= x_t \cos\hat{\theta} - y_t \sin\hat{\theta} \\ Y &= x_t \sin\hat{\theta} - y_t \cos\hat{\theta} \end{aligned} \tag{3.3}$$

Eşitlik 3.3'de x, y $\in$ [−çap, çap] olmak üzere;

$$\begin{aligned} x_t &= \frac{x}{\lambda_{\text{descr}} \text{scl}} \\ x_t &= \frac{x}{\lambda_{\text{descr}} \text{scl}} \end{aligned} \tag{3.4}$$

şeklinde hesaplanmaktadır. Böylece döndürülmüş piksel koordinatları X, Y $\in$ $[-r\sqrt{2}, r\sqrt{2}]$ aralığına ötelenmiş olmaktadır. Bu durumda öznicelik vektörü, [−r, r] aralığı içerisinde kalan piksel koordinatlarından hesaplanmaktadır.



Şekil 3.1. Öznicelik vektörünün hesaplanacağı karesel bölgenin (yama) ölçeklendirilmiş ve döndürülmüş gösterimi.

Öznicelik vektörü, Şekil 3.1'de gösterilen kırmızı kesikli çizginin içinde kalan bölgede, alt bölgelere ayrılarak hesaplanır. Bu alt karesel bölgelere birer hücre denir. Her bir hücre için ayrı ayrı HOG hesaplanır ve bunlar bir araya getirilerek anahtar-noktanın öznicelik vektörü oluşturulur. Bu karesel bölgenin (veya yamanın) alt bölgelere (veya hücrelere) bölünmesi işlemine 'özel gruplama prosesi-(SBP)' denir. Gözler her bir hücrenin iki boyutlu koordinatlarıdır.

3.1. ÖZEL GRUPLAMA PROSESİ (SBP)

SBP'de, tüm piksel koordinatlarının hangi hücrede oldukları ve bu hücre içindeki koordinatları belirlenmelidir.

Yamanın her iki yönde de (x ve y) bölünme sayısının d olduğunu farz edelim. Bu takdirde, karesel hücrelerin bir kenar uzunluğu;

$$r_0 = \frac{2\mathrm{r}+1}{d} \tag{3.5}$$

olur. Hücre içindeki bir pikselin (X, Y) koordinatı için aşağıdaki eşitlikler yazılabilir;

$$X = -\mathrm{r} + x_{\mathrm{bin}} r_0 \tag{3.6}$$

$$Y = -\mathrm{r} + y_{\mathrm{bin}} r_0$$

veya eşdeğer olarak;

$$\begin{aligned} x_{\mathrm{bin}} &= (X + r)/\mathrm{r}_0 \\ y_{\mathrm{bin}} &= (Y + r)/\mathrm{r}_0 \end{aligned} \tag{3.7}$$

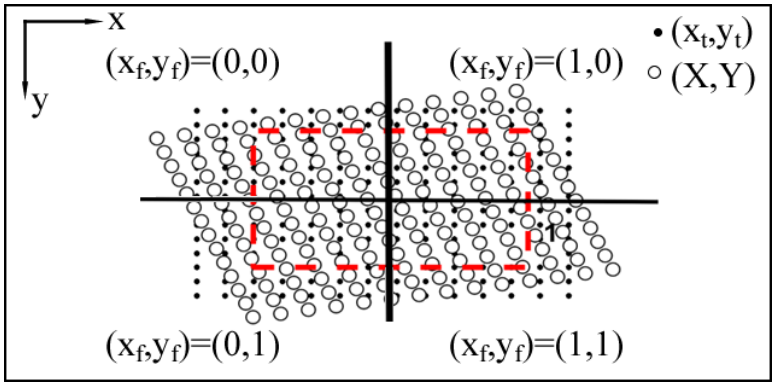
eşitlikleri yazılabilir. Bu takdirde, $(x_{\mathrm{bin}}, \mathrm{y}_{\mathrm{bin}})$ çifti için;

$$\begin{aligned} x_f &= floor(x_{bin}) \\ y_f &= floor(y_{bin}) \end{aligned} \tag{3.8}$$

eşitliklerini yazarsak (x_f, y_f) çifti (X, Y) koordinatına sahip pikselin içinde bulunduğu hücrenin koordinatları olur. Üstelik bu pikselin bulunduğu hücre içindeki koordinatlarını (x_d, y_d) ile gösterirsek, bu koordinatlar aşağıdaki eşitlikler ile belirlenir;

$$\begin{aligned} x_d &= x_{bin} - x_f \\ y_d &= y_{bin} - y_f \end{aligned} \tag{3.9}$$

Eşitlik 3.8 ile belirlenen sayılar $0 \leq x_f, y_f \leq d$ aralığındaki tam sayılar, Eşitlik 3.9 ile belirlenenler ise $0 \leq x_d, y_d \leq 1$ aralığındaki rasyonel sayılardır. Şekil 3.2'de yama iki parçaya bölünmüştür. Örneğin, sağ üst köşedeki hücrenin koordinatları (1, 0) dır. Bu bölgenin içindeki değerlerde rasyonel sayılardır.



Şekil 3.2. Yamanın özel gruplama prosesi (SBP).

Şayet bir hücrede hesaplanacak olan histogramdaki göz sayısını n ile gösterirsek, bu durumda öznicelik vektörünün uzunluğu d x d x n olacaktır. Her bir hücreye ait histogramın öznicelik vektöründeki başlangıç indeksi aşağıdaki gibi belirlenir;

$$offset = x_f.n + y_f.d.n \tag{3.10}$$

SIFT algoritmasında standart olarak n=8 ve d=4 olarak belirlenmiştir. Dolayısıyla da öznicelik vektörünün toplam uzunluğu 128'dir.

(x_i, y_i) koordinatlarına sahip bir piksel için gradyan şiddetinin m ve gradyan açısının θ olduğunu varsayalım. Bu değerler aşağıda verilen eşitlikler ile hesaplanır;

$$\begin{aligned} g_x &= x_{i+1} - x_{i-1} \\ g_y &= y_{i+1} - y_{i-1} \\ m &= \sqrt{g_x^2 + g_y^2} \\ \theta &= arctan\left(\frac{g_y}{g_x}\right) \end{aligned} \tag{3.11}$$

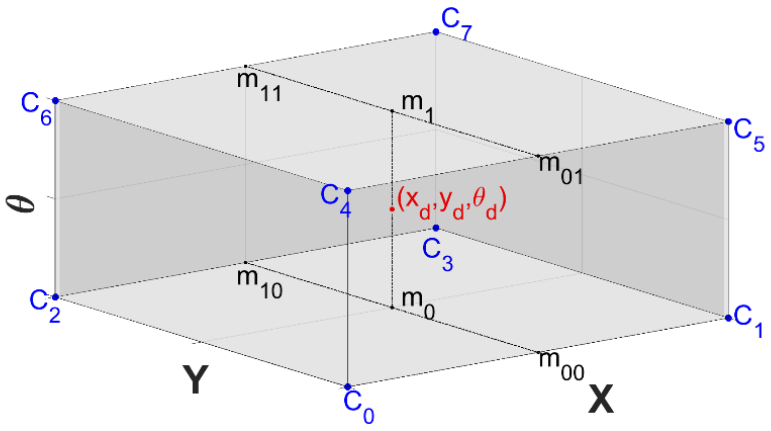
Pikselin bulunduğu bölge anahtar noktanın temel yönelim açısı $\hat{\theta}$ ile saat yönünün tersi istikametinde döndürüldüğü için, bu pikselin döndürülmüş koordinatlarının (X_i, Y_i) için gradyan açısı;

$$\theta_r = \theta - \hat{\theta} \tag{3.12}$$

olur. Bu durumda şayet açı derece olarak hesaplanıyor ise aşağıdaki eşitlikler yazılabilir;

$$\begin{aligned} \theta_{bin} &= \frac{\theta_r n}{360} \\ \theta_f &= floor(\theta_{bin}) \\ \theta_d &= \theta_{bin} - \theta_f \end{aligned} \tag{3.13}$$

Eşitlik 3.13'de θ_f HOG' nin gözünü, θ_d ise pikselin bulunduğu birim küp içerisindeki koordinatını ifade etmektedir (Şekil 3.3). Burada $0 \leq \theta_f \leq n$ bir tam sayı ve $0 \leq \theta_d \leq 1$ bir rasyonel sayıdır.



Şekil 3.3. Trilineer interpolasyon küpü. Trilineer interpolasyon ile piksele ait gradyan şiddeti m, küpün köşelerine ötelenir.

Böylece döndürülmüş bir (X_i, Y_i) koordinatı için, hücre gözlerini (koordinatlarını) (x_f, y_f, θ_f) ve hücre (küp) içindeki üç boyutlu birim koordinatları (x_d, y_d, θ_d) hesaplamış oldu. Başka bir ifade ile (x_d, y_d, θ_d) pikselin birim küp içerisindeki koordinatlarıdır ve bu koordinattaki değer, gradyan şiddeti m dir. Gradyan şiddeti m ise Gauss fonksiyonu ile ağırlıklaştırılmıştır. Böylece, merkezden uzak piksellerin öznicelik vektörüne olan katkısı azaltılmıştır. Bu hesaplamanın temel amacı,

pozisyondaki hafif bir değişimin öznicelik vektöründe yüksek bir değişikliğe sebep olmasını engellemektir.

Özel gruplama prosesinden sonra, bu bölümün geri kalanında basit HOG, hücre içi lineer interpolasyon, hücre içi trilinear interpolasyon ve hücreler arası trilinear interpolasyon algoritmaları açıklanacaktır

3.2. BASİT HOG

Basit HOG histogramdaki oryantasyon gözlerinin, gradyan şiddet değerleri hiçbir interpolasyon uygulanmadan oylanmasıdır. Çizelge 3.1'de basit HOG algoritması verilmiştir.

Çizelge 3.1. Algoritma 1; Basit HOG.

Giriş	:	$x_f, y_f, \theta_f, x_d, y_d, \theta_d, m$
Çıkış	:	f_V *// Uzunluğu d* x *d* x *n olan öznicelik vektörü*
Adım 1	:	$\boldsymbol{for\ each}\ i;\ f\_V[i] \leftarrow 0$ *// f_V' nin tüm girdilerini sıfırla.*
Adım 2	:	$o \leftarrow x_f.n + y_f * d * n$ *//histogramın vektördeki başlangıç indisi*
Adım 3	:	$h\_i \leftarrow \theta_f$ *//histogram göz indeksi*
Adım 4	:	$f\_V[o + h\_i] \leftarrow f\_V[o + h\_i] + m$

3.3. HÜCRE İÇİ LİNEER İNTERPOLE EDİLMİŞ HOG

Her bir hücrenin kendi içinde, gradyan şiddet değeri m, iki komşu histogram gözü arasında, hücre içindeki θ_d oryantasyon pozisyonuna göre paylaştırılmıştır. Eğer piksel histogramın son gözüne gidiyor ise gradyan şiddeti değeri son ve ilk göz arasında paylaştırılır. Bu durum mode operasyonu ile garanti altına alınmıştır. Çizelge 3.2'de hücre içi lineer interpole edilmiş HOG algoritması verilmiştir.

Çizelge 3.2. Algoritma 2; Hücre içi lineer interpole edilmiş HOG

Giriş	:	$x_f, y_f, \theta_f, x_d, y_d, \theta_d, m$
Çıkış	:	f_V *//Uzunluğu d* x *d* x *n olan öznicelik vektörü*

Adım 1	:	$\boldsymbol{for\ each}\ i;\ f\_V[i] \leftarrow 0$ *// f_V' nin tüm girdilerini sıfırla.*
Adım 2	:	$o \leftarrow\ x_f.n + y_f * d * n$ *//histogramın vektördeki başlangıç indisi*
Adım 3	:	$h\_i \leftarrow \theta_f$ *//histogram göz indeksi*
Adım 4	:	$h\_n\_i \leftarrow mod\ e\ (\theta_f + 1, n)$ *//bir sonraki histogram göz indeksi*
Adım 5	:	$f\_V[o + h\_i] \leftarrow f\_V[o + h\_i] + m * \theta_d$
Adım 6	:	$f\_V[o + h\_n\_i] \leftarrow f\_V[o + h\_n\_i] + m * (1 - \theta_d)$

3.4. HÜCRE İÇİ TRİLİNEER İNTERPOLE EDİLMİŞ HOG

(x_d, y_d, θ_d) üçlüsünün bir pikselin birim küp içindeki koordinatları olduğunu ve bu koordinattaki değerinde gradyan şiddeti m olduğunu düşünelim. Hücre içinde, gradyan şiddet değeri m, trilineer interpolasyon ile birim küpün yalnızca bir köşesine ötelenmiştir. Eğer $\theta_f = 0$ ise, gradyan şiddeti birim küpün C_0 köşesine ötelenmiştir (Şekil 3.3).

Trilineer interpolasyon tekrarlı bir lineer interpolasyondur. İnterpolasyon adımlarının sırası birbirinden bağımsız olduğundan herhangi bir sırada uygulanabilir. Hücre içi trilineer interpolasyon algoritması Çizelge 3.3'de verilmiştir.

Çizelge 3.3. Algoritma 3; Hücre içi trilineer interpole edilmiş HOG.

Giriş	:	$x_f, y_f, \theta_f, x_d, y_d, \theta_d, m$
Çıkış	:	f_V *//Uzunluğu dxdxn olan öznicelik vektörü.*
Adım 1	:	$\boldsymbol{for\ each}\ i;\ f\_V[i] \leftarrow 0$ *//f_V' nin tüm girdilerini sıfırla.*
Adım 2	:	$o \leftarrow\ x_f.n + y_f * d * n$
Adım 3	:	$m_0 \leftarrow \theta_d m$ *//Gradyan şiddet değeri m, z-ekseni boyunca paylaştırılmıştır* $m_1 \leftarrow (1 - \theta_d)m$
Adım 4	:	$m_{00} \leftarrow y_d m_0,\ m_{10} \leftarrow (1 - y_d)m_0$ *// m_0 ve m_1 , y-ekseni boyunca paylaştırılmıştır.* $m_{01} \leftarrow y_d m_1,\ m_{11} \leftarrow (1 - y_d)m_1$
Adım 5	:	$C_0 \leftarrow x_d m_{00}, C_1 \leftarrow (1 - x_d)m_{00}$, $C_2 \leftarrow x_d m_{10}, C_3 \leftarrow (1 - x_d)m_{10}$, $C_4 \leftarrow x_d m_{01}, C_5 \leftarrow (1 - x_d)m_{01}$, $C_6 \leftarrow x_d m_{11}, C_7 \leftarrow (1 - x_d)m_{11}$ *// Son olarak; m_{00}, m_{10}, m_{01}, ve m_{11} , x-ekseni boyunca paylaştırılmıştır.*

Adım 6	:	$h\_i \leftarrow \theta_f$ *// Tek bir hücre içinde, gradyan şiddet değeri m, birim küpün yalnızca tek bir köşesine ötelenmiştir.*
Adım 7	:	$f\_V[o + h\_i] \leftarrow f\_V[o + h\_i] + C_{h\_i}$

3.5. HÜCRELER ARASI TRİLİNEER İNTERPOLE EDİLMİŞ HOG

Trilineer interpolasyon ile elde edilen birim küpün köşelerindeki sekiz değer, birbirine komşu olan dört hücreye ikişer ikişer dağıtılmıştır. Örneğin hücre koordinatlarının (gözlerinin) (0, 0) ve histogram gözünün değerinin de $\theta_f = 0$ olduğunu varsayalım. Bu durumda C_0 ve C_1 köşelerindeki değerler (0, 0) hücresindeki histogramın ilk ve ikinci gözüne , C_2 ve C_3 köşelerindeki değerler (0, 1) hücresindeki histogramın ilk ve ikinci gözüne, C_4 ve C_5 köşelerindeki değerler (1, 0) hücresindeki histogramın ilk ve ikinci gözüne ve son olarak da C_6 ve C_7 köşelerindeki değerler (1, 1) hücresindeki histogramın ilk ve ikinci gözüne gidecektir (Şekil 3.3). Komşu hücre koordinatları mode operasyonu ile belirlenmiştir. Çizelge 3.4'de hücreler arası interpole edilmiş HOG algoritması verilmiştir. Küpün köşelerindeki değerler, $C_0 - C_7$, Çizelge 3.3'deki algoritmada verildiği şekilde hesaplandığından burada tekrar verilmemiştir.

Çizelge 3.4. Algoritma 4; Hücreler arası interpole edilmiş HOG.

Giriş	:	$x_f, y_f, \theta_f, x_d, y_d, \theta_d, m$
Çıkış	:	f_V *//Uzunluğu dxdxn olan öznicelik vektörü.*
Adım 1	:	$for\ each\ i;\ f\_V[i] \leftarrow 0$ *// f_V' nin tüm girdilerini sıfırla.*
Adım 2	:	$h\_i \leftarrow \theta_f$
Adım 3	:	$x_{f\_m} \leftarrow mode\ (x_f + 1, d)$ *// Komşu hücrelerin koordinatları* $y_{f\_m} \leftarrow mode\ (y_f + 1, d)$
		// Her bir hücredeki histogram lokasyonu;.
Adım 4	:	$o0 \leftarrow x_f.n + y_f.n.d + h\_i$ *// İlk hücredeki lokasyon.*
Adım 5		$o1 \leftarrow x_f.n + y_f.n.d + h\_i + 1$
Adım 6	:	$o2 \leftarrow x_{f\_m}.n + y_f.n.d + h\_i$ *//İkinci hücredeki lokasyon.*
Adım 7	:	$o3 \leftarrow x_{f\_m}.n + y_f.n.d + h\_i + 1$
Adım 8	:	$o4 \leftarrow x_f.n + y_{f\_m}.n.d + h\_i$ *// Üçüncü hücredeki lokayon.*

Adım 9	:	$o5 \leftarrow x_f.n + y_{f\_m}.n.d + h\_i + 1$
Adım 10	:	$o6 \leftarrow x_{f\_m}.n + y_{f\_m}.n.d + h\_i$ *// Dördüncü hücredki lokasyon*
Adım 11	:	$o7 \leftarrow x_{f\_m}.n + y_{f\_m}.n.d + h\_i + 1$
		// Histogram değerlerini vektördeki yerlerine yaz.
Adım 12	:	$f\_V[o0] \leftarrow f\_V[o0] + C_0$, $f\_V[o1] \leftarrow f\_V[o1] + C_1$
Adım 13	:	$f\_V[o2] \leftarrow f\_V[o2] + C_2$, $f\_V[o3] \leftarrow f\_V[o3] + C_3$
Adım 14	:	$f\_V[o4] \leftarrow f\_V[o4] + C_4$, $f\_V[o5] \leftarrow f\_V[o5] + C_5$
Adım 15	:	$f\_V[o6] \leftarrow f\_V[o6] + C_6$, $f\_V[o7] \leftarrow f\_V[o7] + C_7$

BÖLÜM 4

SSA FİLTRESİNİN GERÇEK ZAMANLI NESNE TAKİBİNE UYARLANMASI

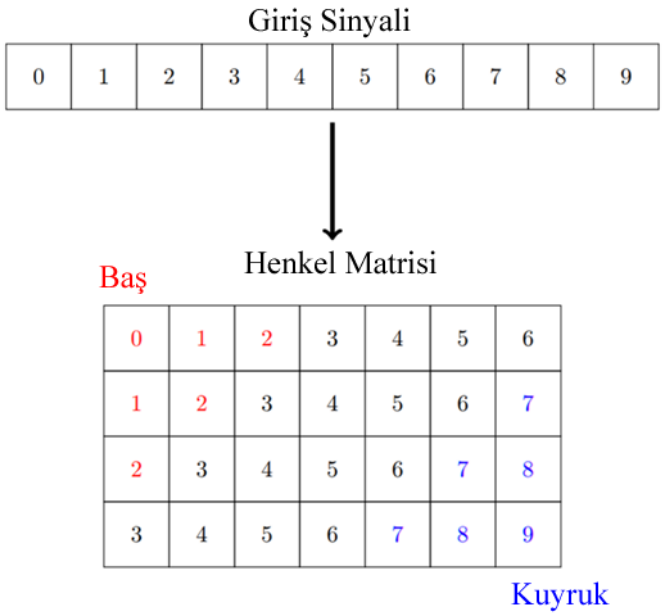
Bu bölümde Tekil Spektrum Analizi (SSA) algoritmasının, bir gerçek zamanlı nesne takibi uygulanmasına uyarlanması üzerinde durulmuştur. SSA algoritması, belirli uzunlukta bir veri kümesine uygulandığı için, sabit uzunlukta bir buffer (ara vektör) kullanılmış ve bu buffer yeni veriler geldikçe sürekli güncellenmiştir. Bu şekilde SSA algoritmasının uygulanmasında, algoritmasının geri-kazanım adımındaki köşegensel ortalama işlemine gerek olmadığı fark edilmiş ve algoritmanın bu kısmı önemli ölçüde sadeleştirilmiştir. Bu bölümü takip eden alt Bölüm 4.1 de SSA algoritmasının adımları açıklanmış ve daha sonra da Bölüm 4.2 de Gerçek Zamanlı SSA algoritması verilmiştir.

4.1. SSA ALGORİTMASI

SSA algoritması dört adımda gerçekleştirilebilir; gömme, Tekil Değer Ayrışımı (SVD), gruplama ve son olarak da geri kazanım.

4.1.1. Gömme Adımı

Gömme adımında, giriş sinyalinden bir Henkel matrisi oluşturulur. Henkel matrisinin her bir sütunu, önceden belirlenmiş sabit uzunlukta bir pencereleme ile elde edilmiş giriş sinyalinin bölümleridir. Bu pencereleme işlemi tırtıl gibi hareket ettiği için, bu metot tırtıl (caterpillar) olarak da isimlendirilmektedir. Eğer giriş sinyalinin boyutu n ve pencere uzunluğu l ise $k = n - l + 1$ olmak üzere Henkel matrisinin satır sayısı l ve sütün sayısı k'dır. Şekil 4.1 gömme adımının nasıl gerçekleştirildiğine dair bir örnek göstermektedir. Bu örnekte, boyutu 10 olan bir giriş sinyali, uzunluğu 4 olan pencere ile örneklenerek 4 x 7 (l x k) boyutunda bir Henkel matrisi oluşturulmuştur.



Şekil 4.1. SSA'da gömme işleminde giriş sinyalinden Henkel matrisinin oluşturulması.

4.1.2. Tekil Değer Ayrışımı (SVD)

SVD adımında yörünge matrisine tekil değer ayrışımı uygulanır. Şayet X Henkel matrisi ise, X^T Henkel matrisinin transpozu olmak üzere, XX^T yörünge matrisi olarak tanımlanır. Bu durumda, Henkel matrisi elde edilen özvektörler yardımı ile aşağıdaki Eşitlik 4.1 de olduğu şekliyle ayrıştırılarak yazılabilir.

$$X = X_1 + X_2 + \cdots + X_m \tag{4.1}$$

Eşitlik 4.1'de her bir X_i matrisine elementer matrisler denir. Elementer matrisler aşağıdaki Eşitlik 4.2'de verildiği gibi hesaplanır;

$$X_i = \sqrt{\lambda_i} U_i V_i^T, i = 1, \dots, m \tag{4.2}$$

Eşitlik 4.2'de U_i, tekil değer ayrışımının özvektörlerini göstermektedir. V_i vektörü ise aşağıdaki eşitlik kullanılarak elde edilir.

$$V_i = \frac{1}{\lambda_i} X^T U_i, i = 1, \dots, m \tag{4.3}$$

Her bir elementer matris X_i için aşağıdaki Eşitlik 4.4 yazılabilir.

$$\|X_i\|^2 = \lambda_i, i = 1, \dots, m \tag{4.4}$$

Dolayısı ile her bir elementer matris SVD'nin yalnızca bir özdeğerine karşılık gelir. Böylece Henkel matrisi X'e olan her bir elementer matrisin katkısı, bu elementer matrislerin karşılık geldiği özdeğerler ile orantılıdır. Küçük özdeğerlere karşılık gelen elementer matrislerin, sinyaldeki gürültüyü temsil ettiği düşünülür.

4.1.3. Gruplama Adımı

Gruplama adımında Eşitlik 4.1'deki gürültüye tekabül ettiği düşünülen elementer matrisler atılır. Bunu aşağıdaki Eşitlik 4.5 ile gösterebiliriz.

$$\hat{X} = X_1 + X_2 + \cdots + X_r \tag{4.5}$$

Eşitlik 4.5'de $r < m$'dir. Gruplama indeksi r, ilk r özdeğer toplamının tüm özdeğerlerin toplamına olan oranına göre belirlenir. Şayet bu oran belirli bir eşik değerin üzerinde ise Eşitlik 4.5'deki toplamın Henkel matrisinin iyi bir yaklaşımı olduğu düşünülür.

4.1.4. Geri Kazanım Adımı

Geri kazanım adımında Henkel matrisinin bir yaklaşımı olan $\hat{X}$ matrisi tek boyutlu sinyale geri dönüştürülür. Bu işlemi Şekil 4.1'deki gömme işleminin tersi olarak düşünebiliriz. Bu geri kazanım işlemi, Henkel matrisi üzerinde köşegensel ortalama alınarak gerçekleştirilir. Köşegensel ortalama üç adımda gerçekleştirilir; matrisin baş, merkez ve kuyruk kısmının ortalanması. Bu ortalama işlemi aşağıdaki Eşitlik 4.6 ile hesaplanır.

$$b_t = \begin{cases} \frac{1}{t+1}\sum_{i=1}^{t+1} a_{i,t-i+2}\,, if\ 0 \le t < L^* - 1 \\ \frac{1}{L^*}\sum_{i=1}^{L^*} a_{i,t-i+2}\,, if\ L^* - 1 \le t < K^* \\ \frac{1}{N-t}\sum_{i=t-K^*+2}^{N-K^*+1} a_{i,t-i+2}\,, if\ K^* \le t < N \end{cases} \quad (4.6)$$

Eşitlik 4.6'da $1 \le i \le L$ ve $1 \le j \le K$ olmak üzere $a_{i,j}$, $\hat{X}$ matrisinin elemanlarını ve $1 \le t \le N$ olmak üzere b_t geri-kazanılmış tek-boyutlu zaman serisinin elemanlarını göstermektedir.

SSA ile filtrelemenin sonucunu iki parametre ile belirlenmektedir. Pencereleme uzunluğu L ve gruplama indeksi r. Gruplama indeksinin nasıl belirleneceği yukarıda açıklanmıştır. Genel olarak, ilk bir veya iki özdeğer oran olarak tüm özdeğerlerin çok büyük bir kısmını oluşturmaktadır. Ancak pencereleme uzunluğu L'nin nasıl belirleneceğine dair kesin bir kural olmayıp deneme yanılma yoluyla bulunmaktadır. SSA algoritması Çizelge 4.1' de verilmiştir. Bu algoritmada gruplama indeksi r, gruplama işleminden önce algoritmanın içinde belirlenmiştir. Bir parametre olarak algoritmaya girişte verilmedi. Dolayısı ile algoritmanın tek giriş parametresi pencere uzunluğu *L* dir.

Çizelge 4.1. SSA algoritması.

Giriş	:	$veri$: Uzunluğu *n* olan tek boyutlu veri vektörü.
Giriş	:	L : Pencere uzunluğu
Çıkış	:	f_veri : Filtrelenmiş veri vektörü
Adım 1	:	$k \leftarrow n - l + 1$
		//Adım 1:Henkel Matrisinin Oluşturulması (Gömme)
Adım 2	:	**for** $i = 0; i < k; i++$ **do**
Adım 3	:	**for** $j = 0; j < l; j++$ **do**
Adım 4	:	$offset \leftarrow i + j$
Adım 5	:	$X(j,i) \leftarrow veri[offset]$

		//Adım 2:Tekil Değer Ayrışımı (SVD)
Adım 6	:	$[U, \Sigma, V] \leftarrow SVD(XX^T)$
Adım 7	:	$\lambda_i \leftarrow \Sigma_{ii}$ *//SVD'nin özdeğerleri*
		//Adım 3: Gruplama
Adım 8	:	$lambda\_trh \leftarrow 0.9$
Adım 9	:	$lambda\_sum \leftarrow \sum_{j=0}^{l} \lambda_j$
Adım 10	:	**for** $i = 0; i < l; i++$ **do**
Adım 11	:	$lambda\_ratio \leftarrow \frac{1}{lambda\_sum} \sum_{j=0}^{i} \lambda_i$
Adım 12	:	**if** $lamda\_ratio \geq lamda\_trh$ **then**
Adım 13	:	$r \leftarrow i + 1$
Adım 14	:	**break;**
Adım 15	:	$\hat{X}$ tüm girdileri sıfır ile doldurulmuş l *x* k boyutunda matris
Adım 16	:	**for** $i = 0; i < r; i++$ **do**
Adım 17	:	$V_i \leftarrow \frac{1}{\sqrt{\lambda_i}} X^T U_i$
Adım 18	:	$X_i \leftarrow \sqrt{\lambda_i} U_i V_i^T$
Adım 19	:	$\hat{X} \leftarrow \hat{X} + X_i$
	:	*//Adım 4: Geri Kazanım (Bkz. Eşitlik 4.6)*
Adım 20	:	$ll \leftarrow min(l, k)\,, kk \leftarrow max(\,l, k)$
Adım 21	:	Uzunluğu n olan *f_veri* vektörünün tüm girdileri sıfır ile doldur.
		//Baş kısmın köşegensel ortalaması
Adım 22	:	**for** $i = 0; i < ll - 1; i++$ **do**
Adım 23	:	$scl \leftarrow 1/(i + 1)$
Adım 24	:	**for** $j = 0; j < i + 1; j++$ **do**
Adım 25	:	$f\_veri[i] \leftarrow scl * \hat{X}(j, i - j)$
		//Orta kısmın köşegensel ortalaması
Adım 26	:	$scl \leftarrow 1/ll$
Adım 27	:	**for** $i = ll - 1; i < kk; i++$ **do**
Adım 28	:	**for** $j = 0; j < ll; j++$ **do**

Adım 29	:	$f\_veri[i] \leftarrow scl * \hat{X}(j, i-j)$
		//Kuyruk kısmının köşegensel ortalaması
Adım 30	:	**for** $i = kk; i < n; i++$ **do**
Adım 31	:	$scl \leftarrow 1/(n-i)$
Adım 32	:	**for** $j = i - kk + 1; j <= n - kk; j++$ **do**
Adım 33	:	$f\_veri[i] \leftarrow scl * \hat{X}(j, i-j)$

4.2. GERÇEK ZAMANLI SSA ALGORİTMASI

SSA'nın gerçek zamanlı uygulanması sabit boyutlu bir buffer kullanılarak gerçekleştirilmiştir (Şekil 4.2). İlk olarak, önceden belirlenmiş uzunluğa göre bufferın dolması beklenir. Dolayısı ile bu metot birçok gerçek zamanlı uygulanan filtrede olduğu gibi, bir zaman ertelemesine ihtiyaç duymaktadır. Buffer dolduktan sonra filtreleme işlemine başlanmıştır. Yeni gelen veriler bufferın son elemanına yazılmış ve ilk elemanı silinmiştir. SSA algoritması ile filtrelenen bufferdan son eleman filtrelenmiş veri olarak çekilmiştir. Dolayısı ile bu işlem SSA algoritmasında, Eşitlik 4.6 ile verilen köşegensel ortalama işlemine gerek duymamaktadır (Çizelge 4.1, adım 20-33). Şayet bufferın SSA algoritmasının giriş sinyali olduğunu düşünürsek, filtrelenmiş veri yaklaşık Henkel matrisinin son girdisidir. Gerçek zamanlı olarak bu filtrelenmiş veri, her bir kare (frame) için aşağıdaki Eşitlik 4.7 ile elde edilir.

Şekil. 4.2. Sabit boyutlu buffer.

$$f = \hat{X}(l,k) \tag{4.7}$$

Eşitlik 4.7'de f, her bir karedeki tek bir filtrelenmiş veriyi göstermektedir. Dolayısı ile gerçek zamanlı SSA algoritmasında Eşitlik 4.6'daki karmaşık geri kazanım işlemine gerek duyulmadığı anlaşılmış oldu. Çizelge 4.1 ile verilen SSA algoritmasındaki geri kazanım algoritmasının (Adım 20-33) yerine sadece Eşitlik 4.7'yi kullanmak yeterli olacaktır. Dolayısı ile gerçek zamanlı bir uygulamada SSA algoritmasının girişi vektör (buffer), çıkışı ise sadece tek bir veridir. Bu algoritmayı Gerçek Zamanlı Tekil Spektrum Analizi (RT-SSA) olarak isimlendirebilir. RT-SSA metodunun gerçek zamanlı veri filtrelemede kullanımını ifade eden algoritma Çizelge 4.2'de verilmiştir.

Çizelge 4.2. RT-SSA Algoritması ile gerçek zamanlı filtreleme.

Giriş		*l :* Pencereleme uzunluğu
Giriş		*bs* : Buffer uzunluğu
Giriş		*buffer :* Buffer vektörü
Giriş		*frame:* Herhangi bir çerçevede elde edilen pozisyon verisi
Çıkış		*f_frame*: Filtrelenmiş pozisyon verisi
		//Buffer vektörünü doldurma
Adım 1	:	**if** $buffer.size() < bs$ **then**
Adım 2	:	$buffer.push\_back(frame)$
		//Buffer vektörü dolduğunda filtrelemeye başla
Adım 3	:	**if** $buffer.size() == bs$ **then**
Adım 4	:	$buffer.erase(buffer.begin())$
Adım 5	:	$buffer.push\_back(frame)$
Adım 6	:	$f\_frame \leftarrow RT\_SSA(buffer, l)$

BÖLÜM 5

MATERYAL VE YÖNTEM

Bu çalışmada nesne tanımlama bir şablon eşleştirme problemi olarak ele alınmaktadır. Şablon eşleştirmede yerine getirilmesi gereken temel görev, verilen model görüntüyü herhangi bir sahne içerisinde konumlandırabilmektir. Nesne tanımlama, model görüntüyü sahne içerisinde mümkün olduğunca doğru bir şekilde bir kutu içerisine alma görevi olarak tanımlanır. Dolayısı ile nesne tanımlama algoritmasının gürbüzlüğü bu kutu içine alma görevinin ne ölçüde doğru yapıldığı ile orantılıdır. Bu nedenden dolayı, çalışmada Kesişimin Birleşime Oranı (IoU) metriği kullanılmıştır [40].

Diğer taraftan, tanımlayıcıların gürbüzlüğünün ölçülmesi için literatürde Doğru Eşleşme Oranı (CMR) sıklıkla tercih edilmiştir [1]. Ancak, şayet tespit edilen kutunun doğruluğunda (IoU metriğinde) herhangi bir artış yok ise, CMR metrik değerindeki artışın bir anlamı yoktur. Dolayısı ile bu çalışmada, değerlendirme metriği olarak, her ikisinin Eşitlik 5.1'de verildiği şekilde çarpımın kullanılmasına karar verilmiştir.

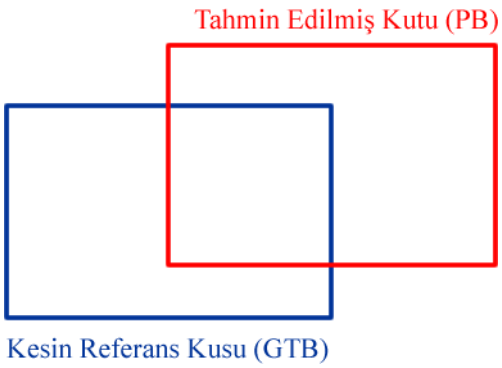
$$Metrik = IoUxCMP \tag{5.1}$$

5.1. KESİŞİMİN BİRLEŞİME ORANI (IoU)

IoU metriğini kullanarak, herhangi bir nesne tanımlama algoritmasının performansını ölçebilmek için, gerçek referans homografilerinin veri kümeleri ile birlikte verilmiş olması gerekir. Bu takdirde, gerçek referans kutusunu elde edebiliriz. Bu takdirde IoU aşağıdaki Eşitlik 5.2 ile hesaplanır;

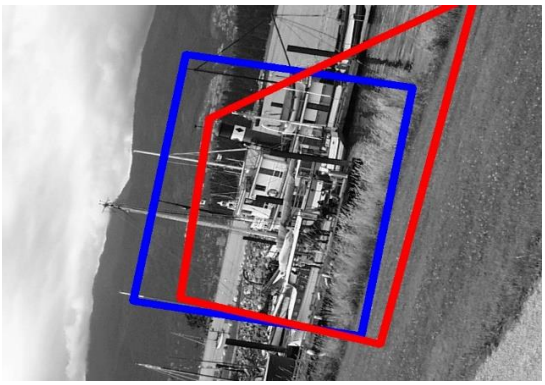
$$IoU = \frac{GTB \cap PB}{GTB \cup PB} \tag{5.2}$$

Eşitlik 5.2'de GTB, kesin referans homografi matrisi ile elde edilmiş Kesin Referans Kutusunu (Ground Truth Box) ve PB ise nesne tanımlama algoritmasının verdiği homografi matrisi ile elde edilmiş Tahmin Edilmiş Kutuyu (Predicted Box) göstermektedir (Şekil 5.1).



Şekil 5.1. IoU metriği GTB ile PB bölgelerinin kesişimlerinin birleşimlerine oranı olarak ifade edilmektedir.

Homografi matrisleri ile elde edilen kutular tam olarak bir dikdörtgen olmadıkları için, bir poligon olarak değerlendirilmelidir (Şekil 5.2). Bu poligonların alanları, birleşimlerinin ve şayet kesişiyorlar ise kesişimlerinin alanları, Clipper kütüphanesinin C++ kodu kullanılarak hesaplanmıştır [41]



Şekil 5.2. Oxford veri kümesinden zayıf bir eşleşme örneği.

5.2. DOĞRU EŞLEŞME ORANI (CMR)

Gerçek referans homografi matrisleri kullanılarak, model görüntünün izdüşümü (projeksiyonu) sahne üzerinde doğru bir şekilde elde edilmiş olur. Dolayısıyla CMR

metriğini hesaplayabiliriz. p_m ve p_s sırasıyla model ve sahne görüntüleri üzerinde eşleşmiş anahtar noktaların piksel pozisyonlarını göstermek üzere, aşağıda verilen Eşitsizlik 5.3 sağlandığında, bu eşleşme doğru bir eşleşme olarak kabul edilir.

$$\|p_s - p_m H_{gr}\| < thr \tag{5.3}$$

Eşitsizlik 5.3'de H_{gr} verilen gerçek referans homografi matrisini temsil etmektedir. Bu çalışmada eşik değer thr, 2 piksel olarak alınmıştır.

Doğru bir şekilde eşleşen anahtar nokta çiftlerinin kümesini içler (inliers) ve yanlış olarak eşleşen anahtar nokta çiftlerinin kümesini ise dışlar (outliers) olarak isimlendirebiliriz (Şekil 5.3). Bu takdirde CMR (veya içlerin oranı), aşağıdaki Eşitlik 5.4 ile hesaplanır.

$$CMR = \frac{\#İçler}{\#(İçler \cup Dışlar)} \tag{5.4}$$

Şekil 5.3. Oxford veri kümesinde eşleşen bir çift görüntü örneği.

Şekil 5.3'de soldaki resim model görüntüyü, sağdaki resim aynı görüntünün ışıklandırması değiştirilerek elde edilmiş sahne görüntüsünü temsil etmektedir. Burada maviler doğru eşleşmeyi, kırmızılar ise yanlış eşleşmeyi göstermektedir.

5.3. ÖZNİCELİK EŞLEŞME ALGORİTMASI

Öznicelik eşleştirme için, OpenCV'nin BruteForce eşleştirme fonksiyonu kullanılabilir. Bu fonksiyon model görüntü üzerinde tanımlanmış bir öznicelik vektörü

ile sahne görüntüsü üzerinde tanımlanmış tüm öznicelik vektörleri arasındaki uzaklığı hesaplayıp, en küçük uzaklığa sahip çifti, eşleşen çift olarak belirlemektedir. BruteForce fonksiyonunda iki vektör arasındaki uzaklık L1 veya L2 normu ile belirlenebilmektedir. Standart (default) olarak L2 normu kullanılmaktadır. Ancak çalışma prensibi gereği bu metot, veri kümeleri üzerinde hesaplanan anahtar-noktaların çok fazla olabilmesi dolayısıyla, hesaplama zamanı olarak çok yavaş kalabilmektedir. Bu sebepten dolayı, OpenCV'nin FlannBasedMatcher algoritması bu çalışmada eşleşme algoritması olarak tercih edilmiştir. FLANN (Fast Library for Approximated Nearest Neighbors) algoritması efektif bir şekilde k-NN algoritması ile verileri gruplayarak, doğruluktan fazla ödün vermeden yaklaşık olarak hızlı bir şekilde aralarında en kısa mesafe olan iki vektörü tespit edebilmektedir [42].

FLANN algoritması aralarında en küçük uzaklık olan iki anahtar nokta çifti vermektedir (2-nearest neighbor). Dışlar kümesinin elemanlarını (yanlış eşleşmeleri) elemek için bu iki çiftin sahip oldukları uzaklıkların birbirine oranı kullanılır. d_1 ve d_2 sırasıyla ilk ve ikinci çiftin uzaklıkları olduğu kabul edilirse, bu takdirde aşağıda verilen Eşitsizlik 5.5 sağlandığında, ilk çift iyi bir eşleşme olarak kabul edilir.

$$\frac{d_1}{d_2} < thr \tag{5.5}$$

Eşitsizlik 5.5'den görüleceği üzere, şayet ikinci çiftin uzaklığı birinci çifte çok yaklaşırsa, birbirlerine olan oranları da bire yaklaşacaktır. Eğer oran bire yaklaşıyor ise bu çift dışlar kümesinin bir elemanı olarak değerlendirilir. Çünkü öznicelik uzayında genellikle dışlar kümesinin üyeleri birbirlerine çok benzer bir şekilde yer almaktadır. Bu çalışmada eşleştirme algoritmasında dışlar kümesinin elemanlarını tespit edebilmek için eşik değeri 0, 8 olarak belirlenmiştir. Bu eşik değer 1'e doğru yaklaştıkça eşleşen çift sayısı azalmakta, sıfıra doğru yaklaştıkça bu değer artmaktadır.

Eşleşen çiftler belirlenip, dışlar elendikten sonra, OpenCV kütüphanesinin findHomography fonksiyonu içinden RANSAC algoritması seçilerek, tanımlama için kullanılacak olan homografi matrisi elde edilmiştir. Daha sonra da bu homografi

matrisi ile OpenCV'nin perspectiveTransform fonksiyonu kullanılarak, model görüntünün köşelerinin sahne üzerindeki izdüşümü hesaplanmıştır (Şekil 5.2).

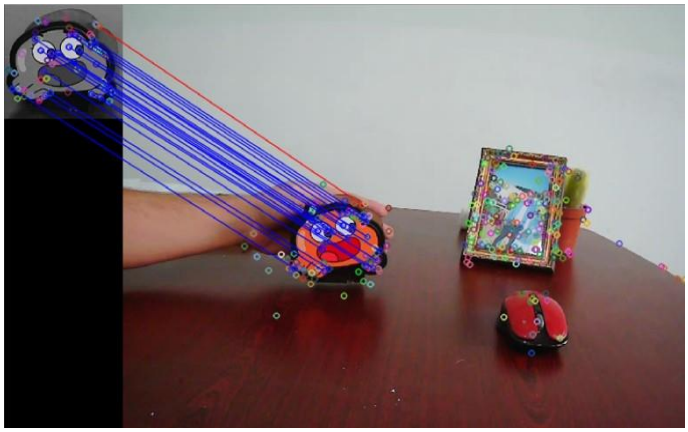
5.4. VERİ KÜMELERİ

Performans testleri Oxford [43] ve HPatches [44] veri kümeleri üzerinde gerçekleştirilmiştir. Oxford veri kümesi, nesne tanımlama algoritmalarını; bakış açısı, dönme, ölçeklendirme, bulanıklaştırma, ışıklandırma, JPEG kompresyonu gibi farklı fotometrik ve afin ötelemelerden kaynaklanan değişikliklere sahip gerçek görüntüler üzerinde test edebilmemize olanak sağlamaktadır. Veri kümesinin her bir alt kümesi, ilki model, diğerleri de model görüntüden görüntü transformasyonları ile farklılaştırılmış toplam 6 görüntüden oluşmaktadır. Dolayısı ile her bir altküme gerçek referans kutularının hesaplanabilmesi için önceden manüel olarak hesaplanmış 5 adet gerçek referans homografi matrisine sahiptir. Bu veri kümesi toplamda 8 alt kümeye sahip olduğu için, tamamı 48 görüntüden oluşmaktadır.

Diğer taraftan, HPatches veri kümesi, Oxford veri kümesine göre çok daha fazla görüntüye sahiptir. Bu veri kümesi görüntü ötelemelerine göre iki alt kümeye ayrılmıştır; Bunlar ışıklandırma ve bakış açısı değişimi alt kümeleridir. Işıklandırmadaki ve bakış açılarındaki değişim alt kümeleri ayrıca sırasıyla 57 ve 59 altkümeye daha ayrılmıştır. Oxford veri kümesinde olduğu gibi bu alt kümelerin de her biri 6'şar görüntüden oluşmaktadır. Toplamda bu veri kümesi HPatches 696 görüntüden oluşmakta olup, Oxford veri kümesine göre daha genel bir sonuç elde etmek mümkündür.

5.5. NESNE TAKİP VİDEOSU

Bu çalışmada veri kümelerinin yanı sıra nesne tanımlama algoritmasının gürbüzlüğünü test etmek için kendi kaydettiğimiz bir video ve model görüntü kullanılmıştır. Video Everest SC-HD03 webcam ile 640x480 çözünürlüğünde 20 fps hızında kaydedilmiştir. Toplam uzunluğu 1439 çerçeve ve 72 saniyedir. Videoda model görüntü (bir kalemlik) el ile hareket ettirilmiştir. Kalemlik kameradan uzaklaştırılmış, yakınlaştırılmış ve bu esnada farklı açılarda döndürülmüş ve bükülmüştür (Şekil 5.4 ve 5.5).



Şekil 5.4. Model görüntünün (sol üst köşe) videodaki nesne ile eşleşmesi.

Şekil 5.5. Tanımlanmış modelin video içerisinde takibi.

Videonun her bir çerçevesi için gerçek referans homografi matrisleri mevcut olmadığı için IoU ve CMR metrikleri, algoritmanın gürbüzlüğünü ölçmek için kullanılamaz. Bu durumda hızı temel alarak bir düzgünlük (smoothness) metriği kullanacağız [45].

5.6. HIZ TEMELLİ YÖRÜNGE DÜZGÜNLÜK METRİĞİ

P_1, P_2, P_3 ve P_4 bir dikdörtgensel poligonun iki boyutlu piksel koordinatları olsun. Bu koordinatlar eşleşme algoritması tarafından sağlanmaktadır. Bu takdirde poligonun merkez noktasının koordinatları aşağıda verilen Eşitlik 5.6 ile hesaplanır.

$$M = \frac{1}{4}(P_1 + P_2 + P_3 + P_4) \tag{5.6}$$

Merkez noktanın koordinatlarının görüntü orijinine olan uzaklığı (normu) aşağıda verilen Eşitlik 5.7 ile hesaplanır.

$$x = \sqrt{M_x^2 + M_y^2} \tag{5.7}$$

Sonuç olarak, modelin hareket yörüngesinin aşağıda verilen Eşitlik 5.8 ile tek boyutlu sinyalini elde etmiş oluruz.

$$X(t) = [x_1, x_2, \dots, x_n] \tag{5.8}$$

Şayet kameranın örnekleme frekansı f ile gösterirsek, videonun birbirini takip eden iki çerçevesi arasındaki zaman aralığı, $\delta t = 1/f$ olacaktır. Bu durumda hareket yörüngesinin hızı ayrık olarak aşağıda verilen Eşitlik 5.9 ile hesaplanır.

$$\dot{X}(t) = (x_{i+1} - x_i)/\delta t \tag{5.9}$$

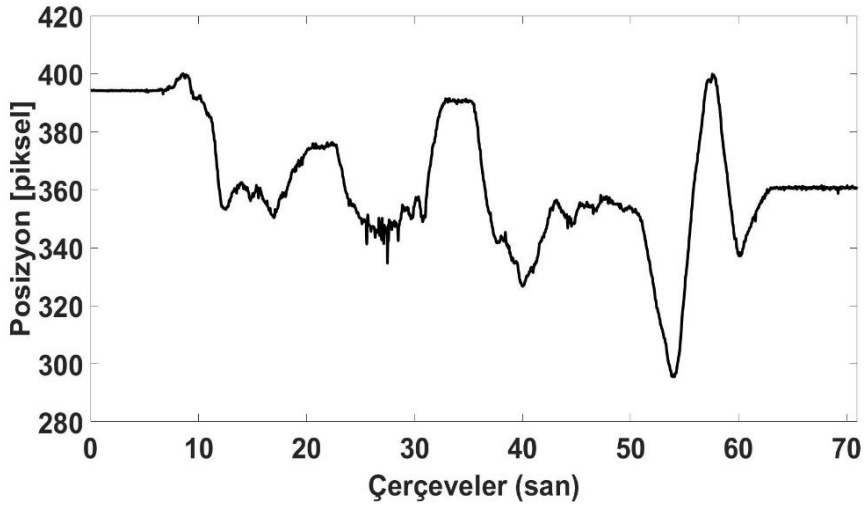
Yörünge hızının standart sapmasını ve ortalamasını kullanarak üst ve alt eşik değeri aşağıdaki Eşitlik 5.10 ile hesaplanır.

$$sp_{trh} = \mu_{\dot{X}} \pm t\sigma_{\dot{X}} \tag{5.10}$$

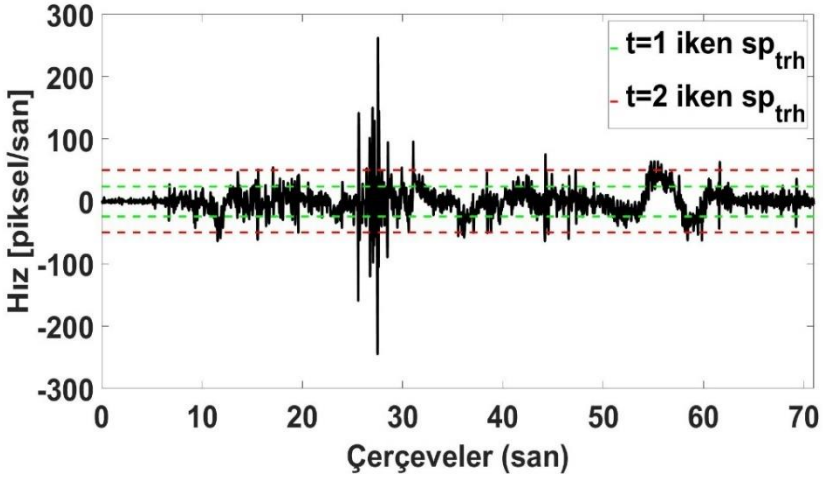
Eşitlik 5.10'da, $\mu_{\dot{X}}$ve $\sigma_{\dot{X}}$ sırasıyla yörünge hızının ortalamasını ve standart sapmasını göstermektedir. t ise kullanıcı tarafından belirlenen bir tolerans değeridir.

Şekil 5.6 ve 5.7, videodaki hareket ve hız yörüngelerini göstermektedir. Bu yörünge OpenCV kütüphanesindeki orijinal SIFT algoritmasının standart parametreleri ile elde edilmiştir. Söz konusu yörünge $\mu_{\dot{X}} \approx 0.47$ ve $\sigma_{\dot{X}} \approx 24$ değerlerine sahiptir. Dolayısıyla, $t = 1$ için $sp_{trh} \approx \pm 24$ ve $t = 2$ için $sp_{trh} \approx \pm 48$ olmaktadır. Bu çalışmada sp_{trh} eşik değeri ± 50 olarak alınmıştır. Şayet herhangi bir çerçevedeki hız bu sınırın dışına çıkarsa, bu çerçevedeki eşleşme dışlar kümesinin (başarısız eşleşme)

bir üyesi olarak değerlendirilmiştir. Şekil 5.6'daki 1439 elemanlı yörüngenin dışlar kümesi 50 eleman olarak gerçekleşmiştir.



Şekil 5.6. Uygulama videosundaki modelin hareket yörüngesi.



Şekil 5.7. Videodaki hız yörüngesi ve farklı tolerans değerlerine göre eşik değer aralıkları.

5.7. DENEYSEL YÖNTEMİN DİĞER AYRINTILARI

IoU ve CMR metrikleri, veri kümelerindeki her bir eşleşen görüntü çifti için hesaplanmıştır. Dolayısı ile her bir parametre veya algoritma değişikliği için ortalama IoU ve CMR metrikleri verilmiştir.

Model ve sahne görüntüsü için, SIFT algoritmasının (anahtar noktaların belirlenmesi ve özniceliklerin hesaplanması) ve öznicelik eşleşme algoritmasının hesaplama zamanları ayrı ayrı ölçülmüştür. Hesaplama zamanı OpenCV'nin getTickFrequency ve getTickCount fonksiyonları ile milisaniye olarak ölçülmüştür. Metrik hesaplamasında olduğu gibi her bir eşleşen görüntü için ölçülen zamanların ortalaması alınmıştır.

Kaynak kodları Ubuntu v20.04 işletim sistemi üzerinde GCC-derleyici ile release modda derlenip, Intel i9-10850K CPU @3.60 GHZ ve 16 GB Ram özelliklerinde bir bilgisayar üzerinde çalıştırılmıştır.

BÖLÜM 6

DENEY SONUÇLARI

Bu bölümde ilk olarak SIFT algoritmasının yanı sıra literatürde en popüler lokal tanımlayıcılar olan SURF [46] ve ORB [47] metotlarının performansları karşılaştırılmıştır. Sonra da sırasıyla Bölüm 2, 3 ve 4'de verilen algoritmaların deney sonuçları, genel olarak Bölüm 5'de ifade edilen deneysel yöntem çerçevesinde verilmiştir. Bu çalışmada SIFT algoritması için önerilen algoritmalar, OpenCV 4.5.5 kütüphanesinin kaynak dosyalarında bulunan 'sift.dispatch.cpp' ve 'sift.simd.hpp' dosyaları üzerinde değişiklik yapılarak uygulanmıştır.

6.1. SIFT, SURF ve ORB ÖZNİCELİK TANIMLAYICILARININ PERFORMANSLARININ KARŞILAŞTIRILMASI

Bu bölümde OpenCV 4.5.5 kütüphanesinde yer alan SIFT, SURF (Speeded-up Robust Feature) ve ORB (Oriented FAST [48] and Rotated BRIEF [49]) lokal öznicelik tanımlama algoritmalarının performansları, algoritmaların orijinal parametrelerinde ve kodlarında herhangi bir değişiklik yapılmadan bir birleriyle karşılaştırılmıştır.

SURF ve ORB öznicelik tanımlayıcılarının her ikisi de SIFT algoritmasının hesaplama hızını geliştirmek için ortaya çıkmıştır. Her iki metot da SIFT algoritmasında hesaplama yoğunluğunu sadeleştirerek, hesaplama hızını geliştirmiştir. Ancak gürbüzlük ile hız arasında ters orantı vardır. Gürbüzlük, bir algoritmanın hemen hemen tüm görüntü değişimleri (bulanıklaşma, uzaklaşma / yaklaşma, dönme, ışıklandırma vb. değişimler) karşısında değişmez kalabilmesi, yani başka bir ifade ile tanımlamayı kaybetmemesidir. Ancak bu yüksek gürbüzlük hesaplama yoğunluğunu beraberinde getirmektedir. Hesaplamadaki sadeleştirmeler de genellikle gürbüzlükten kayıp olarak karşımıza çıkmaktadır.

SURF algoritması, SIFT algoritmasında GSSP'nin oluşturulmasında Gauss konvolüsyonun neden olduğu hesaplama yoğunluğunu ortadan kaldırmak için, integral görüntü (integral image or summed area table) metodu ile hızlandırılmış tekrarlı kutu filtresini kullanmıştır. SURF metodundaki anahtar noktaları konumlandırma, anahtar noktalara yönelim atama ve öznicelikleri tanımlama yöntemleri de SIFT'den farklıdır. Anahtar noktaları konumlandırmak için DoG uzayındaki ekstremum noktalar yerine Hessien matrisinin determinantının maksimumlarına bakılmıştır. Anahtar noktaların yönelimlerini belirlemek ve öznicelik vektörlerini hesaplamak için ise HOG metodu yerine Haar wavelet dönüşümü kullanılmıştır.

Diğer taraftan ORB öznicelik tanımlayıcısı anahtar noktaların hesaplanmasında FAST (Features from Accelerated Segment Test) ve özniceliklerin hesaplanmasında BRIEF (Binary Robust Independent Elementary Features) metotlarını kullanmaktadır. ORB anahtar noktaları konumlandırmak için SIFT veya SURF metodunda olduğu gibi bir GSSP hesaplamasına girmeden, yalnızca giriş görüntüsü üzerinde anahtar noktaları konumlandırmaktadır. Bu da normal olarak büyük bir hız kazancı sağlamakla beraber, algoritmanın bulanıklaşma ve uzaklaşma/ yakınlaşmaya karşı gürbüzlüğünü büyük ölçüde düşürmektedir. FAST'da anahtar noktalar bir pikselin dairesel komşularına bakarak, makine öğrenmesi yöntemi olan karar verme ağacı (decision tree) ile belirlenmiştir. Bu karar verme ağacı görüntü üzerindeki köşeleri (corners) sınıflandırmaktadır. Sonuç olarak da bu sınıflandırmadaki köşelerin hangilerinin anahtar nokta olarak seçileceğine karar vermektedir.

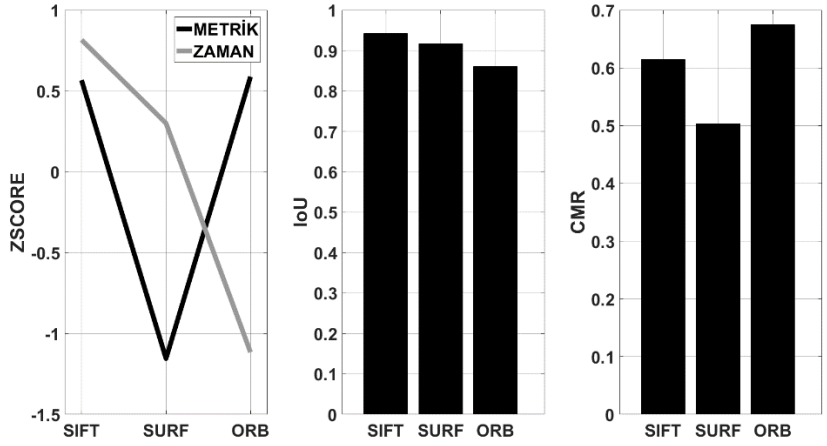
ORB metodunun geliştirilmesindeki temel düşünce nesne tanımlama algoritmasının cep telefonları gibi sınırlı hafızaya sahip ve düşük güç tüketimi gerektiren cihazlarda çalışmasını sağlamaktır. SIFT algoritmasında her bir anahtar nokta, her bir girdisi bir byte olan 128 uzunluğunda bir öznicelik vektörüne sahiptir. Dolayısı ile bir öznicelik, 128x8=1024 bit hafızada yer kaplamaktadır. Bu değer SURF metodunda, öznicelik vektörü uzunluğu 64 olduğu için 64*8=512 bittir. ORB de ise özniceliklerin (Binary Robust Independent Elementary Features) her bir girdisi binary (0 veya 1) olarak hesaplanmakta, bu da hafıza da kapladığı yer açısından önemli bir avantaj sağlamaktadır. BRIEF de öznicelik vektör boyutu standart (default) olarak 256 olup her bir öznicelik vektörü 256 bit hafızada yer kaplamaktadır.

SIFT ve SURF öznicelik tanımlama metotları ile eşleştirme gerçekleştirilirken OpenCV algoritmasındaki FlannBasedMatcher eşleştirme fonksiyonunda L2 norm kullanılmıştır. ORB ile eşleştirme gerçekleştirilirken, tanımlanan anahtar noktaların sayısı az ve özniceliklerin girdileri binary olduğu için BruteForce algoritmasıyla Humming mesafe (Humming distance) metriği kullanılmıştır. Test sonuçlarını gösterirken şekiller üzerinde verilen kısaltmaların açıklamaları Çizelge 6.1'de verilmiştir.

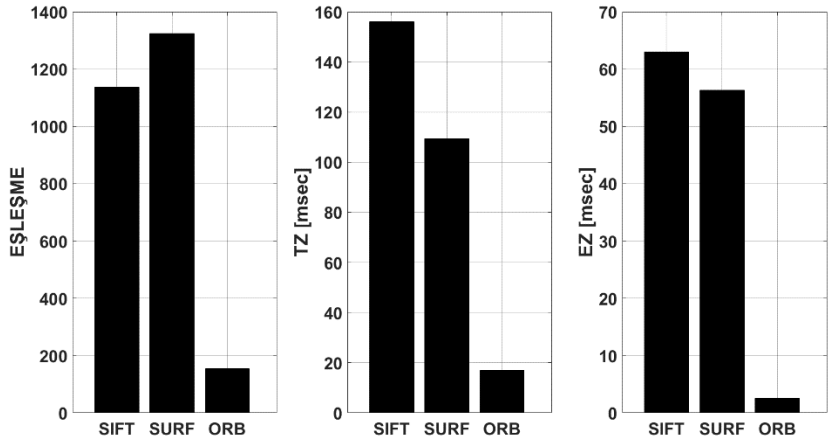
Çizelge 6.1. Sonuçlarda kullanılan kısaltmalar ve ifadeler.

Kısaltmalar ve İfadeler	Açıklamaları
IoU	Kesişimin Birleşime Oranı
CMR	Doğru Eşleşme Oranı
METRİK	IoU*CMR
TZ	Tanımlama zamanı (anahtar noktalar ve öznicelikler)
EZ	Eşleşme zamanı
ZAMAN	TZ+EZ
EŞLEŞME	Eşleşen anahtar noktaların sayısı
V-DIŞLAR	Videodaki dışlar
ZSCORE	z-score veya standart skor

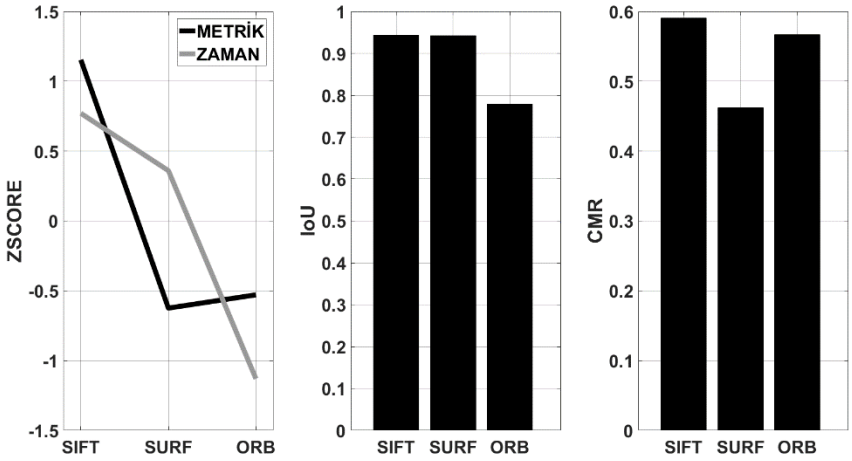
Şekil 6.1 ve 6.2 öznicelik tanımlama algoritmalarının Oxford veri kümesi üzerinde, Şekil 6.3 ve 6.4 HPatches veri kümesi üzerinde, Şekil 6.5 ve 6.6 ise nesne takip videosu üzerindeki test sonuçlarını göstermektedir. Veri kümeleri ve video üzerindeki sonuçları incelediğimizde, test edilen algoritmalardan gürbüzlük açısından en güçlü algoritmanın SIFT, en zayıf algoritmanın ise ORB olduğu, hız söz konusu olduğunda ise tam tersine en yavaş algoritmanın SIFT ve en hızlı algoritmanın ORB olduğu anlaşılmaktadır. Test edilen algoritmaların hesaplama kompleksliği göz önüne alındığında bu sonuçlar gürbüzlük ile hız arasındaki ters orantıyı göstermektedir.



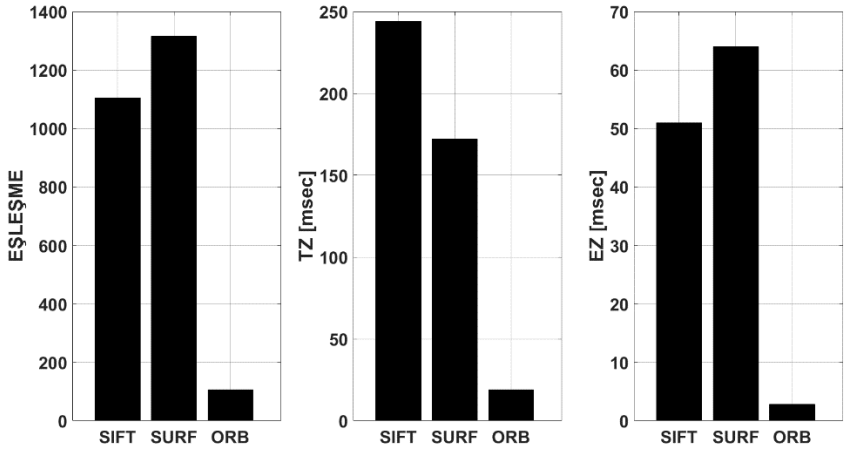
Şekil 6.1. SIFT, SURF ve ORB metotlarının performanslarının Oxford veri kümesi üzerinde karşılaştırılması (ZSCORE, IoU ve CMR).



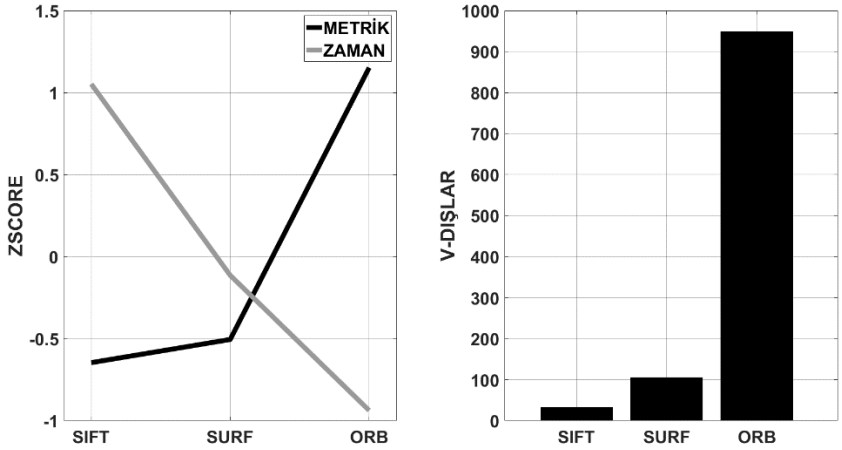
Şekil 6.2. SIFT, SURF ve ORB metotlarının performanslarının Oxford veri kümesi üzerinde karşılaştırılması (EŞLEŞME, TZ ve EZ).



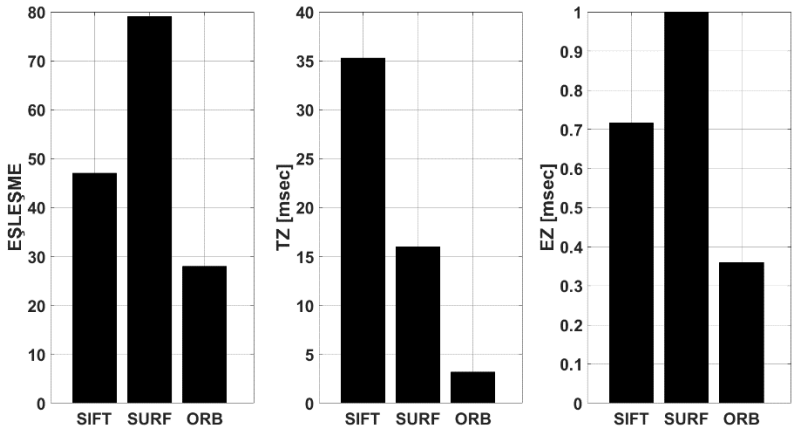
Şekil 6.3. SIFT, SURF ve ORB metotlarının performanslarının HPatches veri kümesi üzerinde karşılaştırılması (ZSCORE, IoU ve CMR).



Şekil 6.4. SIFT, SURF ve ORB metotlarının performanslarının HPatches veri kümesi üzerinde karşılaştırılması (EŞLEŞME, TZ ve EZ).



Şekil 6.5. SIFT, SURF ve ORB metotlarının performanslarının nesne takip videosu üzerinde karşılaştırılması ((ZSCORE ve V-DIŞLAR).



Şekil 6.6. SIFT, SURF ve ORB metotlarının performanslarının nesne takip videosu üzerinde karşılaştırılması (EŞLEŞME ve TZ ve EZ).

6.2. GAUSS ÖLÇEK UZAY PRAMİDİ ALGORİTMASI İÇİN DENEY SONUÇLARI

SIFT algoritmasının performansı üzerinde, giriş görüntüsünün sahip olduğu farz edilen bulanıklık seviyesinin etkisini test edebilmek amacıyla Çekirdek Görüntü Hesaplama algoritmasındaki GaussKonvolüsyon fonksiyonu iptal edilmiştir (Çizelge 2.1). Dolayısı ile bu değişiklikle Çizelge 2.1'deki algoritmanın tek fonksiyonu, giriş görüntüsünün boyutunu iki katına çıkarmaktır. Sonuçları gösterirken bu değişikliği Mod-1 ile ifade edeceğiz.

GSSP uzayının hesaplanmasında, Erf'yi temel alarak belirlenen Gauss fonksiyonundan örneklem uzunluğunun veya kernel boyutunun etkisini gözlemleyebilmek amacıyla, OpenCV'nin GausssianBlur fonksiyonu kendi TDGC algoritmamızla yer değiştirilmiştir. Sonuçları gösterirken bu değişikliği Mod-2 ile ifade edeceğiz. Bu noktada, Gauss fonksiyonunun ayrılabilirlik özelliğinden dolayı, TDGC'nin, sırası ile ilk önce satır, daha sonra da sütunlara tek boyutlu olarak uygulanabileceği göz önünde bulundurulmalıdır. Dolayısı ile satır ve sütün indekslerinin birbirinden bağımsızlığı sebebiyle TDGC paralel olarak uygulanabilir. Paralel TDGC algoritmasının uygulanması OpenCV'nin parallel_for fonksiyonu kullanılarak gerçekleştirilmiştir. Diğer taraftan, Gauss dağılım fonksiyonu simetrik olduğundan, yalnızca yarısı örneklenip, simetrik konvolüsyon uygulanmıştır. Bu durum konvolüsyon hızını artıran önemli bir etkendir.

Metrik ve hesaplama zamanı değerleri farklı skalalarda (ölçeklerde) olduğundan, her iki değeri de aynı grafikte gösterebilmek için bu değerlerin zscore veya standart skorundan faydalanılmıştır. Zscore farklı skalalardaki verileri bu verilerdeki değişimi koruyarak aynı skalaya çevirmektedir.

OpenCV'nin SIFT algoritmasının bazı önemli parametreleri ve aldığı varsayılan değerler Çizelge 6.2'de verilmiştir. Burada sigma hariç diğer parametre değiştirilmemiştir. Sigma ve erfHata parametreleri için test değerleri Çizelge 6.3'de verilmiştir. Burada erfHata Eşitlik 2.27 ile verilen denklemdeki Hata'yı temsil etmektedir. 'sift.simd.hpp' dosyası SIFT algoritmasının bir çok diğer parametrelerini de içermektedir. Bu parametrelerin her biri olduğu gibi bırakılmıştır.

Çizelge 6.2. OpenCV'nin SIFT algoritmasının bazı önemli parametreleri.

Parametreler	Değerleri
octaveLayers	3
contrastThreshold	0.04
edgeThreshold	10
sigma	1.6

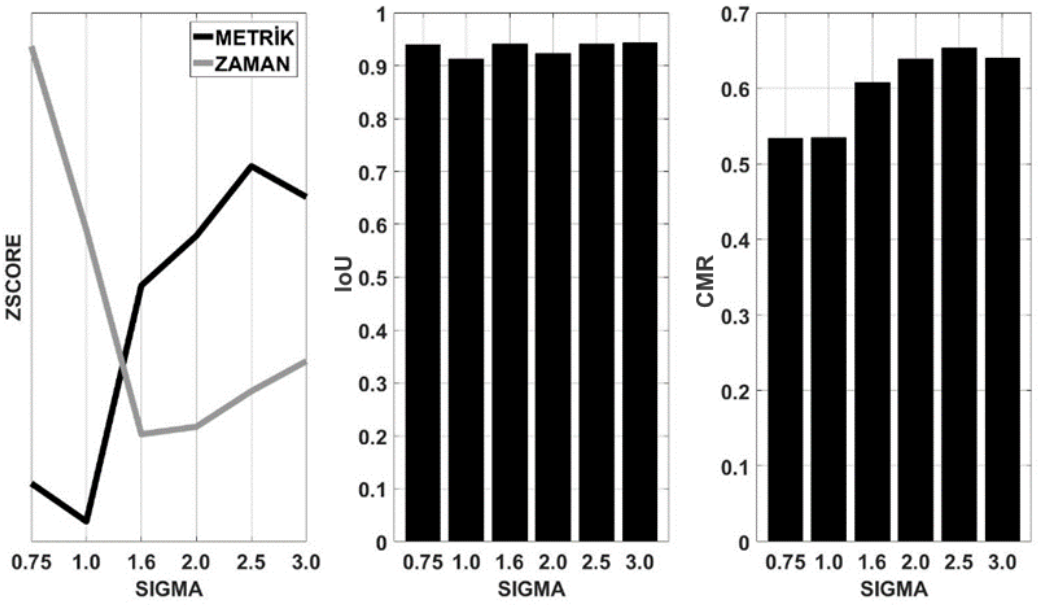
Çizelge 6.3. SIFT algoritmasının iki parmetresi için test değerleri.

Parametreler	Test Değerleri
sigma	$0.75; 1.0; 1.6; 2.0; 2.5; 3.0$
erfHata	$10^{-2}; 10^{-3}; 10^{-4}; 10^{-5}; 10^{-6}$

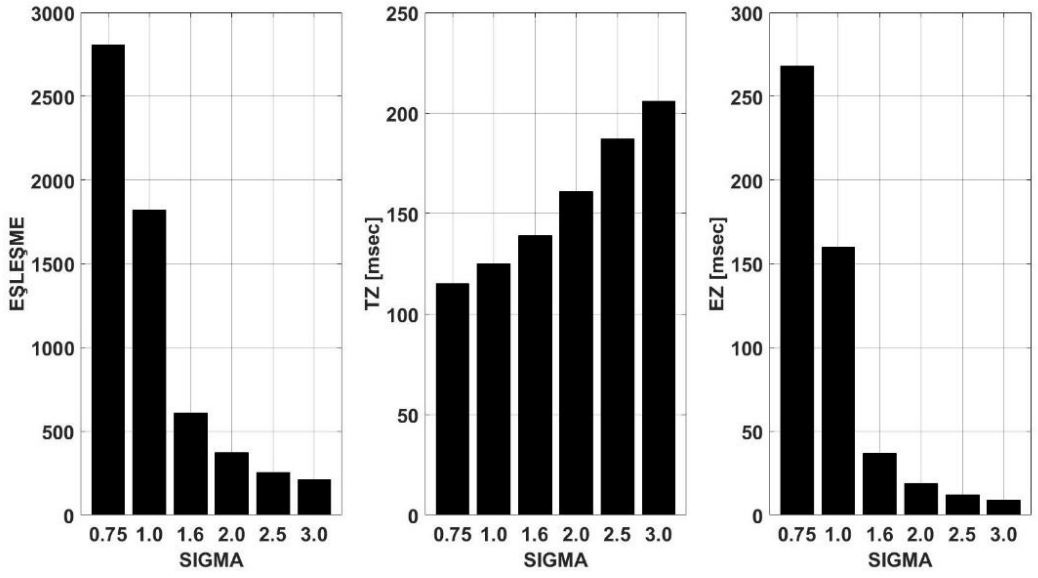
6.2.1. Giriş Görüntüsünde Varsayılan Bulanıklaşma Seviyesi (Mod-1)

Şekil 6.7-6.14, bu bölüm için Oxford ve HPatches veri kümeleri üzerindeki test sonuçlarını göstermektedir. Dikkat edilirse, sigma değerleri ile birlikte EŞLEŞME'nin de azaldığı görülmektedir. Dolayısı ile EZ değeri de azalmaktadır. Ancak, tersine TZ değeri sigma değerleri ile birlikte artmaktadır. Burada şunu belirtmeliyiz ki, Mod-1, konvolüsyonu için OpenCV'nin GaussianBlur fonksiyonunu kullanmaktadır. Test sonuçları, GaussianBlur fonksiyonunda, Gauss konvolüsyonunun örneklem aralığını belirlemek için bizim yukarıda belirttiğimiz Erf'yi temel alan benzer bir metot kullandığını gösteriyor. Dikkat edilirse, bu metotta örneklem boyutu sigma ile birlikte doğru orantılı olarak artmaktadır (Eşitlik 2.27). Sonuç olarak da konvolüsyon hesaplama zamanı artmaktadır.

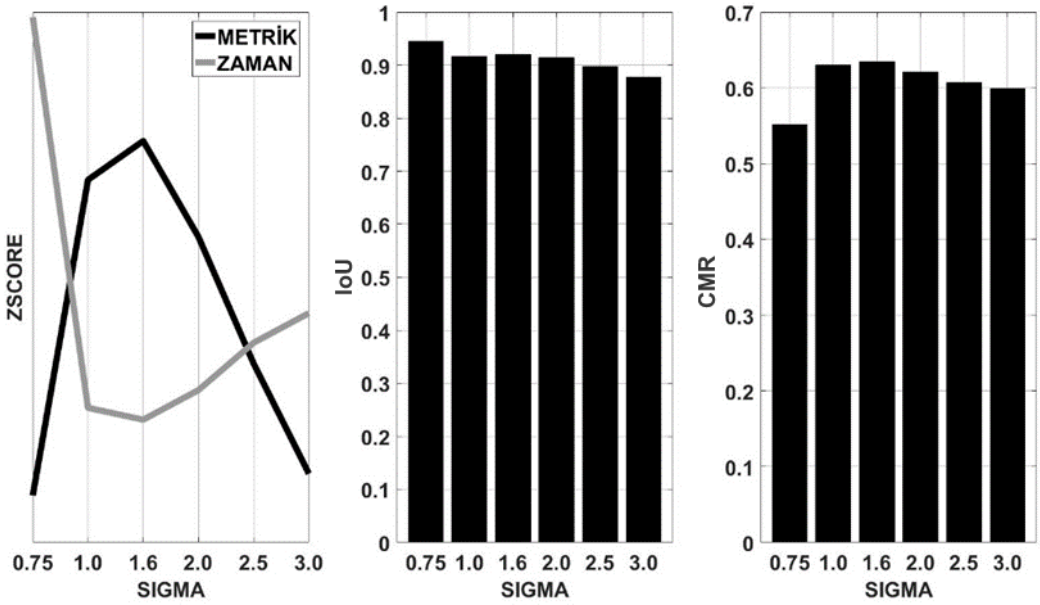
Oxford ve HPatches veri kümeleri üzerinde IoU ve video üzerinde V-DIŞLAR metriklerini incelediğimizde, farklı trendlere sahip olduklarını görüyoruz (Şekil 6.15-6.18). Giriş görüntüsü iki katına çıkartıldığında, IoU metriğinin SIGMA değerlerine göre değişimi hemen hemen sabittir. Ancak, giriş görüntüsü iki katına çıkartılmadığında, yavaş bir azalma eğilimi göstermektedir. METRİK ve ZAMAN verilerinin, SIGMA=1.6 noktasında ZSCORE'larını incelediğimizde, bu nokta en düşük ZAMAN değerine sahip olduğu için, optimum bir nokta olarak değerlendirilebilir. Ancak HPatches veri kümesi üzerinde, giriş görüntüsü iki katına çıkarılmadığında, SIGMA=1.0 noktası en yüksek METRİK ve en düşük ZAMAN değerlerine sahiptir. Ayrıca dikkat edilirse, veri kümeleri üzerinde, giriş görüntüsü iki katına çıkartıldığında ve çıkartılmadığında kayda değer bir gürbüzlük değişimi yoktur. Dolayısıyla, veri kümeleri üzerinde, giriş görüntüsünün iki katına çıkartmanın bir faydası yoktur.



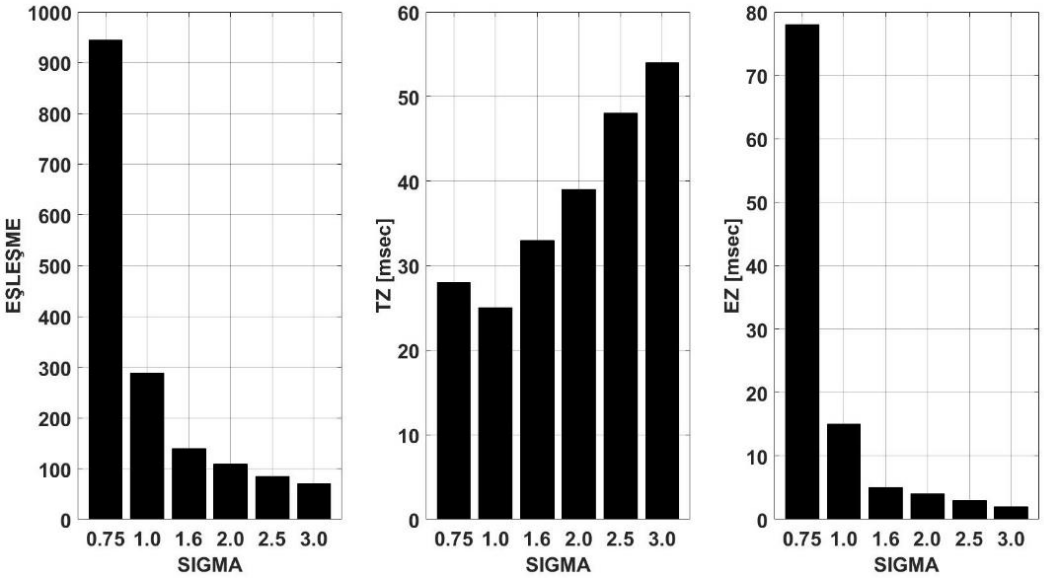
Şekil 6.7. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında Oxford veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



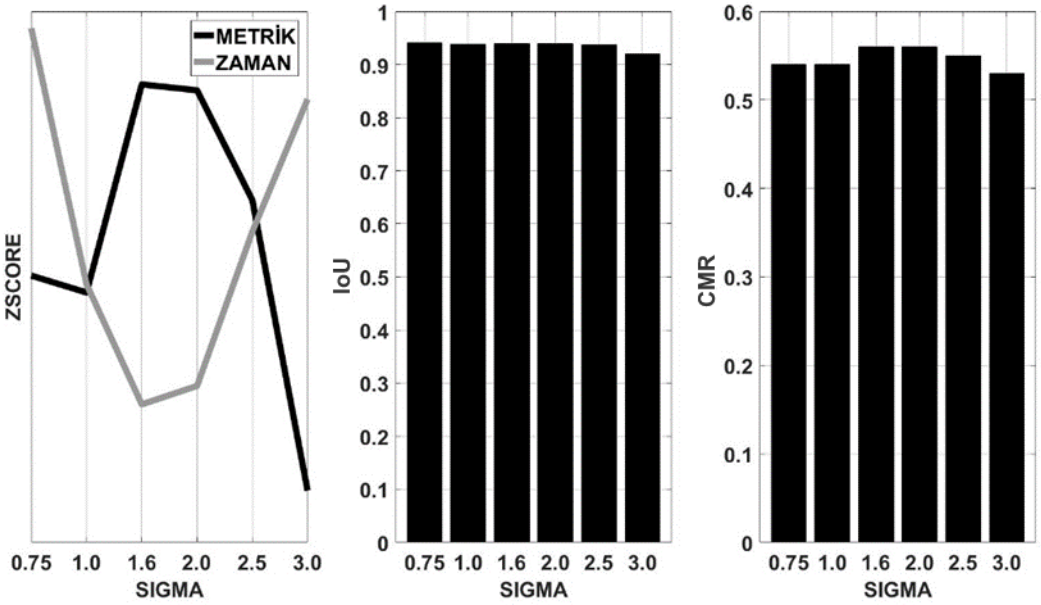
Şekil 6.8. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında Oxford veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



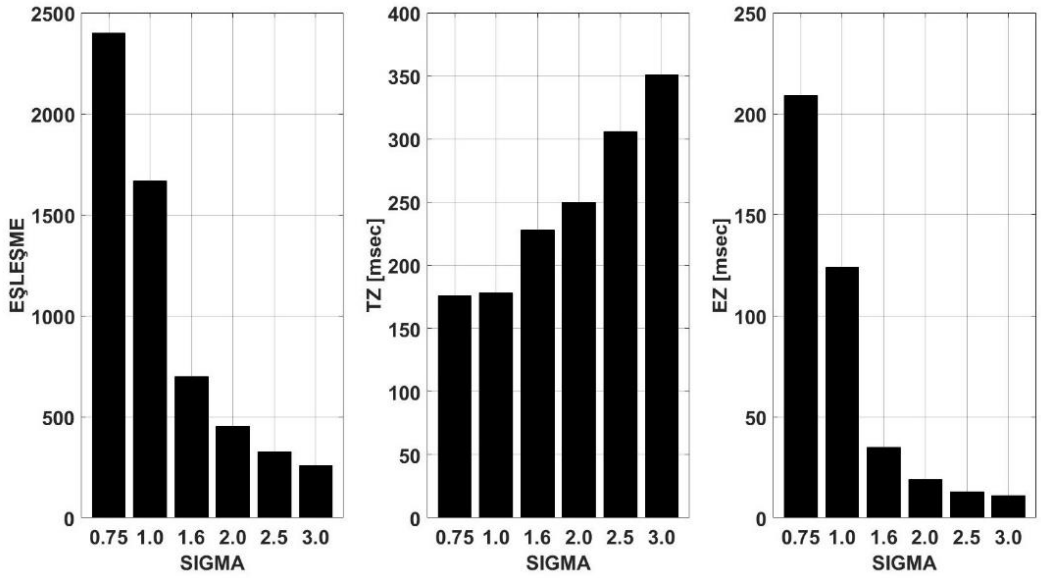
Şekil 6.9. Mod-1'in, giriş görüntüsü iki katına çıkarılmadığında Oxford veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



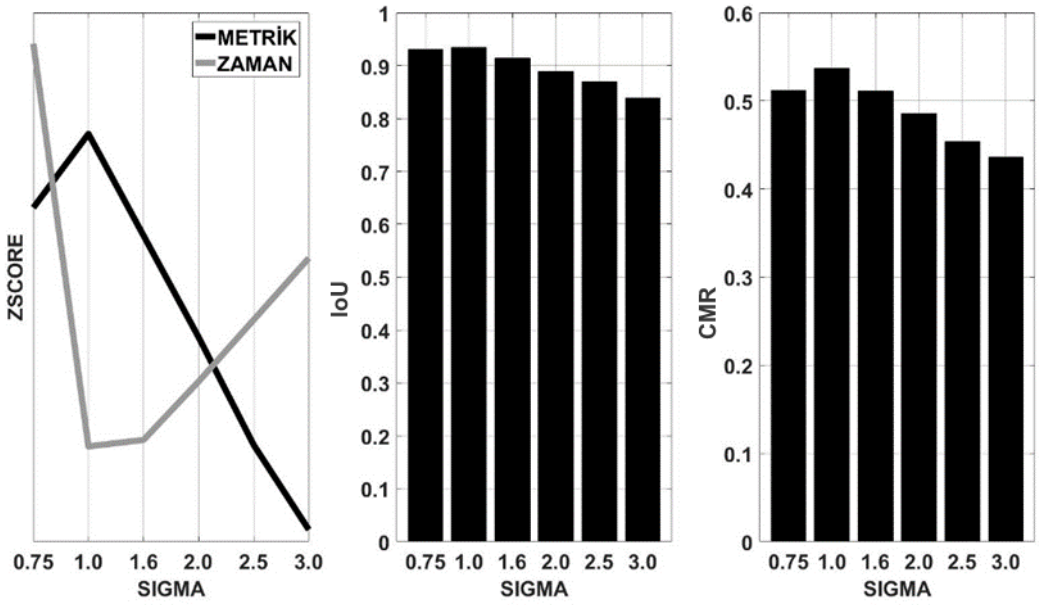
Şekil 6.10. Mod-1'in, giriş görüntüsü iki katına çıkarılmadığında Oxford veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



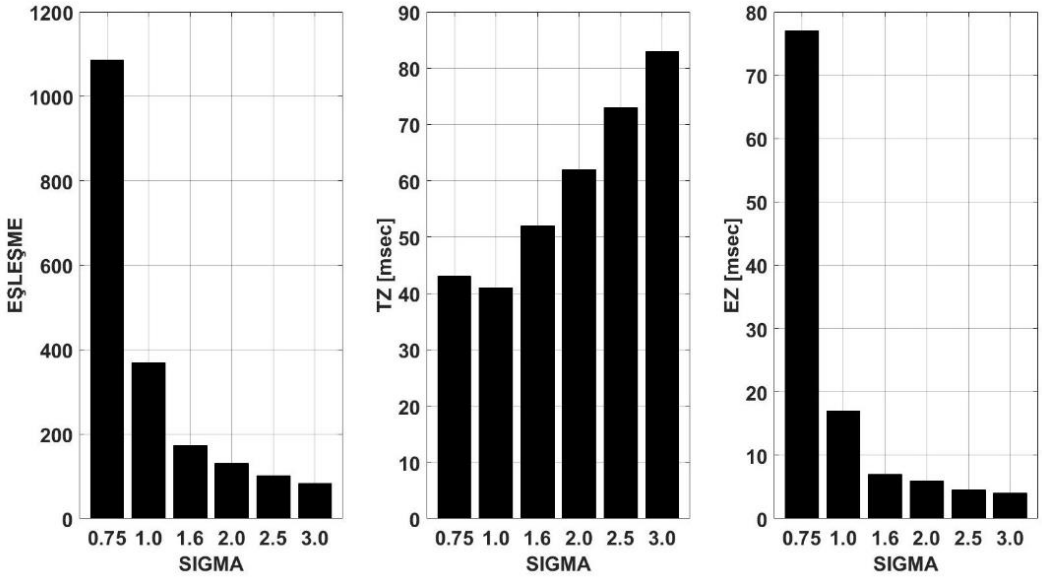
Şekil 6.11. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında HPatches veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



Şekil 6.12. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında HPatches veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



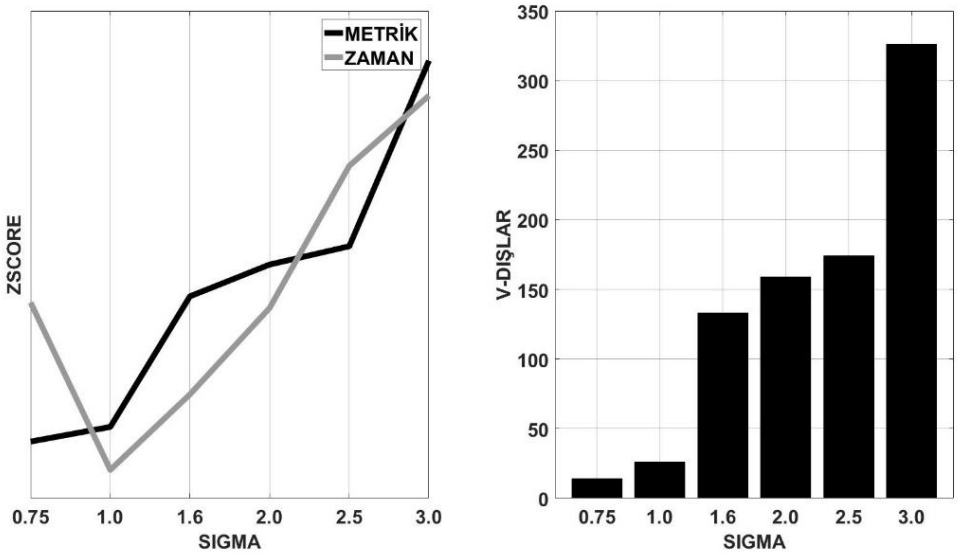
Şekil 6.13. Mod-1'in, giriş görüntüsü iki katına çıkartılmadığında HPatches veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



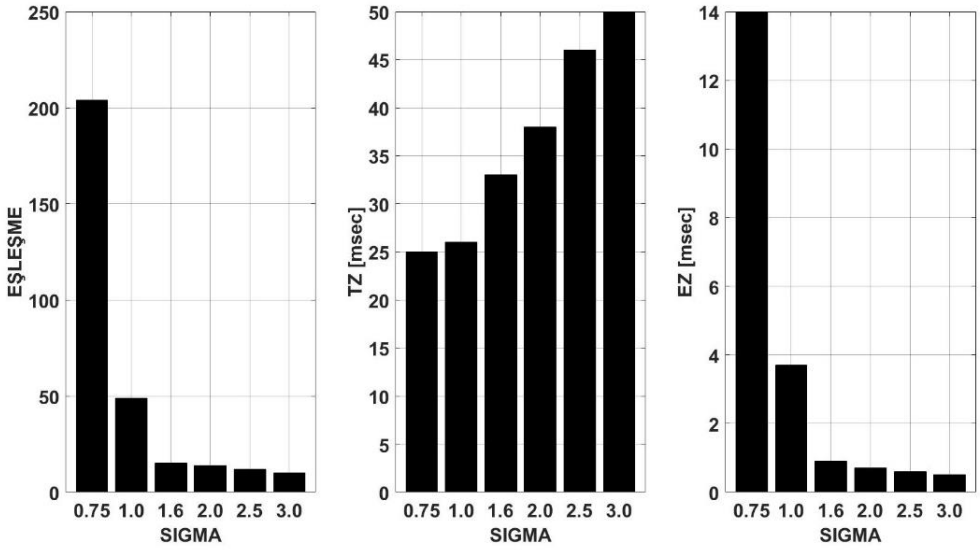
Şekil 6.14. Mod-1'in, giriş görüntüsü iki katına çıkartılmadığında HPatches veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).

Diğer taraftan, video üzerinde V-DIŞLAR metriğini incelediğimizde (Şekil 6.15, 6.16), giriş görüntüsünün iki katına çıkarıldığı ve çıkarılmadığı her iki durumda da sigma değerleri ile birlikte artış eğiliminde olduğunu görürüz. V-DIŞLAR metriğindeki artış, gürbüzlüğün azaldığı anlamına gelmektedir. Üstelik giriş

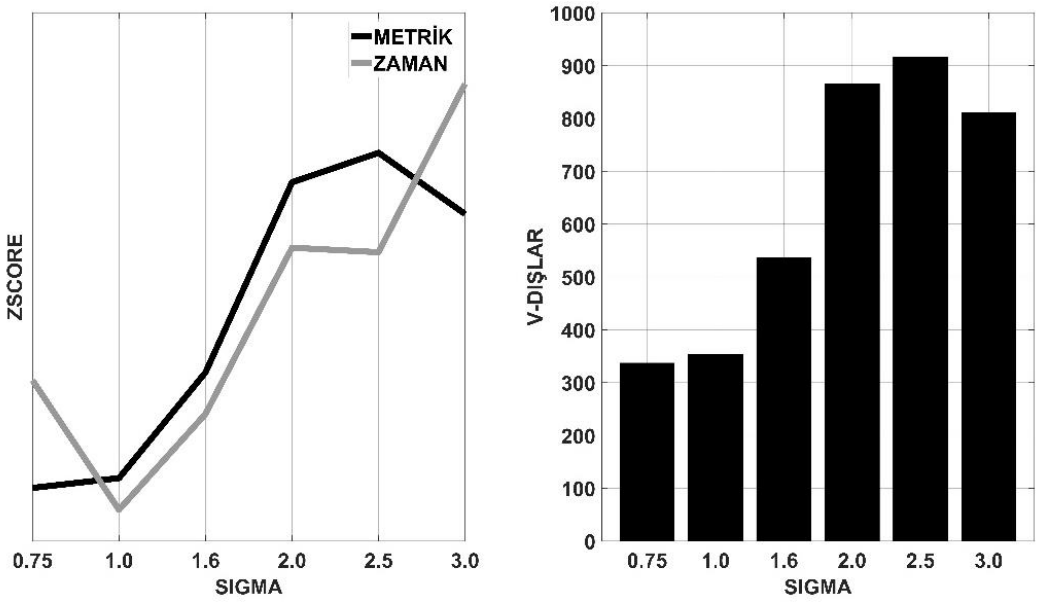
görüntüsünün iki katına çıkarıldığı ve çıkarılmadığı durumlar arasında önemli bir gürbüzlük farkının var olduğu anlaşılmaktadır. Örneğin, SIGMA=1.0 noktasına bakarsak, V-DIŞLAR metriğinin değerlerinin giriş görüntüsü iki katına çıkartıldığında 26, çıkartılmadığında is 354 olduğunu görürüz. SIGMA=1.0 noktası, V-DIŞLAR ve ZAMAN değerlerinin en küçük değerlerini aldığı nokta olduğu için, video üzerinde optimum nokta olarak değerlendirilebilir.



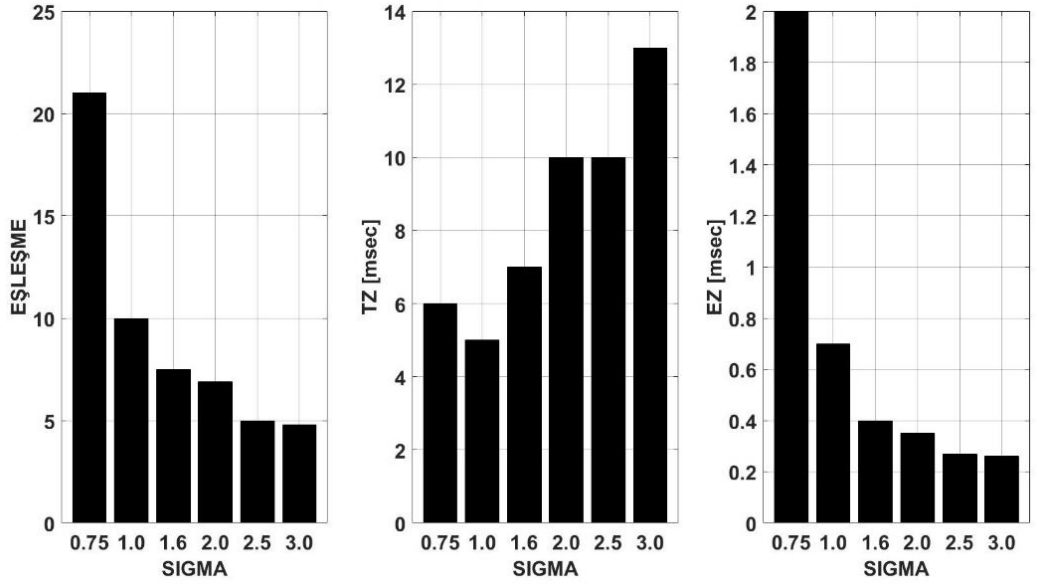
Şekil 6.15. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında Video üzerindeki test sonuçları (ZSCORE, V-DIŞLAR).



Şekil 6.16. Mod-1'in, giriş görüntüsü iki katına çıkartıldığında Video üzerindeki test sonuçları (EŞLEŞME, TZ, EZ).



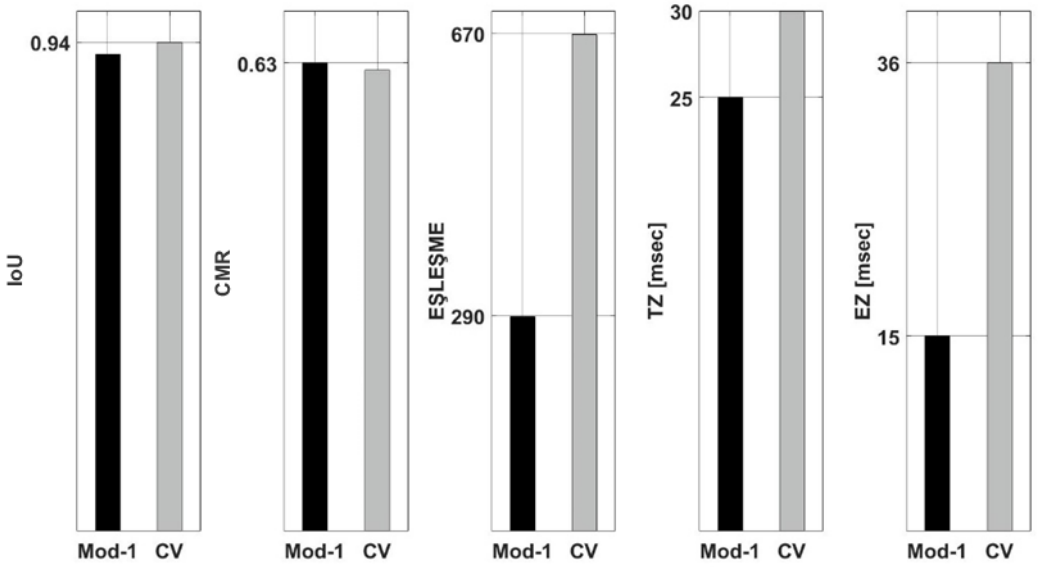
Şekil 6.17. Mod-1'in, giriş görüntüsü iki katına çıkartılmadığında Video üzerindeki test sonuçları (ZSCORE, V-DIŞLAR).



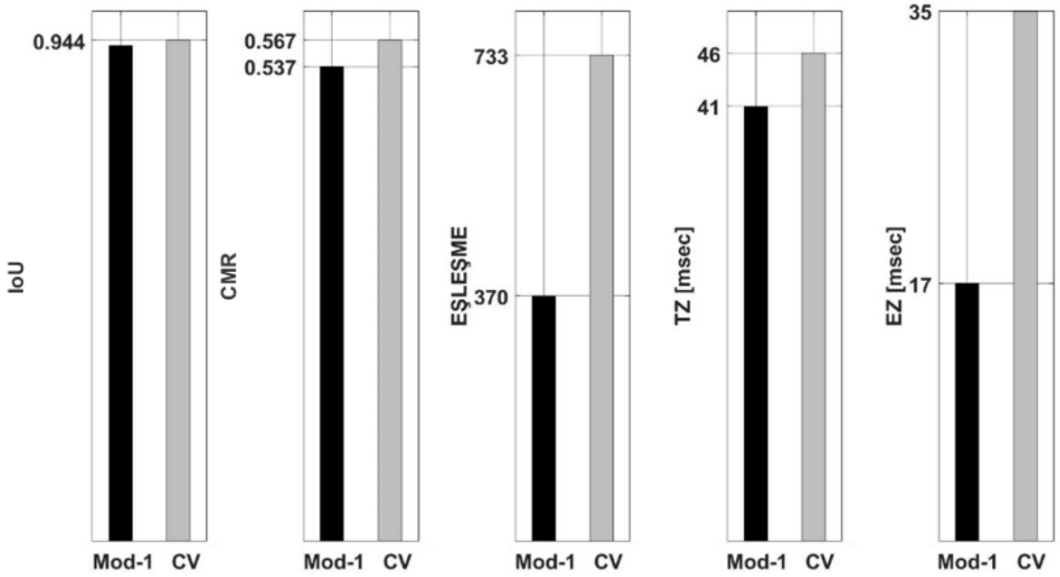
Şekil 6.18. Mod-1'in, giriş görüntüsü iki katına çıkartılmadığında Video üzerindeki test sonuçları (EŞLEŞME, TZ, EZ).

Şekil 6.19, 6.20, giriş görüntüsü iki katına çıkarılmadığında ve SIGMA=1.0 olduğunda, Mod-1 ve CV algoritmalarının veri kümeleri üzerinde birbirleriyle karşılaştırılmasını göstermektedir. Burada, önemli bir gürbüzlük farkı olmadan Mod-1'in CV'den daha hızlı olduğu gözükmektedir. Buradaki hız kazanımının sebebi Mod-

1’in daha az anahtar nokta eşleşimiyle benzer bir gürbüzlüğü yakalamış olmasından kaynaklanmaktadır.



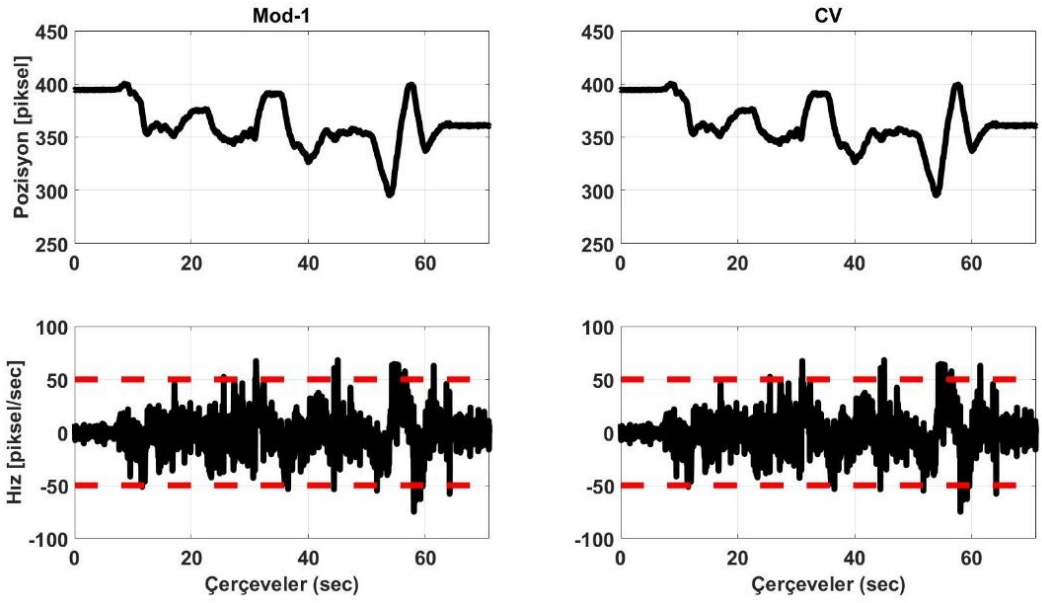
Şekil 6.19. Sigma=1.0 ve giriş görüntüsü iki katına çıkarılmadığında, Oxford veri kümesi üzerinde Mod-1 ve CV’nin birbirleri ile karşılaştırılması.



Şekil 6.20. Sigma=1.0 ve giriş görüntüsü iki katına çıkarılmadığında, HPatches veri kümesi üzerinde Mod-1 ve CV’nin birbirleri ile karşılaştırılması.

Diğer taraftan Şekil 6.21, Mod-1 ve CV ile ayrı ayrı elde edilen video üzerindeki hareket ve hız yörüngelerini göstermektedir. Her iki yörünge arasında bir fark yoktur. Dolayısı ile V-DIŞLAR metrikleri arasında da bir fark yoktur. Sonuç olarak, veri

kümeleri üzerindeki test değerleri göz önüne alınarak, giriş görüntüsü üzerinde başlangıç bulanıklık varsayımının faydası olmadığını söyleyebiliriz. Tersine, bu varsayım eşleşen anahtar noktaların sayısını gereksiz bir şekilde artırarak, algoritmanın hesaplama zamanını kötü yönde etkileyebilmektedir.

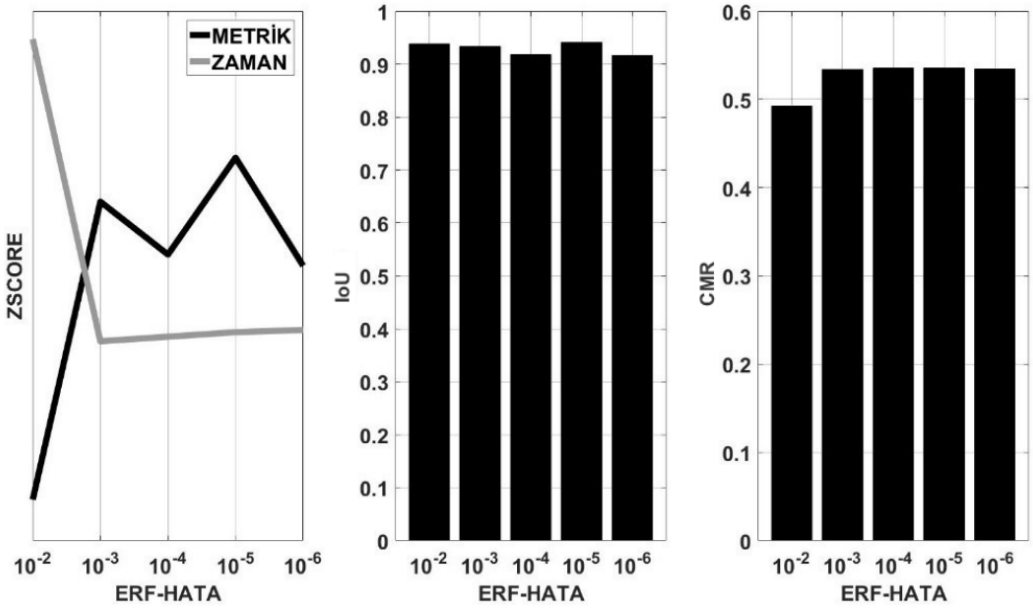


Şekil 6.21. Sigma=1.0 ve giriş görüntüsü iki katına çıkarıldığında, Video üzerinde Mod-1 ve CV'nin birbirleri ile karşılaştırılması.

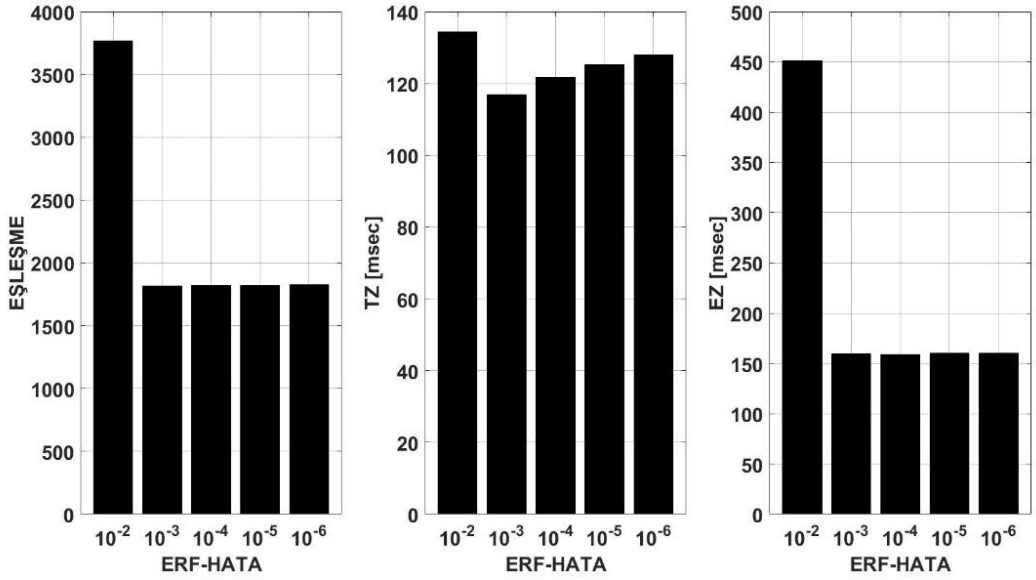
6.1.2. Erf Temelli Denklem ile Hesaplanan Örneklem Boyutunun Etkisi.

Şekil 6.22-6.33, bu bölüm için veri kümeleri ve video üzerindeki test sonuçlarını göstermektedir. ERF-HATA=10^{-2} olduğunda EŞLEŞME ve EZ değerlerinin çok yüksek olduğu göze çarpmaktadır. ERF-HATA=10^{-2} noktasında, gürbüzlük metrik (IoU ve CMR) değerlerinde önemli bir artışın olmadığı gözükmektedir. Üstelik bu noktada veri kümelerinin CMR değerlerinde önemli bir düşüş gözlemlenmektedir. ERF-HATA=10^{-2} değerinden sonra EŞLEŞME ve EZ değerleri bir kararlılığa kavuşmakta ve TZ değeri ise yükselen bir eğilim göstermektedir. Bu grafiklerden ERF-HATA=10^{-3} noktasının optimal olduğu sonucuna ulaşılabilir. ERF-HATA=10^{-2} olduğunda konvolüsyon algoritmasının doğruluğu düşük olduğundan, birçok gereksiz anahtar-nokta ortaya çıkmaktadır.

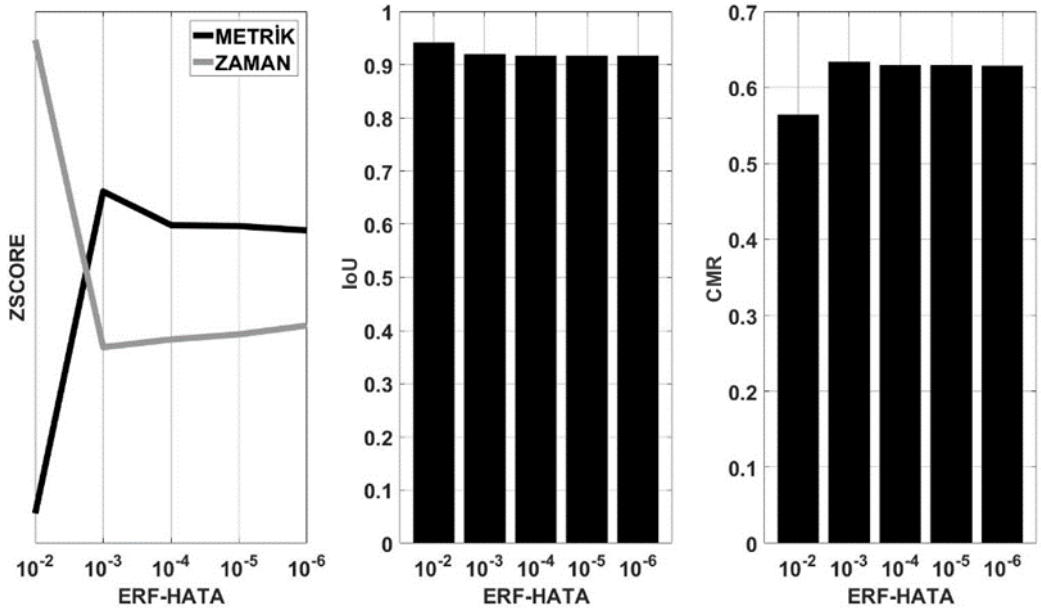
ERF-HATA=10^{-3} değeri ile elde edilen sonuçlar CV'nin değerleri ile hemen hemen aynıdır. Dolayısıyla buradan anlaşılmaktadır ki, OpenCV'nin GaussianBlur fonksiyonunda da benzer bir algoritma kullanılmıştır.



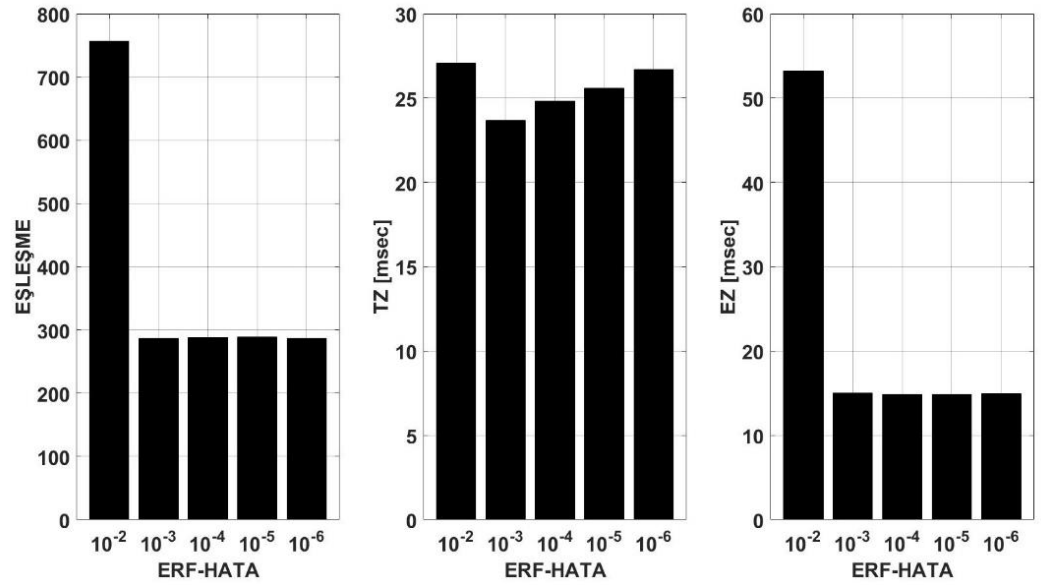
Şekil 6.22. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında Oxford veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



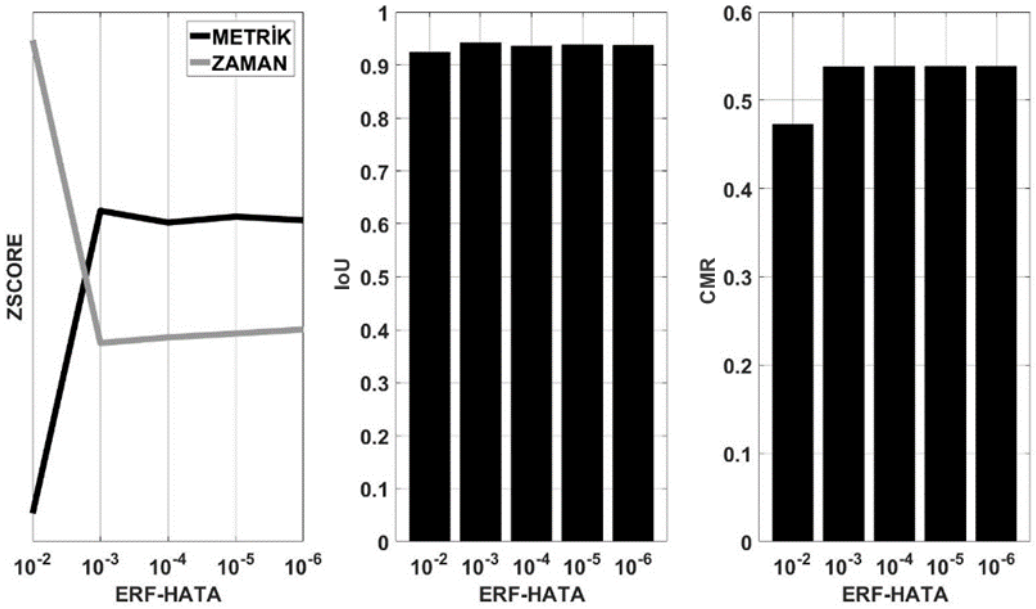
Şekil 6.23. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında Oxford veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



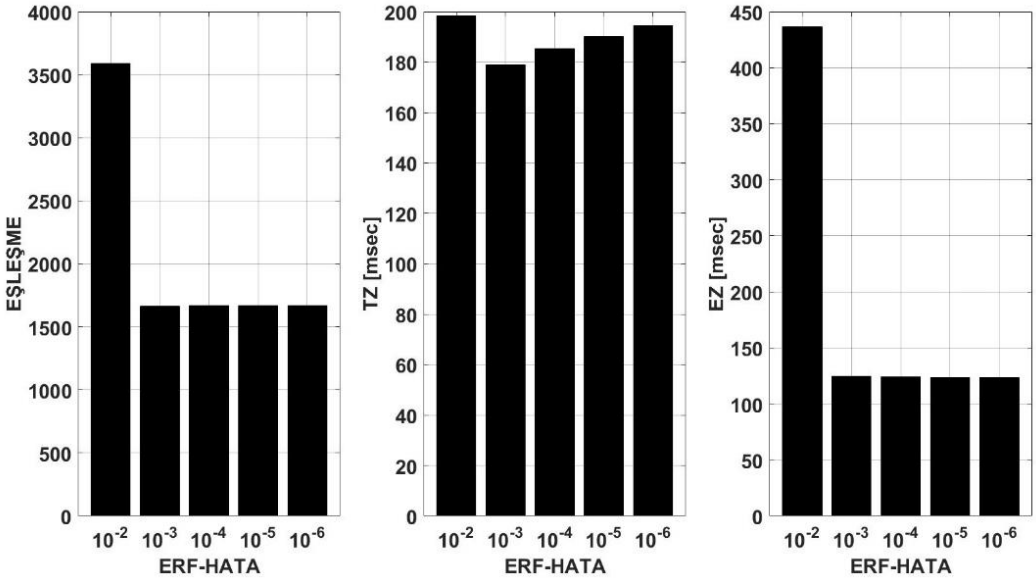
Şekil 6.24. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında Oxford veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



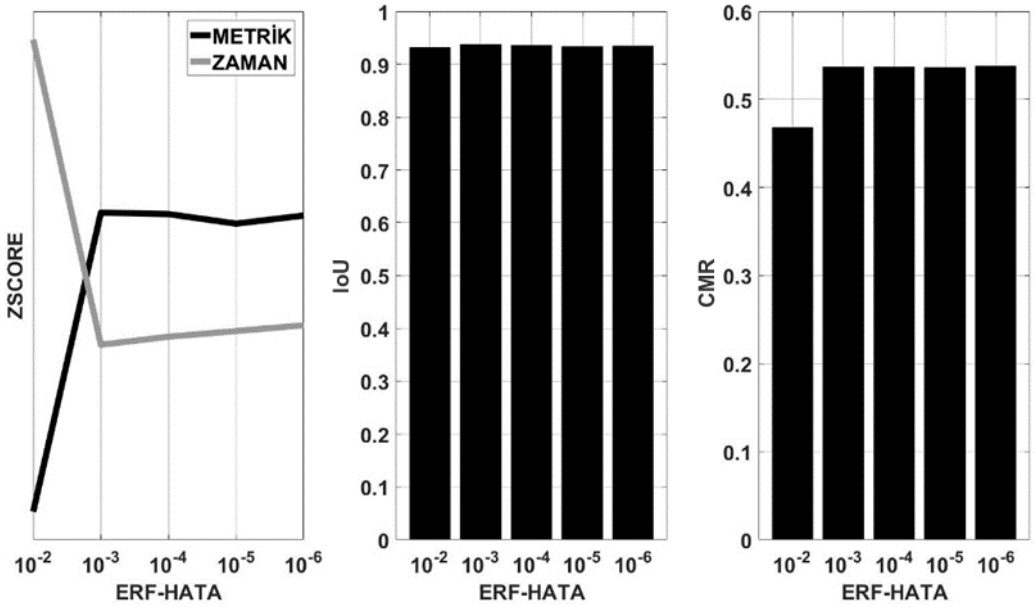
Şekil 6.25. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında Oxford veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



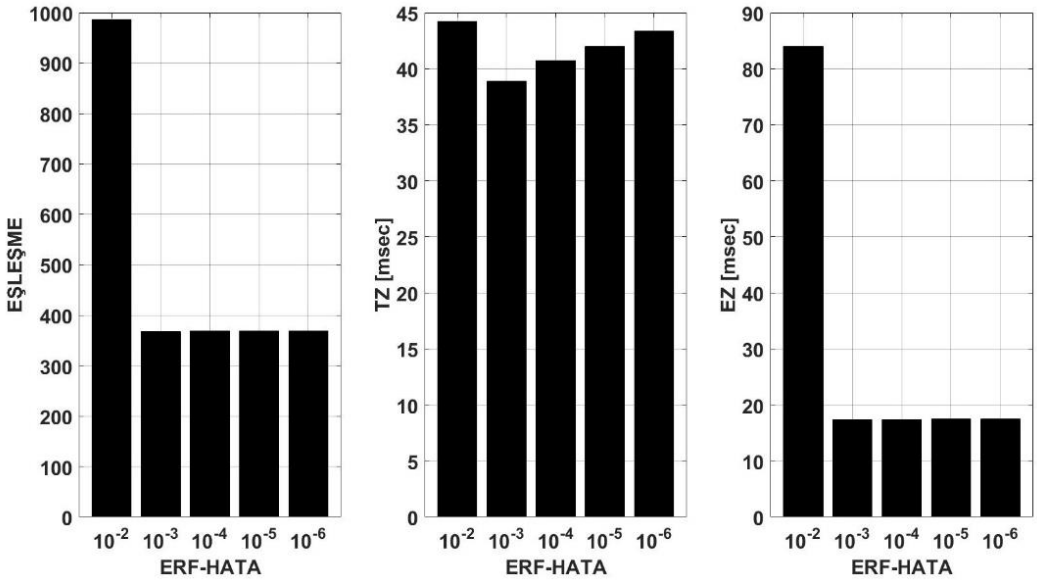
Şekil 6.26. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında HPatches veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



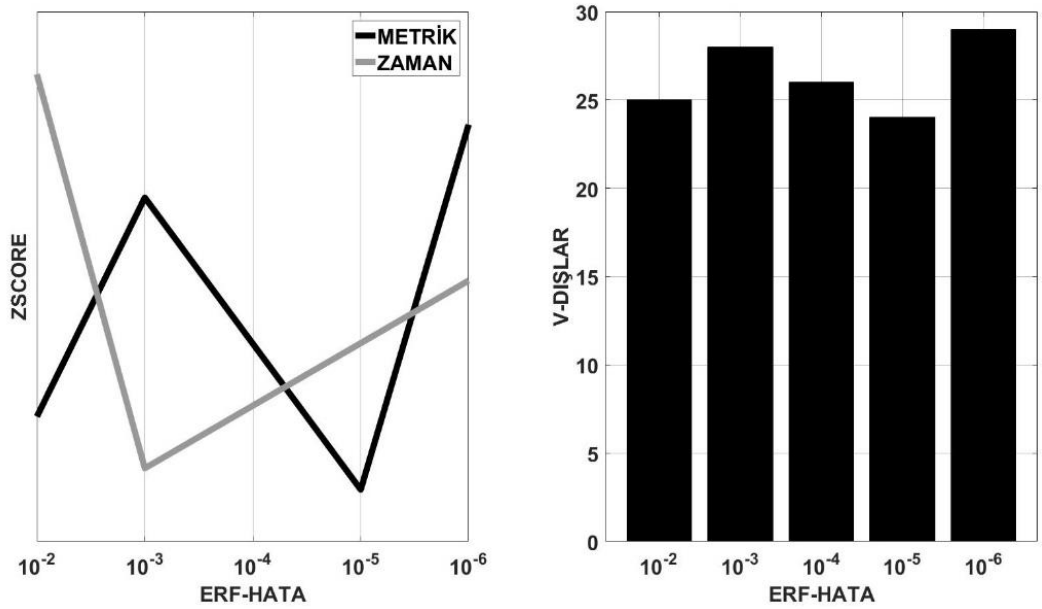
Şekil 6.27. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında HPatches veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



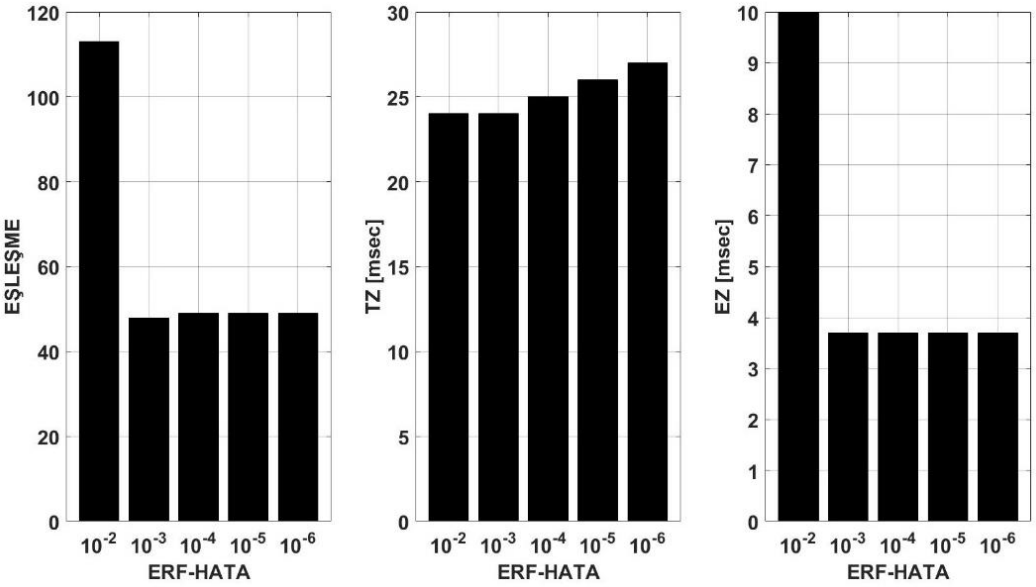
Şekil 6.28. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında HPatches veri kümesi üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



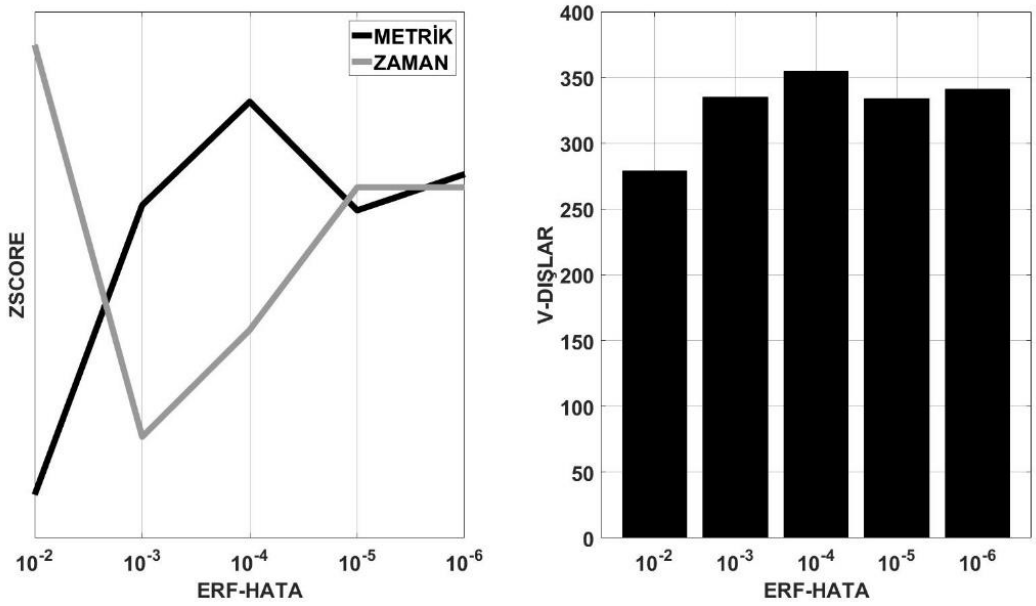
Şekil 6.29. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında HPatches veri kümesi üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



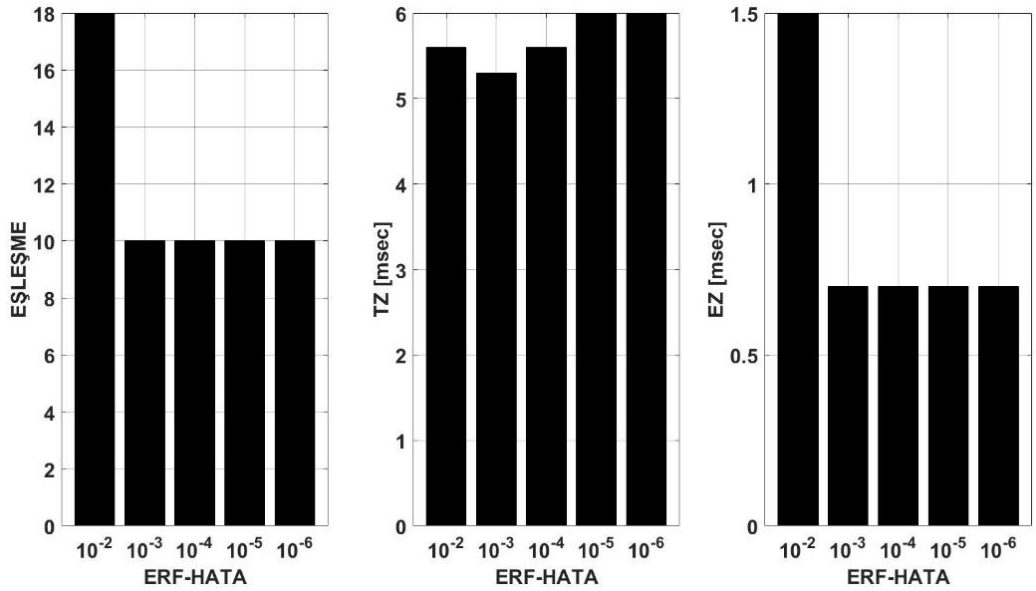
Şekil 6.30. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında video üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



Şekil 6.31. Mod-2'nin, giriş görüntüsü iki katına çıkartıldığında video üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).



Şekil 6.32. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında video üzerindeki test sonuçları (ZSCORE, IoU ve CMR).



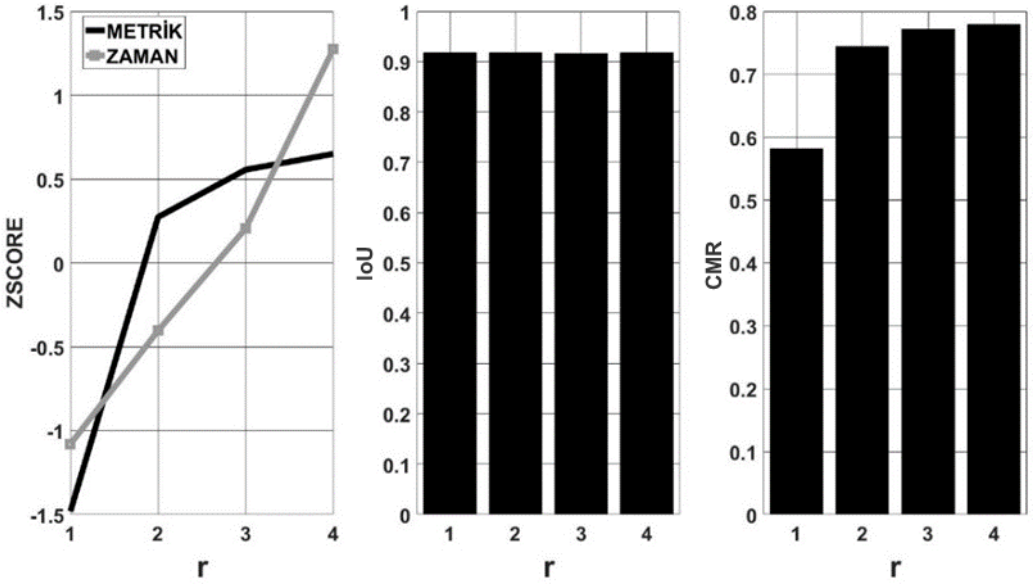
Şekil 6.33. Mod-2'nin, giriş görüntüsü iki katına çıkartılmadığında video üzerindeki test sonuçları (EŞLEŞME, TZ ve EZ).

6.2. ÖZNİCELİKLERİN HESAPLAMA ALGORİTMASI İÇİN DENEY SONUÇLARI

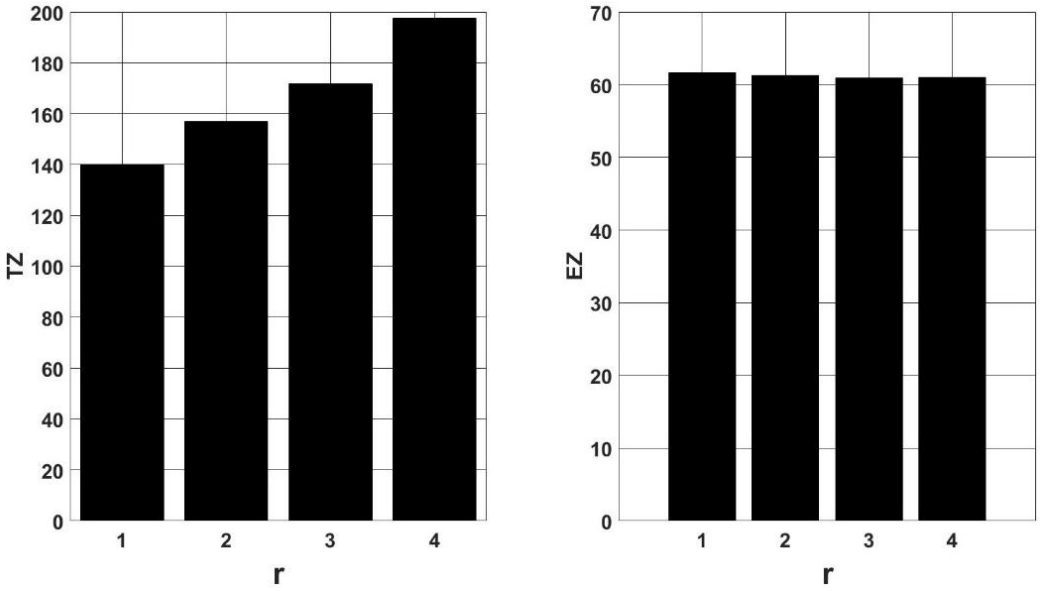
Bu bölümde ilk olarak yamanın (öznicelik vektörünün hesaplandığı karesel alanın) genişlik parametresi r değeri için optimum değer, Oxford ve HPatches veri kümeleri üzerinde test edilerek belirlenmiştir. Bu testte Hücre İçi Trilineer İnterpolasyon ile HOG Algoritma 3 kullanılmıştır. r için test değerleri 1, 2, 3 ve 4 'tür. r değeri belirlendikten sonra, öznicelik belirleme algoritmalarının, Algoritma 1, 2, 3 ve 4 (Bkz. Bölüm 3), performansı karşılaştırmalı olarak veri kümeleri üzerinde test edilmiştir.

6.2.1. Yama Genişlik Parametresi r Değerinin Belirlenmesi

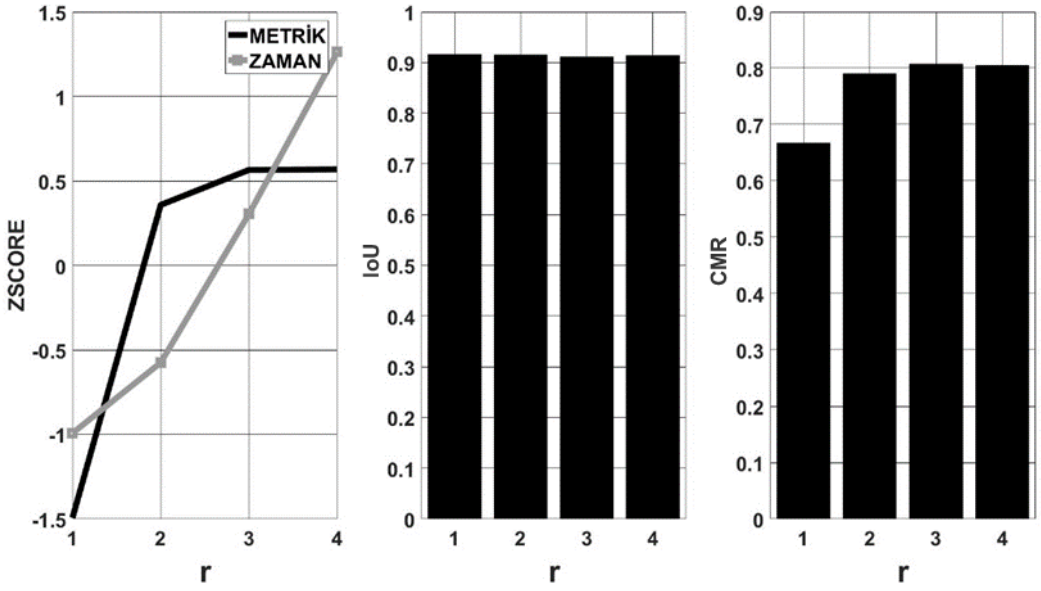
Şekil 6.34, 6.35, Oxford veri kümesi üzerinde, giriş görüntüsü iki katına çıkartıldığında, öznicelik belirleme algoritmasının r değerlerine göre performansını göstermektedir. METRİK'in ZSCORE değerlerine baktığımızda r değerleri ile birlikte arttığını görüyoruz. Ancak bu artış yalnızca CMR değerindeki artıştan kaynaklanmaktadır. IoU metriği hemen hemen, r değerlerindeki değişime göre sabittir. Bu sonuç bize CMR metriğindeki artışın IoU metriğinde artışa neden olmayabileceğini gösteriyor. Eşleşen noktaların yüzdesinin artmasının tanımlamanın performansını artırmaması, tanımlama için yeterli eşleşmeden fazla eşleşmenin gerçekleştiğine işaret eder. Şekil 6.35'de r değeri ile birlikte TZ değerinin arttığını ancak EZ değerinin hemen hemen sabit kaldığını gösteriyor. r değeri arttıkça, yamada işlenmesi gereken piksel sayısı arttığı için bu beklenen bir durumdur. Bu grafiklere bakarak, METRİK değerinin en yüksek olup ZAMAN değerinin en düşük olduğu noktanın r=2 olduğunu söyleyebiliriz. Aynı durum giriş görüntüsü iki katına çıkarılmadığı zaman da geçerlidir (Şekil 6.36-6.37).



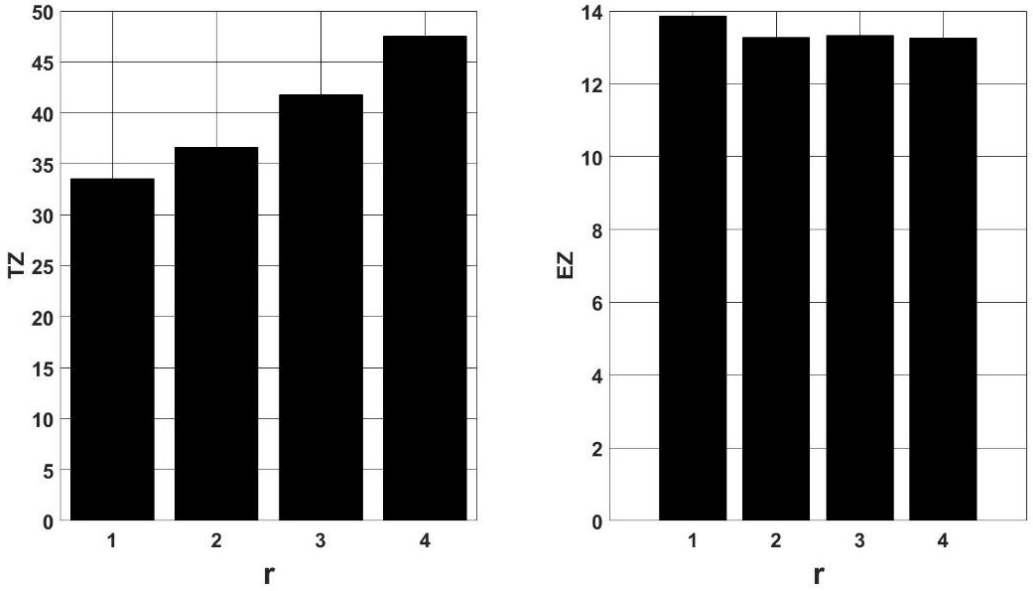
Şekil 6.34. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre Oxford veri kümesi üzeinde test sonuçları (ZSCORE, IoU ve CMR). Giriş görüntüsü iki katına çıkarılmıştır.



Şekil 6.35. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre Oxford veri kümesi üzeinde test sonuçları (TZ ve EZ). Giriş görüntüsü iki katına çıkarılmıştır.

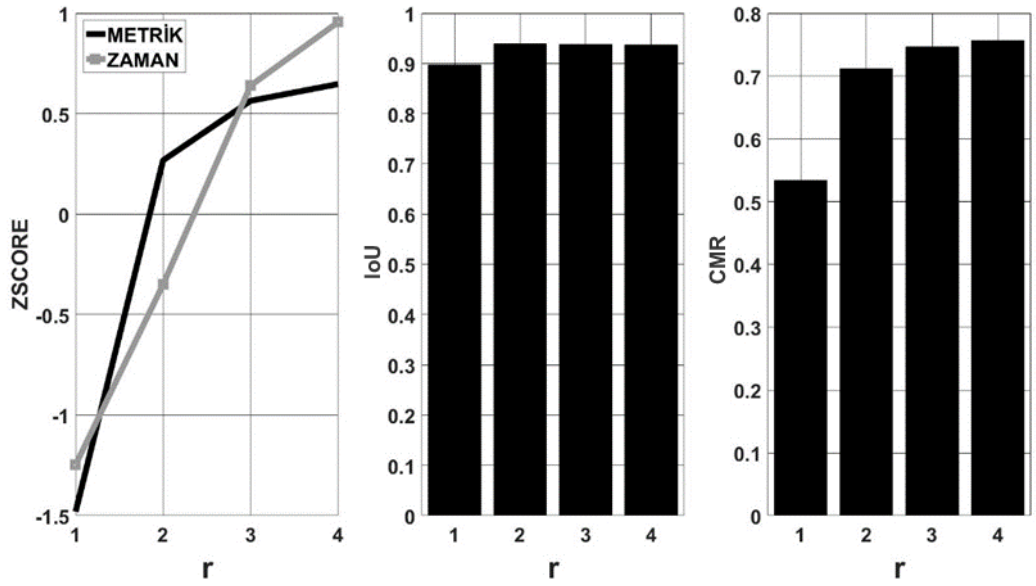


Şekil 6.36. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre Oxford veri kümesi üzeinde test sonuçları (ZSCORE, IoU ve CMR). Giriş görüntüsü iki katına çıkarılmamıştır.

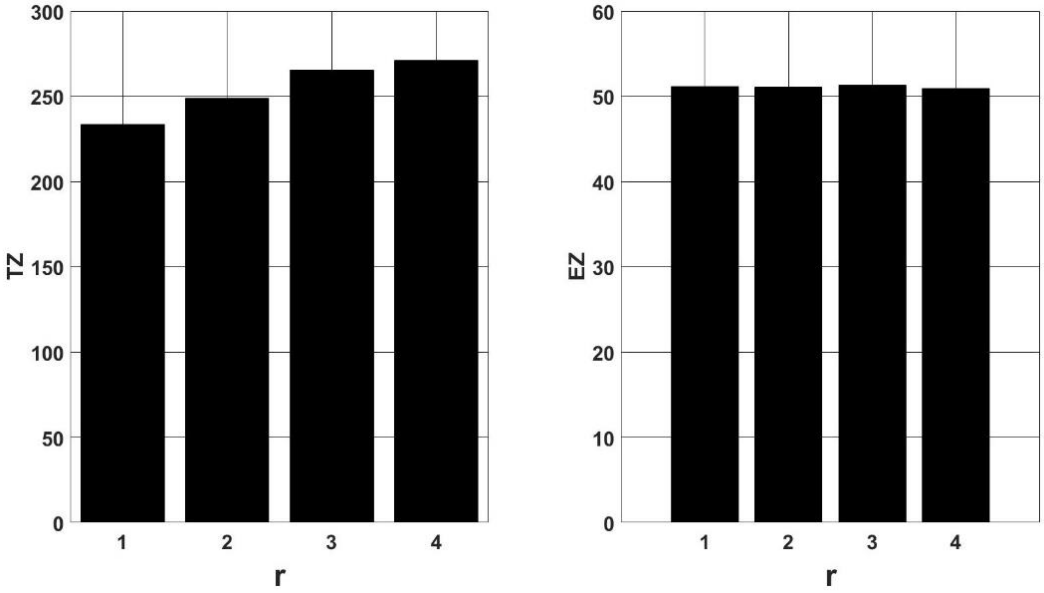


Şekil 6.37. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre Oxford veri kümesi üzeinde test sonuçları (TZ ve EZ). Giriş görüntüsü iki katına çıkarılmamıştır.

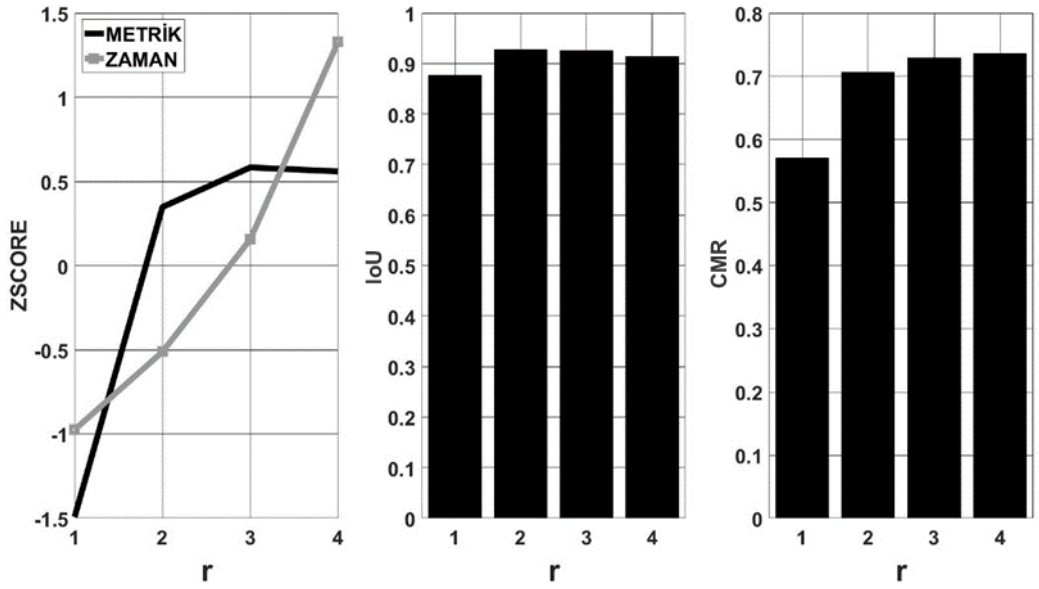
Şekiller 6.38–6.41, HPatches veri kümesi üzerinde r değerlerine göre performans değişimini göstermektedir. Bu durumda da sonuçlar Oxford veri kümesi üzerinde elde edilen sonuçlara benzerdir.



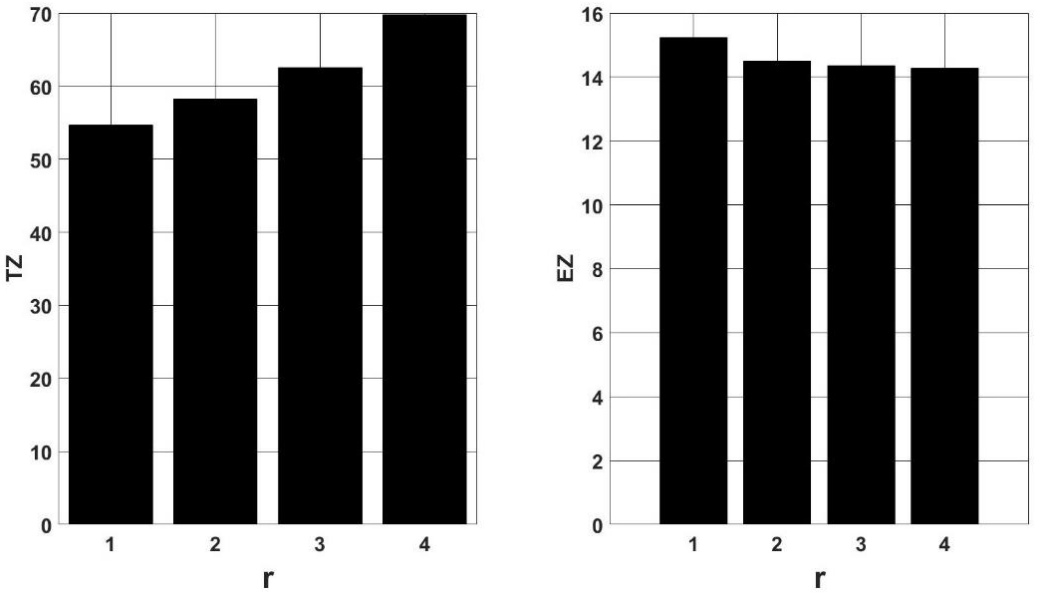
Şekil 6.38. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre HPatches veri kümesi üzeinde test sonuçları (ZSCORE, IoU ve CMR). Giriş görüntüsü iki katına çıkarılmıştır.



Şekil 6.39. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre HPatches veri kümesi üzeinde test sonuçları (TZ ve EZ). Giriş görüntüsü iki katına çıkarılmıştır.



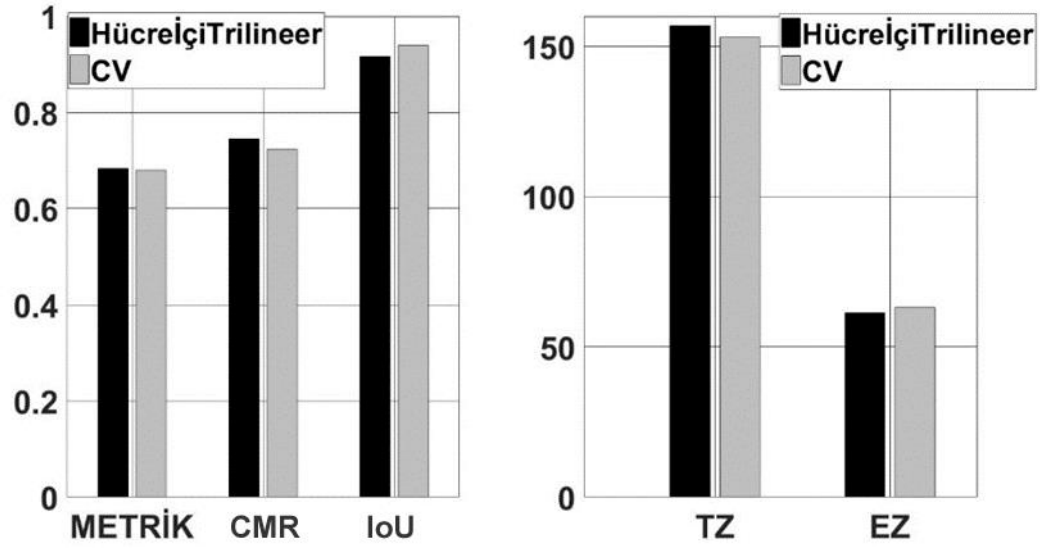
Şekil 6.40. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre HPatches veri kümesi üzeinde test sonuçları (ZSCORE, IoU ve CMR). Giriş görüntüsü iki katına çıkarılmamıştır.



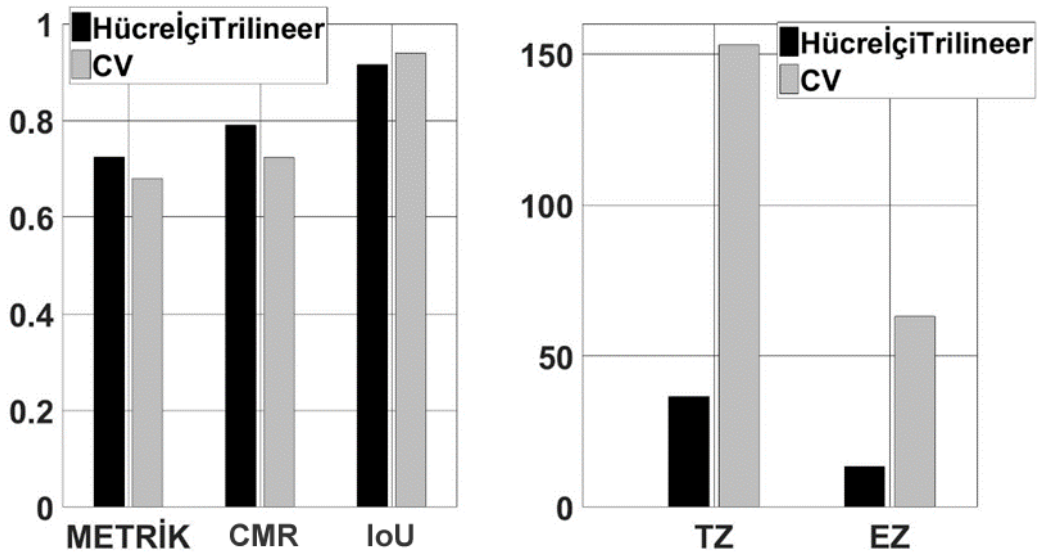
Şekil 6.41. Hücre İçi Lineer İnterpolasyon ile HOG algoritmasnın r parametrsine göre HPatches veri kümesi üzeinde test sonuçları (ZSCORE, IoU ve CMR). Giriş görüntüsü iki katına çıkarılmıştır.

Şekil 6.42, 6.43, CV algoritması ile r=2 ve sırası ile giriş görüntüsü iki katına çıkarıldığında ve çıkarılmadığı durumlar için Oxford veri kümesi üzerinde performans karşılaştırılması verilmiştir. Burada CV algoritması için her iki durumda da giriş

görüntüsü iki katına çıkarılmıştır. Bu sonuçlar r=2 seçildiğinde test edilen algoritmanın performansının, hem gürbüzlük hem de hesaplama zamanı açısından bir birine yakın olduğunu göstermektedir. Ayrıca Şekil 6.43, giriş görüntüsü iki katına çıkarılmadığında performansta önemli bir kayıp olmadan hesaplama zamanında ciddi bir kazanım olduğunu göstermektedir.

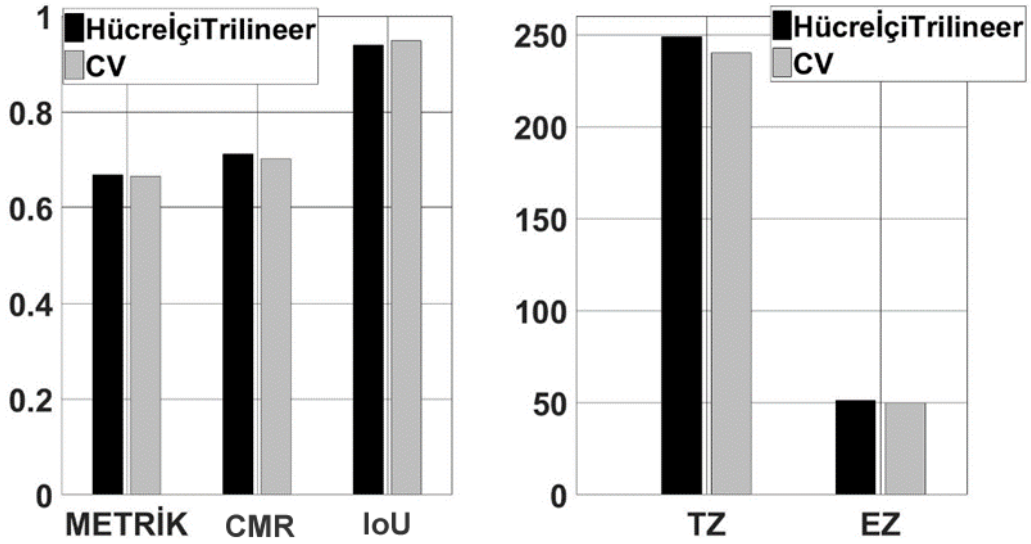


Şekil 6.42. CV ve Hücre İçi Trilineer İnterpolasyon İle HOG algoritmalarının Oxford veri kümesi üzerinde karşılaştırılması. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir.

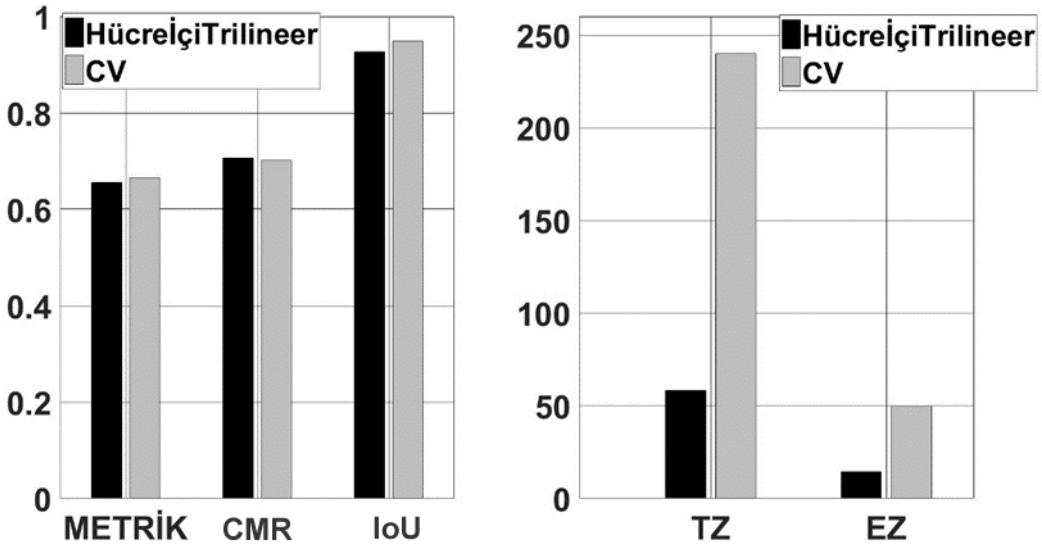


Şekil 6.43. CV ve Hücre İçi Trilineer İnterpolasyon İle HOG algoritmalarının Oxford veri kümesi üzerinde karşılaştırılması. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir.

Şekil 6.44, 6.45, CV algoritması ile r=2 ve sırası ile giriş görüntüsü iki katına çıkarıldığında ve çıkarılmadığı durumlar için HPatches veri kümesi üzerinde performans karşılaştırılması verilmiştir. Sonuçlar Oxford veri kümesinde olduğu gibi benzerdir.



Şekil 6.44. CV ve Hücre İçi Trilineer İnterpolasyon İle HOG algoritmalarının HPatches veri kümesi üzerinde karşılaştırılması. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir.



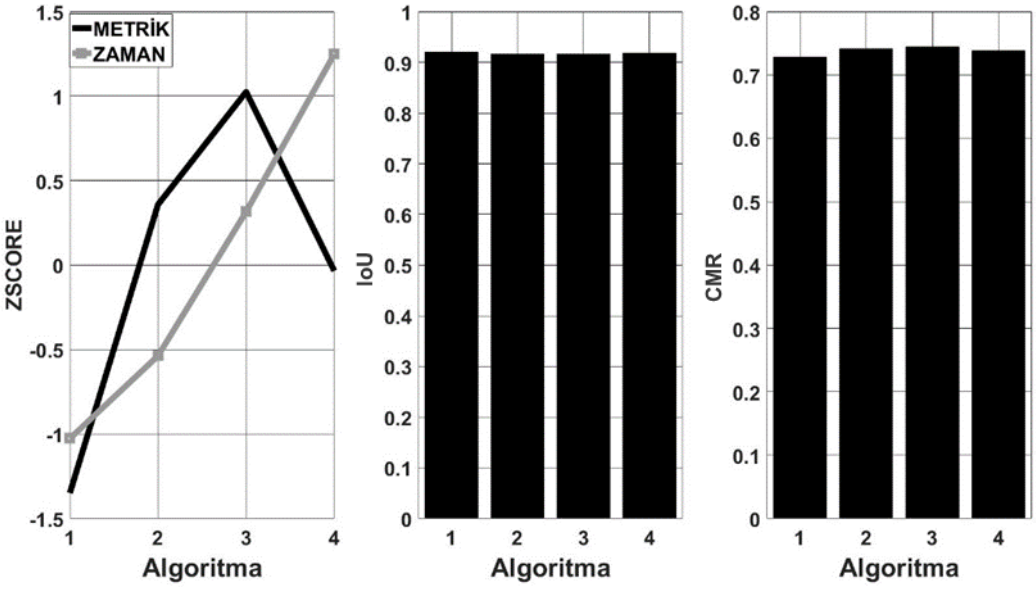
Şekil 6.45. CV ve Hücre İçi Trilineer İnterpolasyon İle HOG algoritmalarının HPatches veri kümesi üzerinde karşılaştırılması. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir.

6.2.3. İnterpolasyon Algoritmalarının Karşılaştırılması

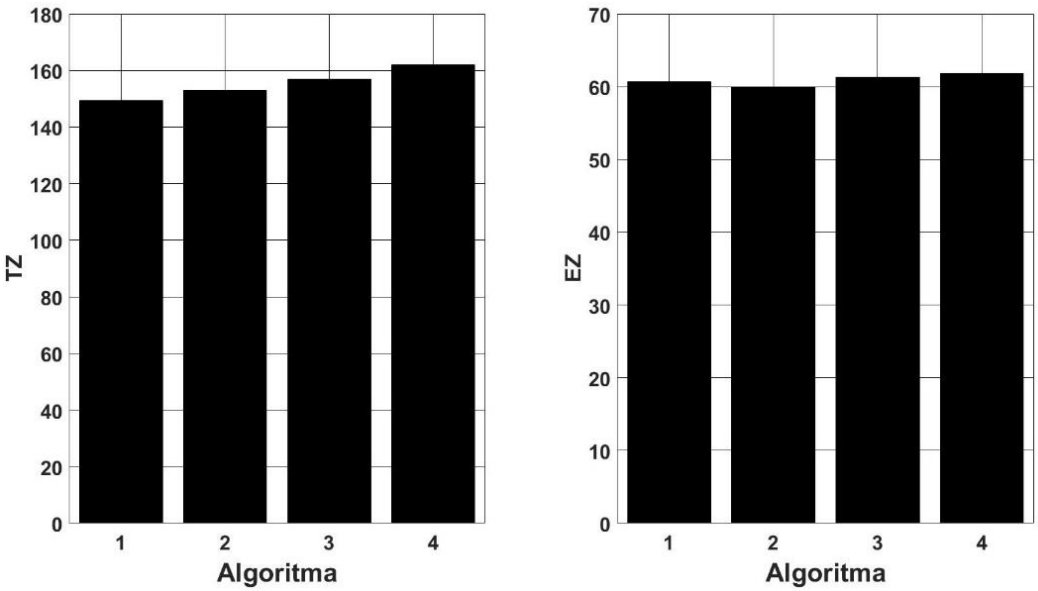
r=2 olduğunda bu bölüm, öznicelik hesaplama algoritmalarını bir birleriyle Oxford ve HPatches veri kümeleri üzerinde karşılaştırmaktadır. Testler, bir önceki bölümde olduğu gibi, giriş görüntüsü iki katına çıkarıldığı ve çıkarılmadığı her iki durum için ayrı ayrı gerçekleştirilmiştir.

Sonuçları incelediğimizde (Şekil 6.46–6.53), algoritmalar arasında önemli bir gürbüzlük farkının olmadığını görüyoruz. Burada ZSCORE metriğinin küçük farkları dahi yansıttığını belirtmeliyiz. Ancak IoU ve CMR metriklerine baktığımız zaman önemli bir farkın olmadığını görüyoruz. Hesaplama zamanına bakacak olursak, Algoritma 1'den Algoritma 4'e doğru algoritma kompleksliği arttığı için yavaş bir artış trendi olduğu söylenebilir.

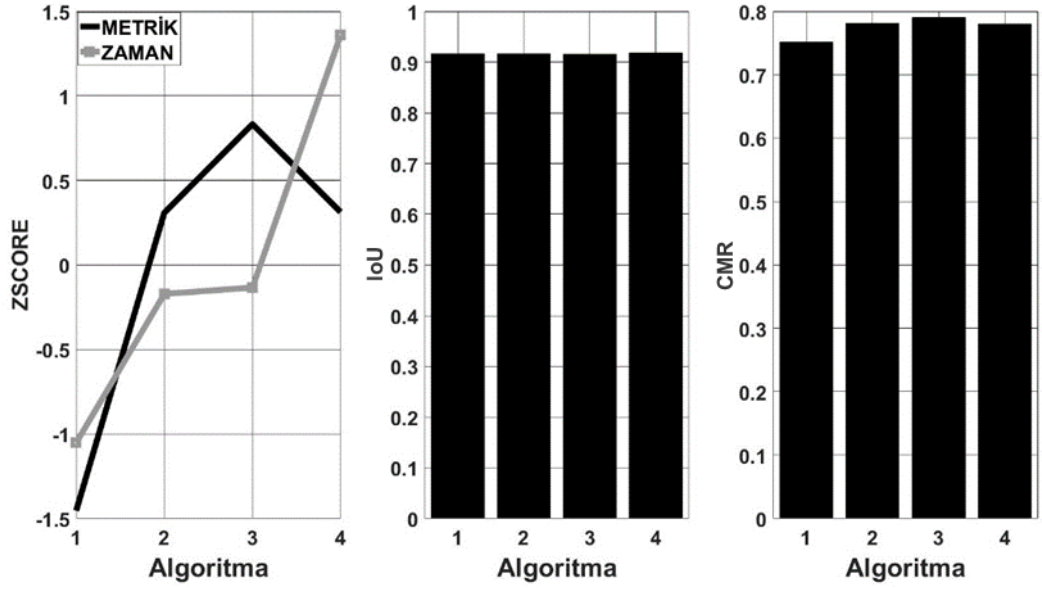
Sonuç olarak interpolasyonun bir etkisinin olduğunu ancak bu etkinin çok da önemli olmadığını söyleyebiliriz



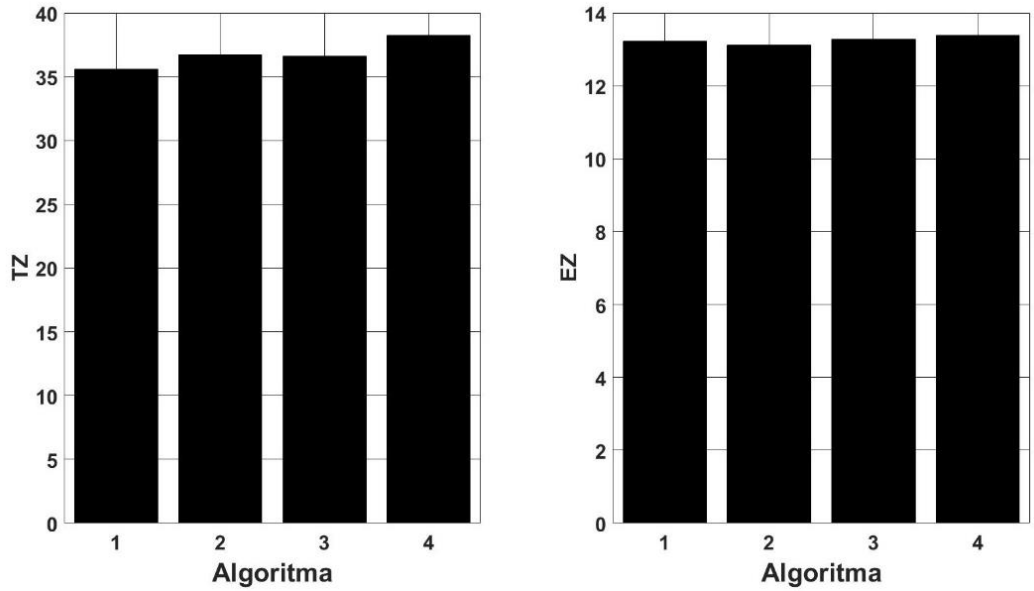
Şekil 6.46. İnterpolasyon algoritmalarının Oxford veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir (ZSCORE, IoU ve CMR).



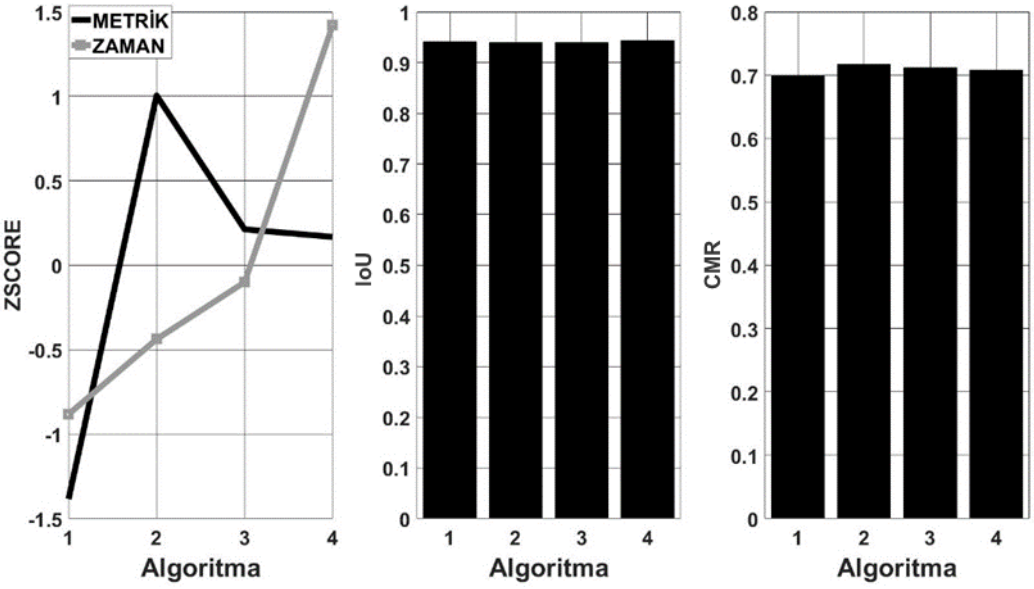
Şekil 6.47. İnterpolasyon algoritmalarının Oxford veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir (TZ ve EZ).



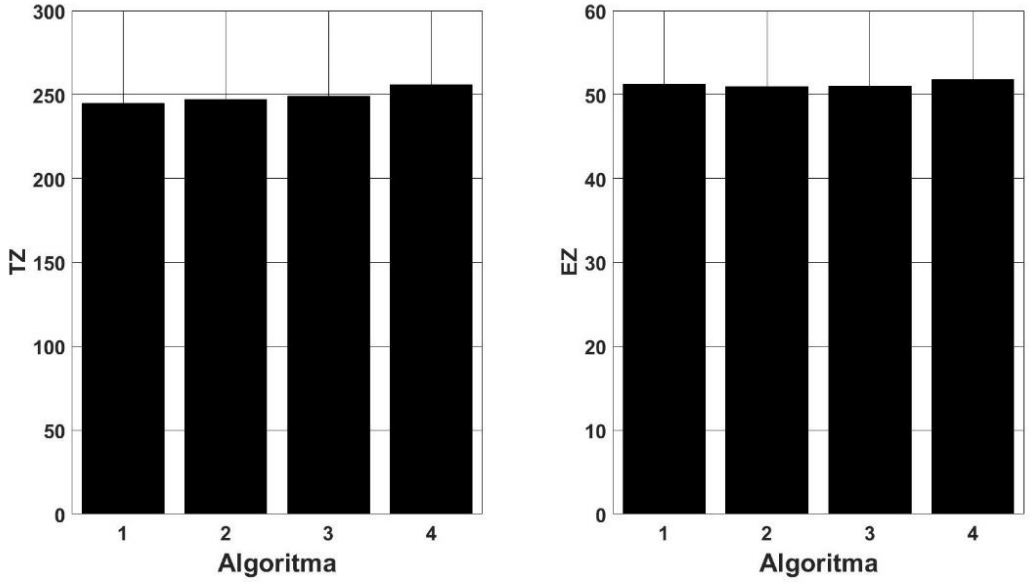
Şekil 6.48. İnterpolasyon algoritmalarının Oxford veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir (ZSCORE, IoU ve CMR).



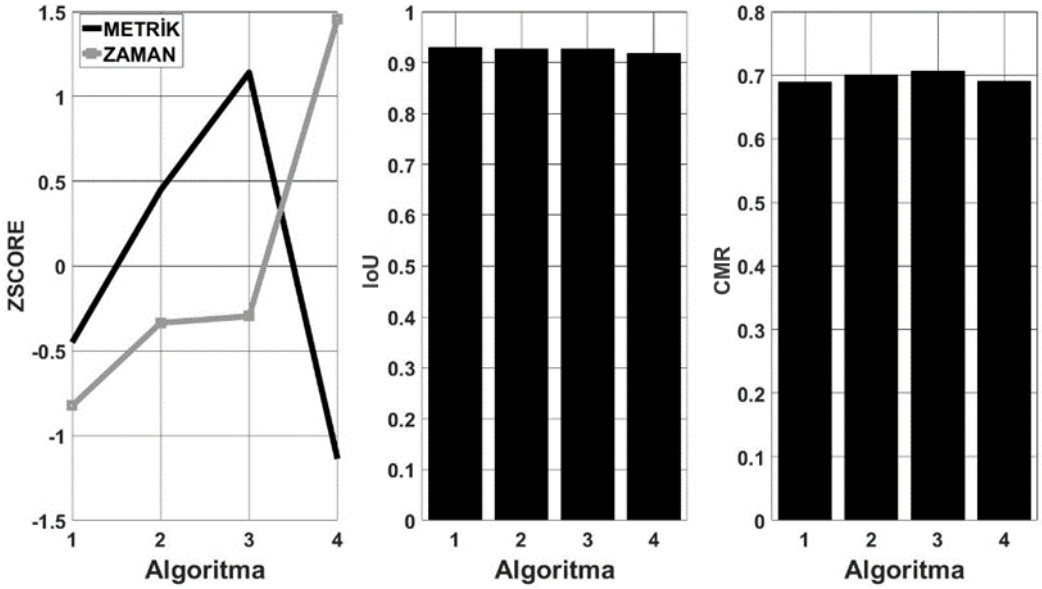
Şekil 6.49. İnterpolasyon algoritmalarının Oxford veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir (TZ ve EZ).



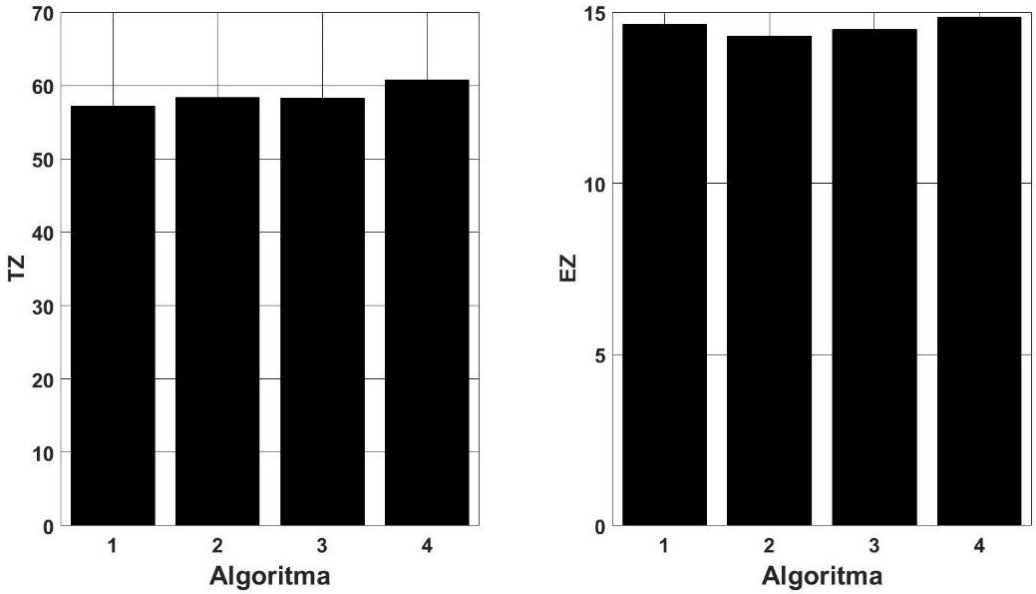
Şekil 6.50. İnterpolasyon algoritmalarının HPatches veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir (ZSCORE, IoU ve CMR).



Şekil 6.51. İnterpolasyon algoritmalarının HPatches veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmış ve r değeri için 2 seçilmiştir (TZ ve EZ).



Şekil 6.52. İnterpolasyon algoritmalarının HPatches veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir (ZSCORE, IoU ve CMR).



Şekil 6.53. İnterpolasyon algoritmalarının HPatches veri kümesi üzerinde test edilmesi. Giriş görüntüsü iki katına çıkarılmamış ve r değeri için 2 seçilmiştir (TZ ve EZ).

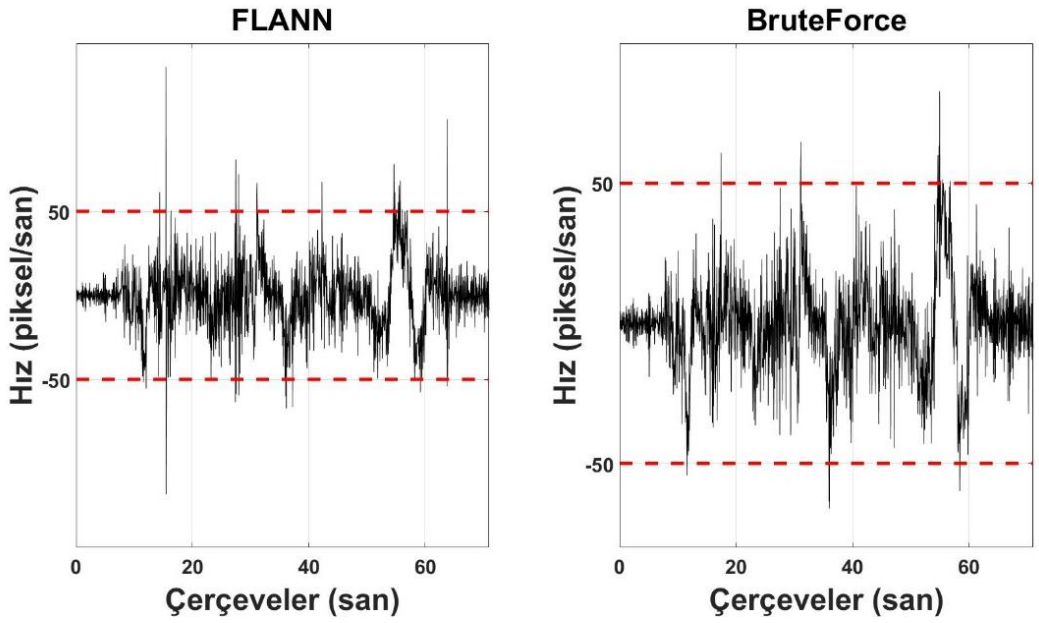
6.3. GERÇEK ZAMANLI SSA FİLTRESİNİN DENEY SONUÇLARI

Bu bölümde ilk olarak, SIFT algoritması ile iki farklı eşleştirme algoritması kullanıldığında sonuç üzerindeki etkisi video üzerinde test edilmiştir. OpenCV'nin BruteForce eşleştirme algoritması ile FlannBasedMatcher algoritması karşılaştırılmıştır. Daha sonra da RT-SSA algoritmasının tek parametresi olan pencereleme uzunluğu L'nin fraklı değerleri için filtreleme sonuçları verilmiştir. RT-SSA algoritması, Eigen kütüphanesi kullanarak C/C++ programlama dili ile yazılmıştır.

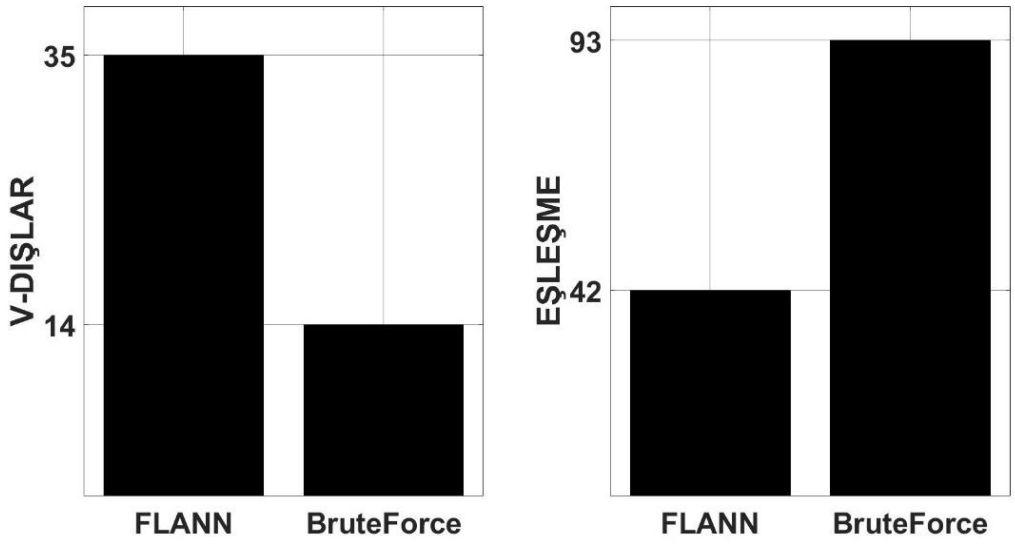
SIFT algoritmasında, daha önceki bölümlerin sonuçları göz önünde bulundurularak, öznicelik hesaplama metodu olarak basit HOG (Bkz. Bölüm 3), GSSP uzayının hesaplanmasında TDGC algoritması için Eşitlik 2.27 ile verilen denklemdeki Hata değeri 10^{-3} (Bkz. Bölüm 2) ve GSSP uzayının ilk görüntüsü olan çekirdek görüntünün bulanıklık seviyesi $sigma = 1.0$ olarak alınmış ve başlangıç görüntüsü iki katına çıkarılmıştır.

6.3.1. Eşleştirme Algoritmalarının Karşılaştırılması

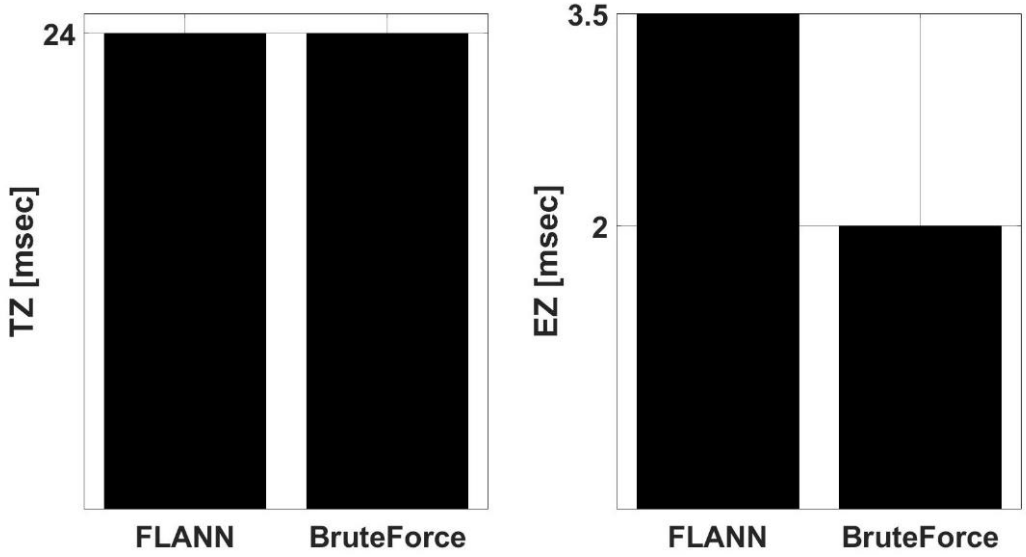
Şekil 6.54, OpenCV'nin FLANN ve BruteForce eşleştirme metotları ile elde edilen video üzerindeki hız yörüngelerini göstermektedir. Bu yörüngelere ait metrik değerleri ise Şekil 6.55, 6.56'de verilmiştir. Bu sonuçlardan, BruteForce eşleştirme yönteminin FLANN eşleştirme yöntemine göre video üzerinde daha iyi bir sonuç verdiğini anlıyoruz. BruteForce eşleştirme metodu daha fazla (FLANN'a göre yaklaşık iki kat) anahtar nokta kullanarak daha düzgün bir yörünge oluşturmuştur. Ancak bu anahtar-nokta fazlalığı, EZ değerini etkilememiştir. FLANN ve BruteForce eşleştirme metotlarının arasındaki 1,5 milisaniyelik fark ölçümden ölçüme değişebilecek çok küçük bir farktır. TZ ve EZ değerlerinin toplamına baktığımızda, tüm SIFT algoritmasının eşleştirme algoritması ile birlikte 640x480 çözünürlüğündeki bir video kaydı üzerinde 30 FPS 'nin üzerinde çalışabileceğini görüyoruz.



Şekil 6.54. OpenCV'nin FLANN ve BruteForce eşleştirme metotları ile elde edilen video üzerindeki hız yörüngüleri.



Şekil 6.55. OpenCV'nin FLANN ve BruteForce eşleştirme metotları ile elde edilen video üzerindeki hız yörüngesine ait metrikler (V-DIŞLAR ve EŞLEŞME).

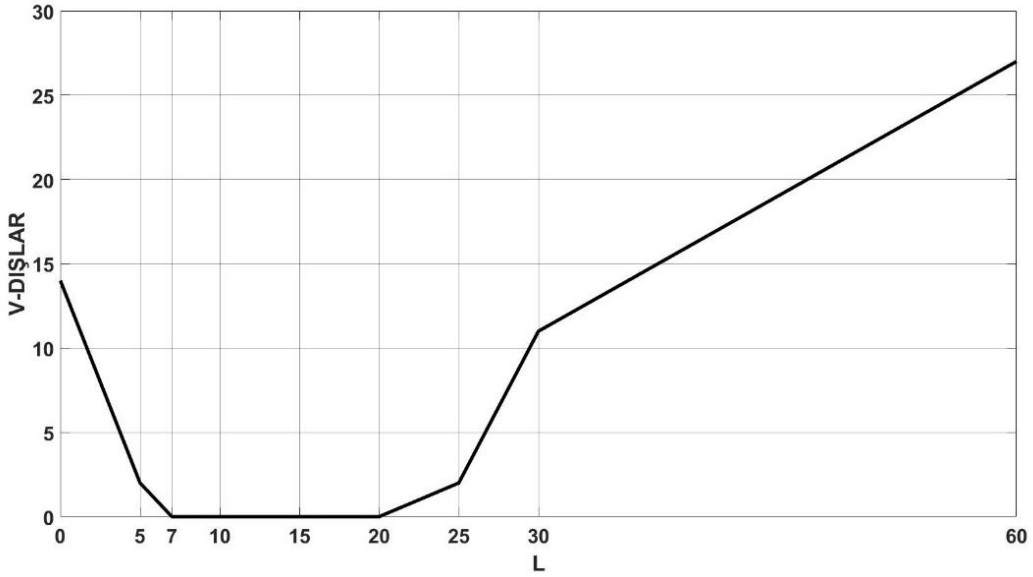


Şekil 6.56. OpenCV'nin FLANN ve BruteForce eşleştirme metotları ile elde edilen video üzerindeki hız yörüngesine ait metrikler (TZ ve EZ).

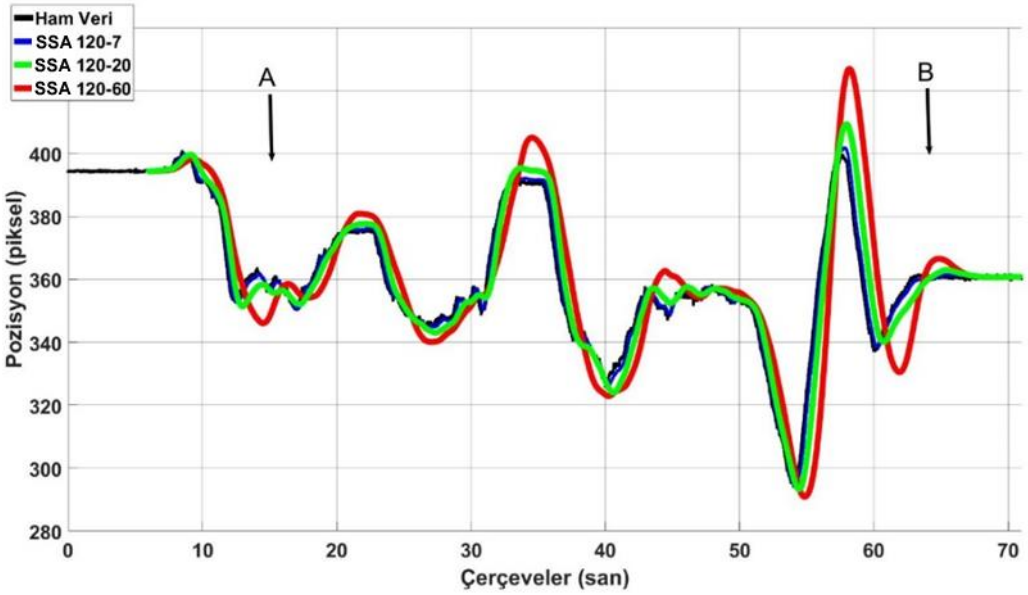
6.3.2. RT-SSA ile Filtreleme Sonuçları

Bir önceki bölümün sonucuna göre bu bölümde SIFT algoritması ile BruteForce eşleştirme algoritması kullanılmıştır. RT-SSA algoritmasında buffer uzunluğu 120 olarak belirlendiğinde, farklı pencereleme uzunluk değerlerine göre, yörünge üzerindeki V-DIŞLAR metriğinin aldığı değerler Şekil 6.57'de verilmiştir.

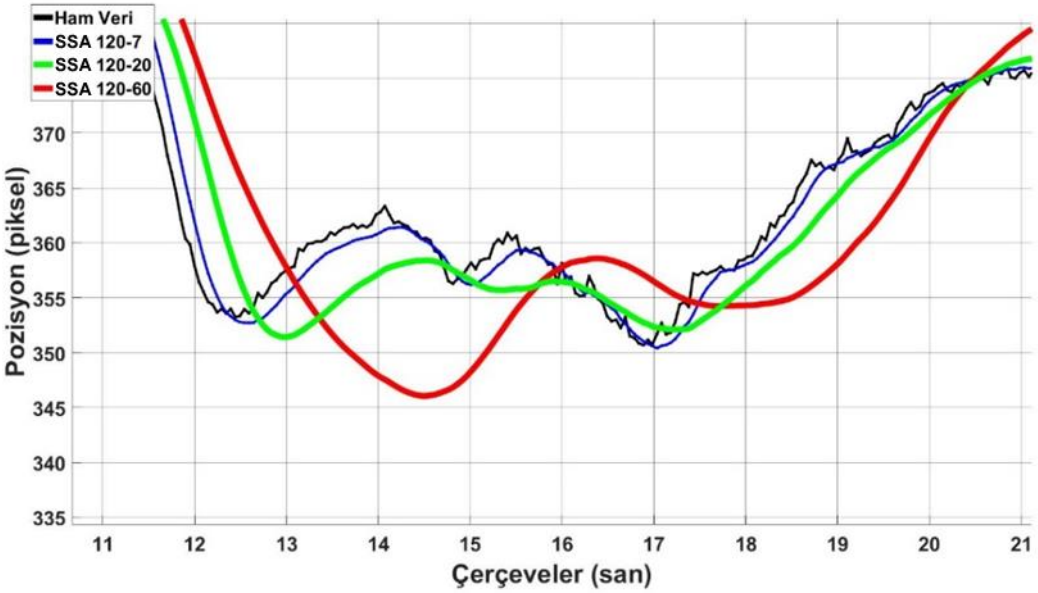
Pencereleme uzunluğu sıfır iken verilen V-DIŞLAR metrik değeri filtrelenmemiş ham yörüngeden elde edilmiş değerdir. Pencereleme uzunluğu arttıkça, filtreleme (veya yumuşatma) oranı da artmaktadır. Pencereleme uzunluğu 7 olduğunda V-DIŞLAR değeri sıfır olmakta ve pencereleme uzunluğu 20 olana kadar bu şekilde kalmaktadır. Pencereleme uzunluğu 20'den sonra V-DIŞLAR değerinin artması filtrenin aşırı düzeltme (over-smoothing) yapmasından kaynaklanmaktadır. Bu durumda, filtre değerleri özellikle nesnenin hızının ani değiştiği noktaların civarında, gerçeklikten oldukça sapmaktadır (Şekil 6.58 – 6.60).



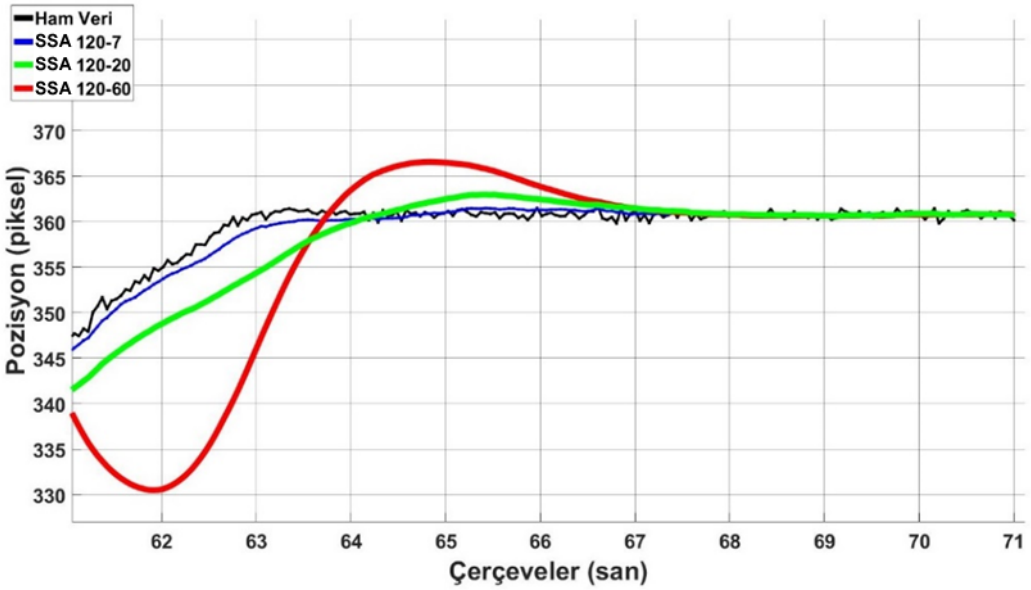
Şekil 6.57. RT-SSA algoritmasında buffer uzunluğu 120 olarak belirlendiğinde, farklı pencereleme uzunluk değerlerine göre, yörünge üzerindeki V-DIŞLAR metriğinin aldığı değerler.



Şekil 6.58. RT-SSA algoritmasında buffer uzunluğu 120 olduğunda farklı pencere uzunluklarına (7, 20, 60) göre filtreleme sonuçları.



Şekil 6.59. RT-SSA algoritmasında buffer uzunluğu 120 olduğunda farklı pencere uzunluklarına (7, 20, 60) göre filtreleme sonuçları. (Şekil 6.58'deki A noktazına zum yapılmıştır).



Şekil 6.60. RT-SSA algoritmasında buffer uzunluğu 120 olduğunda farklı pencere uzunluklarına (7, 20, 60) göre filtreleme sonuçları. (Şekil 6.58'deki B noktazına zum yapılmıştır)

BÖLÜM 7

SONUÇ VE HEDEFLER

7.1. GAUSS ÖLÇEK UZAY PRAMİDİ ALGORİTMASI İÇİN SONUÇ VE HEDEFLER

Bu bölümün literatüre olan katkılarını aşağıdaki gibi sıralayabiliriz;

1. Gauss fonksiyonunun yarı grup özelliği ve ısı transfer fonksiyonundan yararlanılarak, GSSP'nin görüntü üyelerinin bulanıklaştırma parametrelerinin nasıl hesaplanması gerektiği açıklanmıştır.

2. TDGC'nin örneklem uzunluğunu belirleme metodu, Erf'yi temel alarak açıklanmıştır.

3. Video üzerinde, doğru referans homografileri olmadan, nesne tanımlama algoritmasının gürbüzlüğü hareket takip yörüngesinin hızı temel alınarak hesaplanmıştır.

4. Giriş görüntüsü üzerinde var olduğu farz edilen başlangıç bulanıklaştırma seviyesinin etkisi SIFT algoritmasının performansı üzerinde iki farklı veri kümesi ve video üzerinde test edilmiştir. Bu varsayımın herhangi bir faydasının olmadığı gözlemlenmiştir.

5. TDGC'nin örneklem uzunluğunu belirleyen Erf tabanlı denklem farklı hata tolerans değerleri ile iki farklı veri kümesi ve video üzerinde test edilerek, SIFT algoritmasının performansı üzerindeki etkisi gözlemlenmiştir. Test sonuçları bu tür bir hesaplamanın oldukça önemli olduğunu göstermiştir.

Tüm SIFT algoritması, bir hiper parametre uzayı tanımlamaktadır. Genellikle bir parametreyi artırıp diğerini azaltmak aynı performans sonuçları göstermektedir. Bu çalışmada etkisini gözlemlemek istediklerimiz dışındakiler sabit tutulmuştur.

Bu bölümde, TDGC algoritmasının örneklem uzunluğunun önemli bir parametre olduğunu gösterdik. Örneklem uzunluğu, Erf'yi temel alarak, bulanıklaşma seviyesi (sigma) ve hata toleransının bir fonksiyonu olan bir denklem ile hesaplanmıştır (Bkz. Eşitlik 2.27).

Örneklem uzunluğu sigma ile doğru ve hata toleransı ile ters orantılı bir şekilde artmaktadır. Algoritma, büyük hata toleransı için, örneğin 10^{-2}, küçük örneklem uzunluğu hesaplamaktadır. Bu da konvolüsyonun daha hızlı gerçekleşmesini sağlamaktadır. Ancak, böyle bir durumda, konvolüsyonunun doğruluk oranı düştüğünden dolayı birçok gereksiz anahtar nokta ortaya çıkmakta ve sonuç olarak tüm nesne tanımlama algoritmasının hesaplama zamanı artmaktadır.

GSSP uzayı hesaplaması, SIFT algoritmasının performansı üzerinde oldukça önemli bir etkisi vardır. Anahtar noktalar, GSSP uzayından elde edilen DoG uzayı üzerindeki ekstremum noktalardır. Dolayısı ile de, tüm algoritmanın hızı ve gürbüzlüğü GSSP hesaplamasında kullanılan DGC algoritmasına bağlıdır. Gauss fonksiyonunun bir yaklaşımı olarak, tekrarlı kutu filtre algoritmaları genellikle konvolüsyonun hızını artırmak için tercih edilmektedir. Ancak bu durumda, doğruluk oranı düştüğü için birçok gereksiz anahtar nokta ortaya çıkarak tüm nesne tanımlama algoritmasının hızını düşürebilmektedir.

Bu bölümde giriş görüntüsü üzerinde var olduğu farz edilen başlangıç bulanıklık seviyesinin etkisi algoritmadan kaldırılarak test edilmiştir. Giriş görüntüsünün bulanıklık seviyesini tahmin eden bir algoritmanın SIFT algoritmasına eklenmesi yararlı olabilir.

Diğer taraftan, giriş görüntüsünün boyutunun iki katına çıkarılması algoritmanın hızını etkileyen oldukça önemli bir etkendir. Görüntünün boyutunun artması konvolüsyon hızını azaltmaktadır. Giriş görüntüsünün boyutunun iki katına çıkarılmasının, veri

kümeleri üzerinde önemli bir faydasının olmadığı gözlemlenmiştir. Ancak, video üzerindeki test sonuçları giriş görüntüsünün boyutunun iki katına çıkarılması ile önemli bir gürbüzlük artışı göstermiştir. Bu farklı sonuçlar veri kümeleri ve video arsındaki çözünürlük farklılığından dolayı ortaya çıkıyor olabilir. Dolayısı ile giriş görüntüsünün iki katına çıkarılıp çıkarılmayacağına görüntü çözünürlüğüne göre karar verilmelidir.

7.2. ÖZNİCELİKLERİN HESAPLAMA ALGORİTMASI İÇİN SONUÇ VE HEDEFLER

Bu bölümün literatüre olan katkılarını aşağıdaki gibi sırlayabiliriz;

1. SIFT algoritmasının öznicelik hesaplama metodunda kullanılan özel gruplama prosesi algoritması açıklanarak verilmiştir.

2. Özniceliklerin hesaplandığı yamanın genişliği (r) deneysel olarak belirlenmiştir.

3. HOG algoritması ile kullanılabilecek interpolasyon algoritmaları; hücre içi lineer interpolasyon, hücre içi trilinear interpolasyon ve hücreler arası trilinear interpolasyon algoritmaları verilmiştir.

4. Oxford ve HPatches veri kümeleri üzerinde yapılan karşılaştırmada interpolasyonun SIFT algoritmasının performansı üzerinde çok önemli bir katkısının olmadığı gözlemlenmiştir.

Burada, HOG algoritmasının SIFT algoritmasında kullanımının, insan tanımlamadaki kullanımından farklı olduğunu hatırlanmalıdır. SIFT algoritmasında öznicelik vektörleri, gelişmiş bir öznicelik algoritması tarafından belirlenen anahtar noktaların her biri için hesaplanmaktadır. Dolayısıyla anahtar noktaların hesaplanma yöntemi özniceliklerin gürbüzlüğünü desteklemektedir.

Basit HOG algoritması, anlaşılması ve dolayısıyla da uygulanması kolay bir algoritmadır. Trilineer interpolasyona göre daha sade bir algoritma olan, lineer interpolasyon ile HOG algoritması da tercih edilebilir. Algoritmaların basitliği, özellikle de paralel hale getirilecekleri zaman önem arz etmektedir.

Bu çalışmada, Oxford ve HPatches veri kümeleri üzerinde gösterilmiş ortalama test sonuçları verilmiştir. Literatürde SIFT algoritmasının performansı üzerine rapor edilen çalışmalar uygulama alanına göre değişiklik gösterebilmektedir. Örneğin, yüz tanımlama ve uydu görüntüleri birbirlerinden farklı karakteristiklerde olduğu için farklı performans sonuçları gösterebilmektedir. Uygulamada kullanılan görüntülerin çözünürlüğüne bağlı olarak, performans değişiklik gösterebilmektedir. Dolayısı ile herhangi bir uygulamada her zaman optimum parametre ve algoritma seçimi için bir performans testinin yapılması faydalı olacaktır.

7.3. GERÇEK ZAMANLI SSA FİLTRESİ İÇİN SONUÇ VE HEDEFLER

Bu bölüm gerçek zamanlı nesne takibi uygulaması için SIFT nesne tanımlama algoritması ile, filtreleme metodu olan SSA'yı bir araya getirmiştir. Bölümün katkılarını aşağıdaki gibi sıralayabiliriz;

1. FLANN ve Brute-Force eşleşme metotları birbirleri ile karşılaştırılmıştır. Video üzerindeki deney sonuçları, düşük çözünürlükteki görüntüler ile Brute-Force eşleştirme algoritmasının kullanımının daha iyi bir sonuç verdiğini göstermiştir.

2. Gerçek zamanlı bir uygulamada kullanılabilecek orijinal bir SSA algoritması tanımlanmıştır (RT-SSA). Sabit bir buffer kullanılarak, SSA algoritmasının son adımı olan geri kazanım aşamasındaki karmaşık köşegensel ortalama işlemine gerek olmadığı gösterilmiştir.

3. RT-SSA algoritmasının tek giriş parametresi olan pencereleme uzunluğunun farklı değerlerine göre filtreleme (yumuşatma veya düzeltme) oranı V-DIŞLAR metriği ile ölçülmüştür. Pencereleme uzunluğu arttıkça düzelmenin

arttığı ancak belirli bir noktadan sonra aşırı düzelmeden (over-smoothing) dolayı (özellikle nesnenin hızındaki ani değişikliklerin olduğu noktalarda), filtrelenmiş verinin, nesnenin gerçek pozisyonundan saptığı gözlemlenmiştir.

Bu bölümde elde edilen sonuçlara göre, RT-SSA algoritmasının tek giriş parametresi olan pencereleme uzunluğu, V-DIŞLAR metriği temel alınarak otomatik olarak belirlenebilir. Böylece hızın sabit veya ivmenin sıfır olduğu yerlerde, yüksek pencereleme uzunluğu ve hızın değiştiği noktalarda ise aşırı filtrelemeden kaçınmak için dengeli bir pencereleme uzunluğu seçilecek şekilde RT-SSA algoritması geliştirilebilir.

Bu çalışmada sabit buffer uzunluğu, 120 olarak belirlenmiştir. Bu buffer uzunluğu ile RT-SSA algoritmasının hesaplama zamanı bir milisaniyenin altında gerçekleşmektedir. Buffer uzunluğu mümkün olduğunca büyük alınabilir. Ancak RT-SSA algoritması buffer'ın dolması için belirli bir bekleme süresi gerektirmektedir. Buffer dolma süresini kısa tutup (10 çerçeve gibi), daha sonra belirli bir uzunluğa kadar büyüyecek şekilde algoritmada değişiklik yapılabilir.

KAYNAKLAR

1. Mikolajczyk, K. and Schmid, C., "A performance evaluation of local descriptors", ***IEEE Transactions On Pattern Analysis and Machine Intelligence***, 27 (10): 1615–1630 (2005).

2. Liu, H., Hong, T.-H., Herman, M., Camus, T., and Chellappa, R., "Accuracy vs Efficiency Trade-offs in Optical Flow Algorithms", ***Computer Vision and Image Understanding***, 72 (3): 271–286 (1998).

3. Acharya, K., Venkatesh Babu, R., and Vadhiyar, S. S., "A real-time implementation of SIFT using GPU.", ***Journal of Real-Time Image Processing***, 14 (2): 267–277 (2018).

4. Wang, Z. and Ban, T., "Design, Implementation, and Evaluation of Stochastic FIR Filters Based on FPGA", ***Circuits, Systems, And Signal Processing***, 1–21 (2022).

5. Crow, F. C., "Summed-area tables for texture mapping.", ***Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques***, 207–212 (1984).

6. Lowe, D. G., "Object recognition from local scale-invariant features", ***Proceedings of The Seventh IEEE International Conference On Computer Vision***, 1150–1157 vol.2 (1999).

7. Lowe, D. G., "Distinctive image features from scale-invariant keypoints", ***International Journal of Computer Vision***, 60 (2): 91–110 (2004).

8. Loncomilla, P., Ruiz-del-Solar, J., and Martínez, L., "Object recognition using local invariant features for robotic applications: A survey", ***Pattern Recognition***, 60: 499–514 (2016).

9. Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., "Speeded-Up Robust Features (SURF)", ***Computer Vision and Image Understanding***, 110 (3): 346–359 (2008).

10. Azad, P., Asfour, T., and Dillmann, R., "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition", ***2009 IEEE/RSJ International Conference on Intelligent Robots and Systems***, 4275–4280 (2009).

11. Birk, J., Kelley, R., Chen, N., and Wilson, L., "Image Feature Extraction Using Diameter-Limited Gradient Direction Histograms", ***IEEE Transactions On Pattern Analysis and Machine Intelligence***, PAMI-1 (2): 228–235 (1979).

12. Dalal, N., "Finding People in Images and Videos", Phdthesis, ***Institut National Polytechnique de Grenoble - INPG***, (2006).

13. Witkin, A. P., "Scale-space filtering: A new approach to multi-scale description", ***ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing***, vol. 9, 150-153, 1984.

14. Koenderink, J. J., "The structure of images.", ***Biological Cybernetics***, 50 (5): 363–370.

15. Lindeberg, T., "On the Axiomatic Foundations of Linear Scale-Space", Gaussian Scale-Space Theory, ***Springer Netherlands***, Dordrecht, 75–97 (1997).

16. Getreuer, P., "A Survey of Gaussian Convolution Algorithms", ***Image Processing On Line***, 2013 (3): 286–310.

17. Romeny, B. M. H., "Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, Written in Mathematica", ***Springer Science & Business Media***, 470 (2008).

18. Ke, Y. and Sukthankar, R., "PCA-SIFT: a more distinctive representation for local image descriptors", ***Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004***, vol. 2.

19. Juan, L. and Gwun, O., "A comparison of sift, pca-sift and surf.", ***International Journal of Image Processing (IJIP)***, 3 (4): 143–152 (2009).

20. Sima, A. A. and Buckley, S. J., "Optimizing SIFT for Matching of Short Wave Infrared and Visible Wavelength Images", ***Remote Sensing***, 5 (5): 2037–2056 (2013).

21. Alonso-Fernandez, F., Tome-Gonzalez, P., Ruiz-Albacete, V., and Ortega-Garcia, J., "Iris recognition based on SIFT features", ***2009 First IEEE International Conference on Biometrics, Identity and Security (BIdS)***, 1–8 (2009).

22. Fan, B., Huo, C., Pan, C., and Kong, Q., "Registration of Optical and SAR Satellite Images by Exploring the Spatial Relationship of the Improved SIFT", ***IEEE Geoscience and Remote Sensing Letters***, 10 (4): 657–661 (2013).

23. Gajjar, B. and Patani, H. M. A., "Parameterizing sift and sparse dictionary for svm based multi-class object classification.", ***International Journal of Artificial Intelligence***, 2021 (19): 95–108 (2021).

24. Chi Qin, L. A. I. and Teoh, S. S., "An efficient method of HOG feature extraction using selective histogram bin and PCA feature reduction", ***Advances in Electrical and Computer Engineering***, 16 (4): 101–108 (2016).

25. Raxle Wang, C.-C. and Lien, J.-J. J., "AdaBoost Learning for Human Detection Based on Histograms of Oriented Gradients", ***Computer Vision – ACCV 2007***, Berlin, Heidelberg, 885–895 (2007).

26. Kim S. and Cho K., "Fast Calculation of Histogram of Oriented Gradient Feature by Removing Redundancy in Overlapping Block", ***Journal of Information Science and Engineering***, 30 (6): 1719–1731 (2014).

27. Abbass, M. Y., Kwon, K. C., and Abdelwehab, S. A., "A survey on online learning for visual tracking.", ***The Visual Computer***, 37 (5): 993–1014 (2021).

28. Masood, H., Zafar, A., Umair Ali, M., Attique Khan, M., Ahmed, S., Tariq, U., Kang, B.-G., and Nam, Y., "Recognition and tracking of objects in a clustered remote scene environment", ***Computers, Materials and Continua***, 70 (1): 1699–1719 (2022).

29. Zuo, J., Jia, Z., Yang, J., and Kasabov, N., "Moving Target Detection Based on Improved Gaussian Mixture Background Subtraction in Video Images", ***IEEE Access***, 7: 152612–152623 (2019).

30. Yilmaz, A., Javed, O., and Shah, M., "Object tracking: A survey", ***ACM Computing Surveys***, 38 (4): 1-45 (2006).

31. Zhou, H., Yuan, Y., and Shi, C., "Object tracking using SIFT features and mean shift", ***Computer Vision and Image Understanding***, 113 (3): 345–352 (2009).

32. Jabar, F., Farokhi, S., and Sheikh, U. U., "Object tracking using SIFT and KLT tracker for UAV-based applications", ***2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)***, 65–68 (2015).

33. More, P. and Mishra, P., "Enhanced-PCA based Dimensionality Reduction and Feature Selection for Real-Time Network Threat Detection", ***Engineering, Technology & Applied Science Research***, 10 (5): 6270–6275 (2020).

34. "Particularities and Commonalities of Singular Spectrum Analysis as a Method of Time Series Analysis and Signal Processing - Golyandina - 2020 - WIREs

Computational Statistics - Wiley Online Library", **https://wires.onlinelibrary.wiley.com/doi/full/10.1002/wics.1487** (2022).

35. Ozturk, A., Tartar, A., Ersoz Huseyinsinoglu, B., and Ertas, A. H., "A clinically feasible kinematic assessment method of upper extremity motor function impairment after stroke", ***Measurement***, 80: 207–216 (2016).

36. Golyandina, N., "Particularities and commonalities of singular spectrum analysis as a method of time series analysis and signal processing.", ***Wiley Interdisciplinary Reviews: Computational Statistics***, 12 (4): 1-39 (2020).

37. Ansari, K., "Real-Time Positioning Based on Kalman Filter and Implication of Singular Spectrum Analysis", ***IEEE Geoscience and Remote Sensing Letters***, 18 (1): 58–61 (2021).

38. "Real-Time Unified Single- and Multi-Channel Structural Damage Detection Using Recursive Singular Spectrum Analysis - Basuraj Bhowmik, Manu Krishnan, Budhaditya Hazra, Vikram Pakrashi, 2019", **https://journals.sagepub.com/doi/abs/10.1177/1475921718760483** (2022).

39. Shulz, T., .Gaussian Cumulative Distribution Function, **https://www.youtube.com/watch?v=iIzpWScAj_o&t=14s** (2022)**.**

40. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S., "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression", ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition***, 658–666 (2019).

41. Clipper - an open-source freeware polygon clipping library, **http://www.angusj.com/delphi/clipper.php** (2022)**.**

42. Muja, M. and Lowe, D. G., "Fast approximate nearest neighbors with automatic algorithm configuration.", ***VISAPP (1)***, 331–340 (2009).

43. Tuytelaars, T. and Mikolajczyk, K., "Local Invariant Feature Detectors: A Survey", ***Now Publishers Inc***, 122 (2008).

44. Balntas, V., Lenc, K., Vedaldi, A., and Mikolajczyk, K., "HPatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors", ***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition***, 5173–5182 (2017).

45. He, J. and Tian, C.-X., "A statistical smoothness measure to eliminate outliers in motion trajectory tracking", ***Human Movement Science***, 17 (2): 189–200 (1998).

46. Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., "Speeded-up robust features (SURF)", ***Computer vision and image understanding***, 110 (3): 346-359 (2008).

47. Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An efficient alternative to SIFT or SURF", ***2011 International conference on computer vision***, IEEE, 2564-2571 (2011).

48. Rosten, E. and Drummond, T., "Machine learning for high-speed corner detection", ***European conference on computer vision***, Springer, Berlin, Heidelberg, 430-443, (2006).

49. Colonder, M., Lepetit, V., Strecha, C., and Funa P., "Brief: Binary robust independent elementary features", ***European conference on computer vision***, Springer, Berlin, Heidelberg, 778-792, (2010).

ÖZGEÇMİŞ

Ali ÖZTÜRK, 2006 senesinde Uludağ Üniversitesi Matematik bölümünden lisans derecesi, 2013 senesinde Tohoku Üniversitesi, Robotik ve Biyomühendislik bölümünden master derecesi almıştır. 2014-2019 seneleri arasında Karabük Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği bölümünde araştırma görevlisi olarak çalışmış, 2019 senesinden bu yana Muş Alparslan Üniversitesi, TBMYO, Makine ve Metal Teknolojileri bölümünde öğretim görevlisi olarak görev yapmaktadır. İlgi alanları insan hareket analizi, veri analizi, nesne tanımlama ve takibidir.