# NESNE TANIMADA KULLANILAN POPÜLER ÖZNİTELİKLERİN KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Kaan Yasin KOCAMAN

Danışman

Dr. Öğr. Üyesi Celal Onur GÖKÇE

BİLGİSAYAR ANABİLİM DALI

Ocak 2025

**AFYON KOCATEPE ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**YÜKSEK LİSANS TEZİ**

# NESNE TANIMADA KULLANILAN POPÜLER ÖZNİTELİKLERİN KULLANILMASI

**Kaan Yasin KOCAMAN**

**Danışman**

**Dr. Öğr. Üyesi Celal Onur GÖKÇE**

**BİLGİSAYAR ANABİLİM DALI**

**Ocak 2025**

**TEZ ONAY SAYFASI**

Kaan Yasin KOCAMAN tarafından hazırlanan "Nesne Tanımada Kullanılan Popüler Özniteliklerin Karşılaştırılması" adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 31 / 01 / 2025 tarihinde aşağıdaki jüri tarafından **oy birliği** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Anabilim Dalı'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

**Danışman** **:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

**Başkan** **:** Prof. Dr. Hakkı Gökhan İLK

TED Üniversitesi, Mühendislik Fakültesi … İmza…

**Üye** **:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

Afyon Kocatepe Üniversitesi, Mühendislik Fakültesi … İmza…

**Üye** **:** Doç. Dr. Gür Emre GÜRAKSIN

Afyon Kocatepe Üniversitesi, Mühendislik Fakültesi … İmza …

Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

..... /..... /..... tarih ve

……. sayılı kararıyla onaylanmıştır.

…….

Prof. Dr. Bekir YALÇIN

Enstitü Müdürü

## BİLİMSEL ETİK BİLDİRİM SAYFASI

### Afyon Kocatepe Üniversitesi

**Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;**

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

**beyan ederim.**

**09 / 01 / 2025**

İmza

Kaan Yasin KOCAMAN

# ÖZET

Yüksek Lisans Tezi

NESNE TANIMADA KULLANILAN POPÜLER ÖZNİTELİKLERİN KARŞILAŞTIRILMASI

Kaan Yasin KOCAMAN
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Anabilim Dalı
**Danışman:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

Bu araştırmada, nesne tanımada kullanılan popüler özniteliklerin karşılaştırılması yapılmıştır. Teknolojinin gelişmesi, beraberinde bilgisayarla görme görevlerindeki yenilikleri de beraberinde getirmiştir. Bilgisayarla görme ve nesne algılama günümüzde otonom araçlar, yüz tanıma, tıbbi görüntüleme ve daha birçok alanda kullanılmaktadır. Bu çalışmada nesne tanımada kullanılan üç popüler öznitelik ve iki veri seti kullanılmıştır. Özniteliklerin, veri setlerinde gösterdikleri performanslar ve farklı veri setlerinin öznitelik kalitesine etkisi incelenmiştir. İnceleme sonucuna göre öznitelik kaliteleri, farklı veri setlerinde değişkenlik göstermiştir. Bu çalışmanın veri setleri VisDrone veri seti ve MNIST veri seti olarak belirlenmiştir. Öznitelik çıkarma yöntemleri ise HOG, SIFT ve ORB öznitelik çıkarıcılar olarak belirlenmiştir. VisDrone veri setinde en iyi performansı SIFT öznitelik çıkarıcı sergilemiştir. MNIST veri setinde en iyi performans ORB öznitelik çıkarıcıya aittir.

**2025, ix +111 Sayfa**

**Anahtar Kelimeler:** Nesne tanıma, Öznitelik çıkarma, Veri seti, Öznitelik kalitesi.

## ABSTRACT

M.Sc. Thesis

COMPARISON OF POPULAR FEATURES USED IN OBJECT RECOGNITION

Kaan Yasin KOCAMAN

Afyon Kocatepe University

Graduate School of Natural and Applied Sciences

Department of Computer

**Supervisor:** Asst. Prof. Celal Onur GÖKÇE

In this research, a comparison of popular features used in object recognition was made. The development of technology has brought innovations in computer vision tasks. Computer vision and object detection are currently used in autonomous vehicles, face recognition, medical imaging and many other areas. Three popular features and two data sets used in object recognition were used in the study. The performances of the features in the data sets and the effects of different data sets on the feature quality were examined. According to the results of the examination, the feature qualities varied in different data sets. The data sets of this study were determined as VisDrone data set and MNIST data set. The feature extraction methods were determined as HOG, SIFT and ORB feature extractors. SIFT feature extractor exhibited the best performance in VisDrone data set. ORB feature extractor had the best performance in MNIST data set.



**2025, ix +111 Pages**

**Keywords:** Object recognition, Feature extraction, Dataset, Feature quality.

## TEŞEKKÜR

Bu araştırmanın konusu, deneysel çalışmaların yönlendirilmesi, sonuçların değerlendirilmesi ve yazımı aşamasında yapmış olduğu büyük katkılarından dolayı tez danışmanım Sayın Dr. Öğr. Üyesi Celal Onur GÖKÇE' ye her konuda öneri ve eleştirileriyle yardımlarını gördüğüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Bu araştırma boyunca maddi ve manevi desteklerinden dolayı aileme teşekkür ederim.

Kaan Yasin KOCAMAN

Afyonkarahisar 2025

İÇİNDEKİLER DİZİNİ

**Sayfa**

## SİMGELER ve KISALTMALAR DİZİNİ

| | |
|---|---|
| BRIEF | Binary Robust İndependent Elementary Features |
| CNN | Convolutional Neural Networks |
| GMM | Gaussian Mixture Model |
| HOG | Histogram of Oriented Gradients |
| LBP | Local Binary Pattern |
| LVQ | Learning Vector Quantization |
| MNIST | Modified National Institute of Standards and Technology database |
| ORB | Oriented FAST ve Rotated BRIEF |
| RANSAC | Random Sample Consensus |
| SIFT | Scale Invarıant Feature Transform |
| SURF | Speed Up Robust Features |
| SVM | Support Vector Machine |
| YOLO | You Only Look Once |

# TABLOLAR DİZİNİ

# RESİMLER DİZİNİ

# 1. GİRİŞ

İnsan zekasına en yakın tasarım olarak adlandırılan yapay zeka, aynı insan zekasına benzer şekilde algılama, bilgi toplama, güncelleme ve iletişim kurma gibi insan zekasına özgü fonksiyonları taklit edebilen yazılımlardır (Kavuncu 2018). Makinelerin yazılımlarla birleşmesiyle, tıpkı insanlar gibi düşünebilen, algılayabilen ve iletişim kurabilen tasarımlar olması hedeflenmiştir. Yapay zeka ilk defa karşılaştığı bir olay ya da durumda, insanlar gibi geçmişten öğrendikleri bilgilerden ve yöntemlerden yola çıkarak doğru kararlar verebilir (Erbir 2022). Nesne tanıma, gelişen teknoloji ile birlikte bilgisayarlı görü alanında oldukça popüler bir alan olmaya devam etmektedir. Geçmişten günümüze birçok nesne tespit yöntemleri geliştirilmiştir. Günümüzde hala gelişmeye açık olan nesne tespitinde kullanılan yöntemler, avantajlarının yanı sıra dezavantajlar da barındırmaktadır.

Teknolojinin gelişmesi, bilgisayarla görme görevlerindeki yenilikleri de beraberinde getirmiştir. Bilgisayarla görme ve nesne algılama günümüzde otonom araçlar, yüz tanıma, tıbbi görüntüleme ve daha birçok alanda kullanılmaktadır. Resimlerdeki ve videolardaki nesneleri tanıma bilgisayarla görme ve makine öğreniminde en çok dikkat çeken başarılardandır. Nesne algılama, nesnenin konumunu belirleyebilir, boyutunu ölçebilir, sınıflandırabilir ve takip edebilir. Bu başarının en önemli faktörleri büyük veri kümeleridir (Gauen vd. 2017). Teknolojinin gelişmesiyle beraber nesne algılama konusu da bu gelişimden olumlu yönde etkilenmektedir. Ancak günümüzde yine de gelişime ihtiyaç duyulmaktadır (Zaidi vd. 2022). Gerçek dünyadaki birçok etmen insanların nesne algılamasında sorun yaratmamaktadır. Ancak aynı durum bilgisayarlar için söz konusu değildir. Ortam değişimi, arka plan çıkarımı, gürültülü arka plan gibi bir çok ortam değişkeni bilgisayarın nesne tanıma becerisini olumsuz etkilemektedir (Kaya vd. 2020).

Makine öğrenmesi temel olarak bilgisayarlı sistemlerinin insanlar gibi eğitim alarak çeşitli görevleri yerine getirebilmesi olarak tanımlanmaktadır. Çeşitli algoritmalar ve teknikler kullanılarak öğrenim yeteneği kazanan sistemlerin eğitimi için veri seti kullanılır ( Adıyaman vd. 2024). Makine öğrenmesi algoritmaları farklı kaynaklardaki verileri yönetebilir, düzenleyebilir ve birleştirebilir. Bu büyük veri setlerini kullanabilmesi bakımından tahmin yeteneği de güçlüdür (Başer vd. 2021). İnsan gücünün yetemediği durumlarda, veri setinden aldığı eğitimle karmaşık problemlerin çözümünde

ve yüksek boyutlu verilerin işlenmesinde önemli rol oynamaktadır. Günümüzde, gelişen teknolojide önemli bir yeri olan makine öğrenmesi, karmaşık verilerin çözümlenmesi, kullanıcı deneyimi geliştirme, risk azaltma, siber güvenlik gibi alanlarda kullanılmaktadır. Geçmişten gelen verileri kullanarak gelecek için yeni tahminler geliştirebilir.

Bilgisayarlarında insanlar gibi nesne tanıması yoğun bir ihtiyaç haline gelmektedir (Yıldırım vd. 2020). Nesne algılama insanlar için önemsiz gibi görünse de bilgisayarlara öğretmek zorlu bir süreçtir. Bilgisayarın görüş alanında bulunan nesneleri (arabalar, insanlar, sokak işaretleri vb.) sınıflama, bölümleme, hareket tahmini vb. gibi görevler bilgisayarın nesne algılama görevlerindeki temel problemler olmuştur. Viola-Jones dedektörü, Yönlendirilmiş Degradelerin Histogramı HOG (Histogram of Oriented Gradients) vb. modeller ilk nesne algılama modelleri olarak literatürde yerini almıştır (Zaidi vd. 2022). Nesne algılama görüntüdeki nesneleri tanımayı amaçlar. Algoritmalar yardımıyla önceden tanımlanmış nesneleri görüntüde tespit ederek amaca yönelik geri dönüt vermektedir.

Girdideki çıkarılan öznitelikler veya hareket bilgileri nesne tespitinde kullanılan yöntemlerdir. Öznitelik çıkarımında girdideki tespit edilecek nesnenin öznitelikleri (renk, doku, şekil) belirlenir ve görselden ayrılarak nesne tespiti gerçekleşmiş olur. Nesne tespitinde kullanılan yöntemlerden birisi de hareketli nesneden arka plan çıkarımıdır. Videodan alınan peş peşe iki görüntü arasında ki farklar birbirinden çıkarılarak nesnenin tespiti yapılmaktadır. Nesne tespitinde başarılı yöntemler arasında arka plan çıkarma işlemi de yer almaktadır. Girdide istenmeyen arka plan, tespit edilmek istenen nesneden çıkarılarak nesne tespiti gerçekleştirilmiş olur. Nesne tespiti gerçekleştikten sonra elde edilen nesnenin sınıflandırma işlemi gerçekleştirilmektedir. Tespit edilen nesneler arasında benzer öznitelikleri taşıyanlar kümelendirilir. Kümelenen öznitelikler piksel değeri, görüntünün yoğunluk değeri gibi sayısal değerler içerir. Nesne sınıflandırma ve takip işlemlerinde öznitelik çıkarma işlemi oldukça önemlidir. Seçilen öznitelik çıkarma yöntemi, performansı önemli ölçüde etkilerken işlem yükünü de arttırabilmektedir. Bu nedenden dolayı nesne tespiti veya nesne takibi yapılırken anlık işlem gerektiren durumlarda basit öznitelik çıkarma yöntemleri kullanılırken diğer durumlarda daha zor öznitelik çıkarma yöntemleri kullanılmaktadır.

Bu çalışmada nesne tanımada kullanılan popüler öznitelik çıkarıcıların bazı veri setlerinde performansları incelenmiştir. Çalışmada üç popüler öznitelik ve iki veri seti kullanılarak öznitelik kaliteleri ölçülmüş ve karşılaştırılmıştır. VisDrone ve MNIST (Modified National Institute of Standards and Technology database / Değiştirilmiş Ulusal Standartlar ve Teknoloji Enstitüsü veritabanı) veri seti çalışmamızda kullanılan veri setleridir. MNIST veri seti el yazısıyla rakamlardan oluşurken, VisDrone veri seti dronla çekilmiş görüntülerden oluşmaktadır. VisDrone veri seti öznitelik çıkarıcılara girmeden önce görüntü içerisinde nesne sınıflandırma işlemleri yapılmıştır. Her iki veri seti, seçilen üç öznitelik çıkarıcıda işleme alınmıştır. Seçilen öznitelikler HOG, SIFT(Scale Invarıant Feature Transform / Ölçek Değişmez Unsur Dönüşümü), ORB (Scale Invarıant Feature Transform / Ölçek Değişmez Unsur Dönüşümü) olarak belirlenmiştir. Çalışmada, kullanılan özniteliklerin performansı karşılaştırılmıştır. Farklı veri setlerine göre performans değişiklikleri incelenmiştir. Elde edilen sonuçlara göre gelecekteki çalışmalara referans olması amaçlanmıştır. Bilgisayarla görünün yaygınlaşması, sağlık, güvenlik, tarım gibi alanlarda otonom sistemlerin gelişmesiyle nesne tanıma görevleri önemli bir etken haline gelmiştir. Geniş uygulama alanına sahip olan nesne tanıma görevlerinde doğru ve yüksek performans sergileyen nesne tanıma yöntemi seçmek oldukça önemlidir.

## 2. LİTERATÜR TARAMASI

Alhurmuzi (2023) yaptığı çalışmada trafik kazalarını önlemek amacıyla görüntü işleme tekniklerini kullanarak sürücünün yüz görüntüsünden sürücünün yorgunluğunu tespit ederek sürücüye uyarıda bulunan bir sistem geliştirmiştir. Anlık olarak sürücünün yüzündeki değişimler CNN (Convolutional Neural Networks), derin öğrenme ve OpenCV görüntü işleme yöntemleri kullanılarak sürücünün yorgunluğu tespit edilmiştir. Yorgunluğu tespit edilen sürücüye görsel veya sesli mesaj yoluyla bilgilendirme yapılmıştır.

Akçay (2023) yaptığı çalışmada, çok robotlu insansız hava aracı tasarlamıştır. Aracı ile, farklı ortam koşulları sağlayan bir simülasyonda, sanal kamera ile nesne tespit çalışmaları yapmıştır. Elde edilen görüntüler YOLO(You Only Look Once) mimarisi kullanılarak işlenmiş ve yapay sinir ağları ile eğitilmiştir. Yaptığı çalışmada son yıllarda önemi artan dijital ikiz kavramına yer vermiştir. Dijital ikiz kavramı, yapılan çalışmaların maliyet ve risklerini azaltmak için, test sürecinde simülasyon kullanılması olarak literatürdeki yerini almıştır.

Almahdi (2023) çalışmasında AlexNet ve GoogleNet sınıflandırma performansını ölçmüştür. Bu iki evrişimli sinir ağı mimarisinin avantajlarını ve dezavantajlarını analiz etmiştir. Elde ettiği bulguları karşılaştırmış, AlexNet ve GoogleNet' in fotoğraf sınıflandırma süreçlerini incelemiştir. AlexNet performansı altı turluk bir eğitimle rakiplerinden üstün gelmiştir. GoogleNet sürekli eğitim tekrarlarıyla yeterli performans sergilese de başlangıçta rakiplerine göre daha zayıf bir performans sergilemiştir. Uygulayıcıların bilgisayarlı görü ve sınıflandırma işlemleri için uygulanacak yöntemin uygunluğunun karar verilmesinde yardımcı olunması amaçlanmıştır

Abduljabbar (2022) çalışmasında görüntü işleme tekniklerinden yaygın olarak kullanılan toplam varyasyon fonksiyonunu ele almıştır. Görüntü işlemede kullanılan bu yöntem sürekli olarak kullanılsa da, türevlenememe gibi sorunların çözümünü olumsuz etkileyen eksiklikleri vardır. Abduljabbar bu sorunların çözümü için yeni bir yumuşatma tekniğini literatüre sunmuştur. Çeşitli yumuşatma teknikleri geçmişte görüntüyü iyileştirmek ve pürüzsüzleştirmekte kullanılmıştır.

Yanık (2022) çalışmasında görüntü işlme tabanlı mesafe tespit yöntemlerini araştırmıştır. Araştırma sonucu mesafe tespit yöntemlerini olumsuz etkileyen temel sorunları ele almıştır. Ele alınan temel sorunların giderilmesi için yeni bir yöntemin teorisini ortaya koymuştur. Günümüzde endüstriyel amaçlı, nesne takip ve mesafe tespiti içi üretilen görüntü sensörlerinin kullanım alanı oldukça geniştir. Bu teknolojiler yaygın olarak kullanılmasının yanı sıra belli başlı temel sorunları bulunmaktadır. Yanık bu sorunların çözümü için yeni bir ölçüm cihazı geliştirmiştir.

Altekin (2021) yaptığı çalışmada makine öğrenmesi ve görüntü işleme tekniklerini kullanarak insan yüzündeki duyguları tespit etmek konularını incelemiştir. 981 adet duygudan oluşan CK+ veri setini kullanan Altekin, öznitelik çıkarıcı olarak HOG ve LBP (Local Binary Pattern ) öznitelik vektörlerini kullanmıştır. İnsan yüzündeki duyguların tespiti için burun, ağız, kaş bölgelerindeki duyguya göre değişimin tespit edilerek öznitelik vektörlerinde işlenmesi amaçlanmıştır. Kullandığı öznitelik vektörlerini Destek Vektör Makinaları, K-En Yakın Komşu Algoritması Lojistik Regresyon gibi yöntemlerle sınıflandırarak duygu tespitinde kullanılan yöntemlerin performansını karşılaştırmıştır.

Yılmaz (2022) bilgisayarlı görüde çizgi tespiti üzerine çalışmalar yapmıştır. Çizgi tespitinde geliştirilen algoritmaların yetersiz olduğunu vurgulamıştır. Literatürde yer alan algoritmaların hız, ve verimlilik açısından yeterince başarılı olmadığını belirtmiştir. HT, LSD ve EDlines algoritmalarına alternatif, hız performans olarak daha başarılı sonuçlar veren bir çizgi tespit algoritması geliştirmiştir.

Shakir (2022) görüntü işleme uygulamalrında, litreratürde yaygın olarak öne çıkan sorunlardan olan gürültü giderme konusunda çalışmalar yapmıştır. Yaptığı çalışmada gürültü giderme konusuna katkı sağlamak amacıyla toplam değişim fonksiyonunu düzgünleştiren bir teknik geliştirmiştir. Geliştirdiği tekniğin geçerliliğini ölçmek için bazı görüntülerde çalışmalar yapmıştır.

Emir (2021) çalışmasında oküler göz rahatsızlıklarının yapay zeka ile sınıflandırılma probleminin çözümünü amaçlamıştır. Retina taramasında kullanılan fundus görüntüleme ile oftalmolojik hastalıkların erken teşhis ile tedavisini sağlamaktadır. Teşhis yöntemlerinin zaman alması tedaviyi geciktirebilir. Bu durumda geç kalınan tedavi hastalığın ilerlemesine sebep olabilir. Günümüzde iş yükünü azaltmak için hastalıkların

otomatik ve hızlı tespiti önemlidir. Fundus görüntüleme ve derin öğrenme mimarilerin birleştirilerek kullanılması hastalıkların hızlı tespit edilmesinde önemli rol oynamaktadır. İnsan retinasına ait çeşitli fundus görüntüleri, barındırdığı hastalıklara ait belirtileri sınıflandırılarak çeşitli anahtar kelimelerle modellerin eğitimi yapılmıştır. Oluşturulan sınıflandırmalardan derin öğrenme modelleri geliştirilmiştir. Geliştirilen modeller fundus test kümesi kullanılarak sınıflandırma performansları ölçülmüştür. Yüksek performans gösteren modeller tanı konma aşamasında uzmanlara destek olmada önemli rol oynaması ve gelecekte daha teknolojik sistemeler tasarlanabileceğini göstermektedir. Hastalıklara ait görüntülerin sınıflandırmasında Inceptionv3, VGG16 ve ResNet50 CNN mimarisi oluşturulmuştur. Modellerin hastalık sınıflandırma performansı öznitelik çıkarımı elde edilerek değerlendirilmiştir.

Tüfekçi (2019) yaptığı çalışmada CNN içerisinde bulunan öğrenme yöntemlerini incelemiştir. İncelediği yöntemlerin görüntü işleme performanslarını test etmiş ve iyileştirmeye yönelik çalışmalar araştırılmıştır. Çalışmasında CNN mimarisindeki öğrenme yöntemlerinden gradyan inişi, adaptif momentum metodu, geri yayılım algoritması, aktif öğrenme ve aşırı öğrenme metodlarının görüntü işlemedeki perfonmansıyla ilgili çalışmalar yapılmıştır. Yaptığı çalışmanın CNN mimarisinin öğrenme metodlarının yüksek performans gösterdiğini ve bu metodlardan en yaygın kullanılanının geri yayılım algoritmalası oldunuğu sonucuna varmıştır.

Şeker (2020) yaptığı çalışmada mevcut yapay zeka uygulamalarının siber savunmaya entegresinin önemi ve zorluklarını ele almıştır. Teknolojinin hızlı ilerlemesi ve işlenen veri miktarının artması ile siber savunma alanında insan gücünün yetemediği durumlar meydana gelmiştir. Savunma sistemlerinin gelişen teknolojilere ve beraberinde oluşan tehditlere karşı, yeni teknolojilerin savunma sanayisine entegre edilmesi oldukça önemlidir. Günümüzde siber saldırılara karşı, sabit geleneksel savunma yazılımları ile korunmak zordur. Teknolojinin hızla gelişmesi ve veri miktarının büyüklüğü, insan gücünün yetemeyeceği durumlarda, saldırıların erken tespiti, önlenmesi ve zararın en aza indirilmesi için yapay zekanın savunma sistemlerine entegrasyonu oldukça önemlidir. Yapay zekanın öğrenebilme kabiliyeti sayesinde siber saldırılara karşı başarılı bir savunma sergilemek mümkündür. Çalışmasında güncel yapay zeka örnekleriyle siber savunmadaki bir çok sorunun yapay zeka ile çözülebileceği sonucuna varılmıştır.

Güzel ve Okatan (2022) yaptığı çalışmada dünyanın gelecekte önemli bir gıda problemi olduğunu belirtmiştir. Artan nüfus ve iklim değişikliği gıda yetersizliğinde önemli rol oynamaktadır. Geleneksel tarım yöntemleri gelecekte gıda ihtiyacı sorunun önüne geçememektedir. Yapay zeka, nesnelerin interneti ve robotik alanlarındaki gelişen teknolojiler, gelecekteki gıda sorununun çözümünde önemli bir rol oynayacağı düşünülmektedir. Yapay zekanın tarım alanında eğitilmesiyle, bitkilerdeki hastalık tespiti, toprak analizi ve sulama yöntemleri gibi çeşitli tespitlerin daha hızlı yapılması, gıda sorununun çözümü için oldukça önemlidir. Zamanında hasat, yabani otların tespiti, sulama yöntemi ve toprağın analizi gibi, bitki yetiştiriciliğinin önemli unsurları bulunmaktadır. Bilgisayarlı görü ve nesne tespiti teknolojileri, bu unsurlarda ortaya çıkacak sorunların, yapay zekanın, erken tespit, zamanında ve otomatik müdahalesi ile gıdanın daha verimli üretilmesi ve çiftçilerin işini kolaylaştırması açısından önemli rol oynamaktadır. Bu alanda yapay zekanın henüz yeni bir teknoloji olduğu bilinmektedir. Gelecekte teknolojinin gelişmesiyle yapay zekanın bu alanda önemli sorunların çözümüne yardımcı olacağı öngörülmektedir.

Hanbay ve Üzen (2017) yaptığı çalışmada nesne tanıma nesne sınıflandırma ve nesne tespitinde kullanılan güncel ve yaygın yöntemlerin kapsamlı bir araştırmasını yapmıştır. Yaptığı çalışmada kullanılan yöntemlerin güçlü ve zayıf yönlerini belirtmiştir. Kullanılan yöntemlerin güçlü ve zayıf yönlerini tablolayarak kıyaslamıştır. Karşılaştırılan yöntemler, nesne tespit yöntemleri, nesne tanımada kullanılan öznitelikler ve nesne takibi yöntemleridir. Karşılaştırma sonuçları temel çalışma prensipleri, avantajları ve dezavantajları şeklinde tablolar halinde verilmiştir. Nesne tespiti, nesne sınıflandırması ve nesne takibi alanlarında yapılan çalışmalar ve bu yöntemlerin performanslarından alınan sonuçlar derin öğrenme yöntemlerine olan güveni arttırmıştır.

Şimşek vd. (2019) yaptığı çalışmada doğada görüntü yakalamak için kullanılan fotokapanlardan elde edilen görüntüler üzerinde nesne tespiti ve öznitelik çıkarma işlemleri yapmıştır. Fotokapanlar doğadaki canlı hayatının takip eden ve görüntü elde eden cihazlardır. Bu cihazlar yerleştirildiği yerde bulunan canlıları, türleri ve canlı nüfusu hakkında bilgi sahibi olmak amacıyla kullanılmaktadır. Çalışmada, fotokapandan elde edilen edilen görüntüler girdi olarak kullanılarak nesne tespitinde kullanılan yöntemler ile nesne tespiti yapılmıştır. Nesne tespiti için kullanılan öznitelikler global ve yerel olmak üzere iki alanda incelenmiştir. Kays ve arkadaşlarından elde edilen veri seti

kullanılarak SIFT, SURF (Speed Up Robust Features), BRIEF Binary Robust İndependent Elementary Features) ve ORB yerel öznitelik çıkarma yöntemleri ile beraberinde kaba kuvvet ve k-en yakın komşu öznitelik eşleştirme yöntemleri kullanılmıştır. Nesne tespitinin yanı sıra kullanılan yöntemlerin performansları incelenmiştir. Kaba kuvvet eşleştirici ile öznitelik çıkarıcıların performansları incelendiğinde, farklı yüzdelik parametrelerde en yüksek performanslar sırasıyla 99.66 ile SIFT, 96.60 ile SURF, 95.23 ile ORB ve 91.52 ile BRIEF şeklinde sonuçlandırılmıştır. SIFT öznitelik çıkarıcı diğer özniteliklere göre en yüksek performansı sergilemiştir. k-en yakın komşu öznitelik eşleştirme yöntemine göre bütün parametrelerde SIFT öznitelik çıkarma yöntemi en yüksek performansı, BRIEF en düşük performansı vermiştir. SURF ve ORB öznitelik çıkarıcıları farklı parametrelerde birbirlerine yakın performans göstermiştir. Kıyaslanan dört parametrenin ikisinde SURF, ORB' a göre daha iyi performans göstermiştir. Çalışma sonucunda yerel öznitelik çıkarma yöntemleri performanslarının başarılı sonuçlar verdiği görülmüştür.

Karakuş ve Karabörk (2014) yaptığı çalışmada Spot uydusundan aynı konuma ait, farklı günlerde çekilen iki görüntünün ortak noktalarını tespit etme çalışması yapmışlardır. İki görüntünün ortak noktalarını bulma işlemi "Geometrik Kayıt" olarak adlandırılmaktadır. Geometrik kayıt uzaktan algılama alanlarında gerekli olan bir işlemdir. Çalışmada SURF (Speed Up Robust Features) algoritması kullanılarak görüntülerin ortak olan noktaları tespit edilmiştir. SURF algoritması kullanılan iki görüntüdeki dönüklük, gürültü önleme ve ölçek noktalarında başarılı sonuçlar vermektedir. SURF algoritması ile eşleşen tüm noktaların RANSAC (Random Sample Concensus) algoritması ile ayrıştırma işlemi yapılmıştır. RANSAC algoritması eşleşen noktalardaki yanlışlıkları elemek ve doğru olan noktalardan homografi matrisi kurmak için kullanılmıştır. SURF algoritması otomatik görüntü kaydı işleminde başarılı sonuç vermiştir.

Aydın ve Akar (2017) çalışmasında öznitelik çıkarma yöntemlerinden SURF, SIFT ve HOG yöntemlerini kullanarak yüz tanıma performanslarını ölçmüştür. Yüz tanıma sistemleri, bilgisayarlı görü konusunda yoğun çalışmalar yapılan bir alandır. Yüz tanıma sistemlerinde öznitelik çıkarımı ve sınıflandırma adımları kullanılmaktadır. Sınıflandırma yöntemi ve öznitelik çıkarma işlemleri başarılı bir yüz tanıma için yüksek performans göstermelidir. Öznitelik çıkarma yöntemleri global ve yerel olarak ikiye ayrılır. Çalışmada kullanılan ve ölçekleme ve afin dönüşüme karşı değişmeyen SURF ve

SIFT yerel öznitelikleri ve global özniteliklerden HOG özniteliği kullanılarak yüz tanıma performansı karşılaştırılması yapılmıştır. Yerel öznitelikler yüz tanıma alanında daha başarılı performans göstermiştir.

Kutuk (2012) yaptığı çalışmada göğüs dokusundaki yoğunlukların sınıflandırılması için bir yöntem önermiştir. Göğüs kanseri teşhisinde ve aşamalarında doku yoğunluğu önemli bir tanı belirtisidir. Yapılan çalışmalarda bazı doku yoğunluklarının göğüs kanseri belirtisi olduğu bilinmektedir. Bununla beraber bazı doku yoğunlukları kanser hücrelerini görmeyi zorlaştırmaktadır. Doku yoğunlukları kanserin erken teşhisinde önemli bir belirtidir. Çalışmada doku yoğunlukları yağlı doku, yağlı-bezel doku, yoğun-bezel doku olarak üç sınıfa ayrılmıştır. Bilgisayarlı görü yöntemleri kullanılarak görüntüdeki doku özelliklerine göre sınıflandırma çalışması yapılmıştır. MIAS görüntü veri tabanı kullanılarak dokuların sınıflandırılması sağlanmıştır. Veri tabanı içerisinde 332 doku görüntüsü bulunmaktadır. Bu veri tabanında her bir kişinin sağ ve sol göğüs dokusundan birer görüntü bulunmaktadır. Bilgisayarlı görü alanında kullanılan öznitelik çıkarma yöntemleri ile sınıflandırma metodları kullanılmıştır. SIFT öznitelik çıkarma yöntemi ile elde edilen veriler, Gaus karışım modeli GMM (Gaussian Mixture Model), SVM (Support Vector Machine) ve LVQ (Learning Vector Quantization) sınıflanıdırıcıları kullanılarak doğru sınıflandırma işlemi yapılmaya çalışılmıştır. Çalışmada sınıflandırma yöntemlerinin performansları incelenmiştir.

# 3. MATERYAL ve METOT

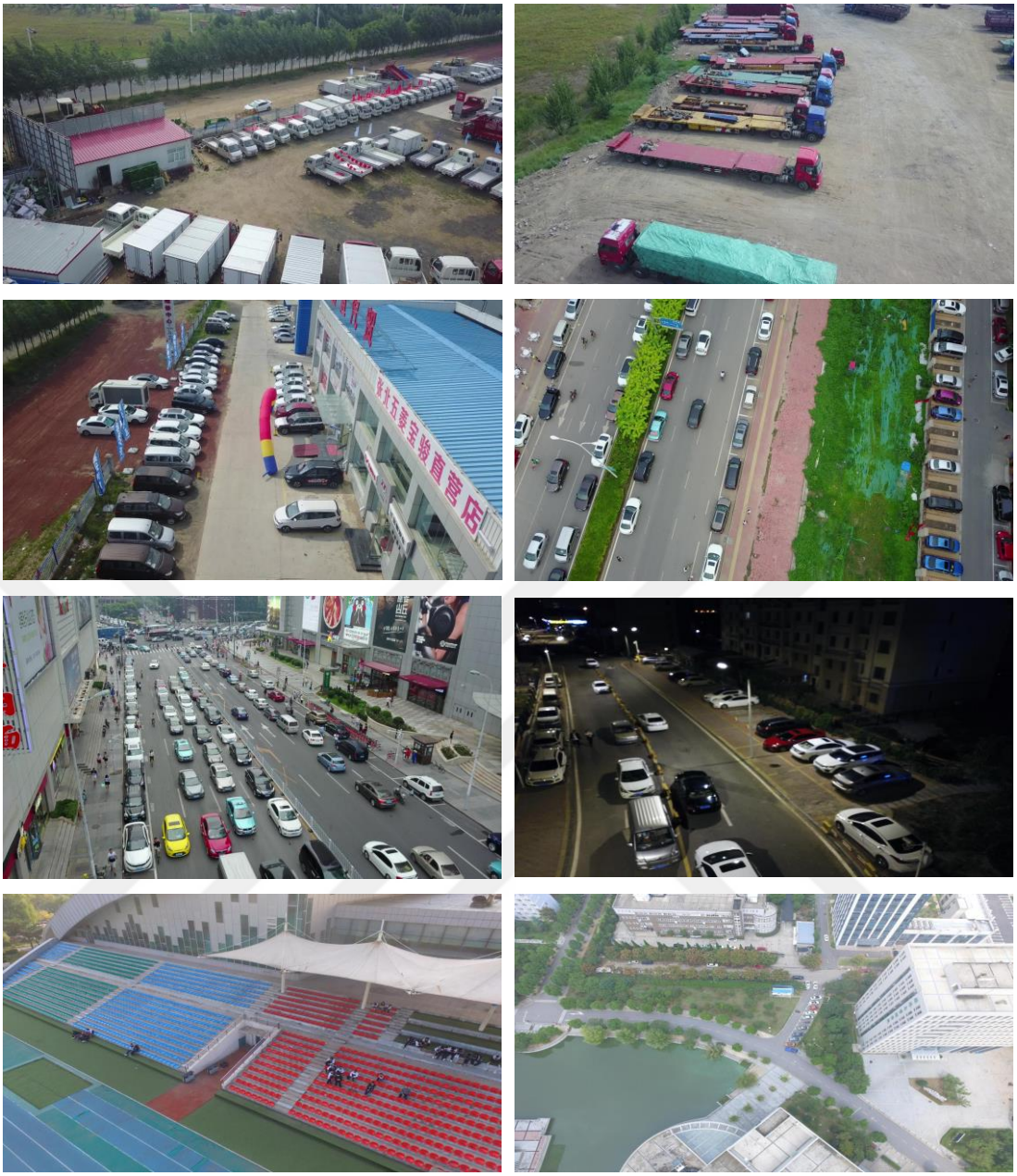
## 3.1 Veri Setleri

Bu çalışmada literatürde popüler olan VisDrone veri seti (Zhu vd. 2022) ve MNIST veri seti (LeCun vd. 1998) kullanılmıştır. VisDrone veri seti drone ile çekilmiş görüntülerden oluşurken, MNIST veri seti farklı el yazılarıyla yazılmış rakamlardan oluşmaktadır.

### 3.1.1 VisDrone Veri Seti

Bu çalışmada öznitelik çıkarımı için kullanılacak verisetlerinden biri VisDrone veriseti olarak belirlenmiştir. 1610 eğitim, 1580 test görüntüsü olmak üzere toplamda 3190, 1920*1080 piksel görüntüden oluşmaktadır. İnsanlar, yayalar, bisiklet, tenteli üç tekerlekli bisiklet, üç tekerlekli bisiklet, motosiklet, otobüs, araba, kamyon, panelvan, olmak üzere 10 sınıf ile beraber, tanımsız ve diğer olmak üzere 2 adet daha sınıf barındıran bu veriseti çeşitli drone çekimlerinden oluşmaktadır. Drone çekimlerinde nesne tespiti, görüntü açısı yukarıdan aşağıya oduğu için geleneksel yöntemlere kıyasla oldukça zordur (Cao vd. 2021).

**Resim 3. 1** VisDrone veri setine ait görseller



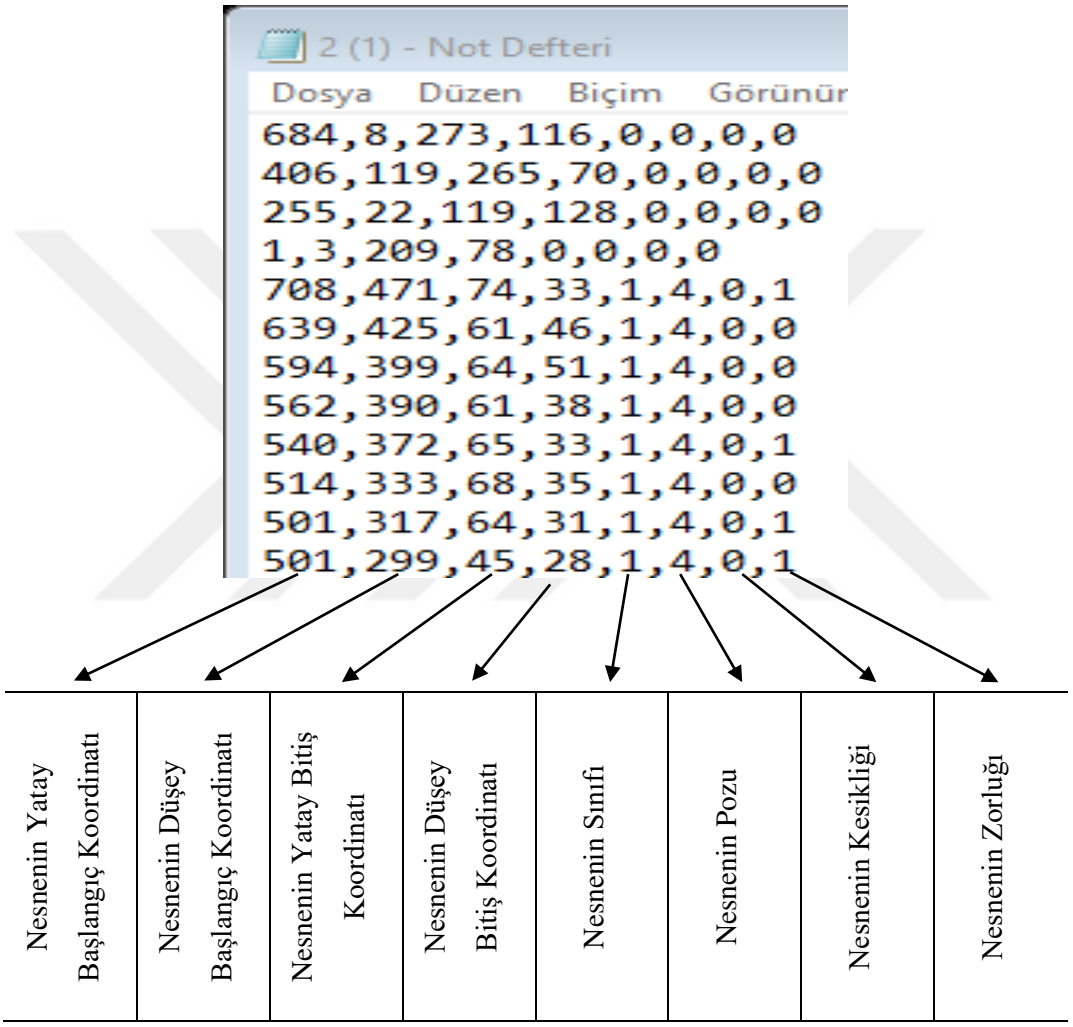
**Resim 3.1 (Devam)** VisDrone veri setine ait görseller

Yukarıdaki görseller VisDrone veri setine ait örnek görüntülerdir. Drone ile çekilen görsellerde çeşitli sınıflara ait görüntüler bulunmaktadır. Bu çalışmada 10 sınıftan, bilgisayarla görü alanında sıklıkla kullanılan 2 sınıf, insan ve araba sınıfları ele alınmıştır.

### 3.1.1.1 VisDrone Veri Seti İşlenmesi

Verisetinden elde edilen görüntülerin her biri python'da yazılan programla işlenmiştir. Program kodları Ek 1'de verilmiştir. Her bir görselin kendine ait bir bilgi "ek açıklamalar (annotations)" dosyası bulunmaktadır. Bu dosya metin belgesi olarak veri setinde yer

almaktadır. Verisetinde “ek açıklamalar (annotations)” klasöründe nesnelerin görüntü üzerindeki konumları ve özellikleri belirtilmiştir. Dosya içeriği, ait olduğu görüntünün içerisindeki nesne kadar satır ve 8 sütun bulunmaktadır. Her bir sütunun ve değerin bir anlamı bulunmaktadır. Her bir nesnenin görüntü içerisindeki başlangıç ve bitiş koordinatları sınıfını, pozunu, güvenilirliğini gösteren değerler bulunmaktadır. Bu değerler “,” ile ayrılmıştır. Resim 3.2’ de değerler örnek dosya yer almaktadır.




**Resim 3. 2** Ek açıklamalar (annotations) klasöründeki metin belgesi (.txt) örneği

Metin belgesindeki verilerin anlamlı bir bütün oluşturması ve görüntü içindeki nesnelerin sınıflandırılıp kırpılması için Ek açıklamalar (annotations) dosyalarının işlenmesi gerekmektedir. Bu dosyadaki veriler Python dilinde örnek yazılım yardımıyla işlenmiştir.

### 3.1.1.2 Ek Açıklamar (Annotations) Dosyalarının İşlenmesi

Metin dosyasındaki verileri anlamlandırmak için her sütun içerisindeki değer gruplandırılmıştır. İlk sütun görüntüdeki nesnenin yatay başlangıç koordinatını (xmin), ikinci sütun düşey başlangıç koordinatını (ymin), üçüncü sütun nesnenin yatay bitiş koordinatını (xmax), dördüncü sütun nesnenin düşey bitiş koordinatını (ymax) vermektedir. İlk dört sütun nesnenin konumu verirken, diğer dört sütun nesnenin sınıfını, pozunu, doğruluk değerini vermektedir. Her nesne için bir satır bulunmaktadır. Metin dosyasındaki verilerin anlamlandırılma süreci sütunların amacına uygun kullanılması işlemidir. Bunu yapabilmek için python dilinde kodlardan yararlanılmıştır. Başlangıçta dosyalarımızın girdi olarak tanımlanması gerekmektedir. Yazılımın dosyaları çekmesi ve işlemler yapıldıktan sonra çıktı olarak kaydetmesi için Resim 3.3' deki kodlardan yararlanılmıştır.

```
input_img_folder = 'images'
input_ann_folder = 'annotations-txt'
output_ann_folder = 'VisDrone2019-DET-train/annotations_new'
output_img_folder = 'VisDrone2019-DET-train/images_new'

os.makedirs(output_img_folder, exist_ok=True)
os.makedirs(output_ann_folder, exist_ok=True)


image_list = os.listdir(input_img_folder)
annotation_list = os.listdir(input_ann_folder)
```

**Resim 3. 3** Dosyaları girdi olarak çekme ve çıktının kayıt edileceği uzantı kodları

Girdi ve çıktı kodları tamamlandıktan sonra sınıflandırma işlemi gerçekleşmektedir. VisDrone veri setinde 10 sınıf ile beraber tanımsız ve diğer sınıfları yer almaktadır. Sınıflandırma işlemi için bu sınıflar kodlanarak yazılıma eklenmiştir. Resim 3.4' de sınıfların kodlanması yapılmıştır.

```
label_dict = {
    "0" : "Ignore",
    "1" : "Pedestrian",
    "2" : "People",
    "3" : "Bicycle",
    "4" : "Car",
    "5" : "Van",
    "6" : "Truck",
    "7" : "Tricycle",
    "8" : "Awning-tricycle",
    "9" : "Bus",
    "10" : "Motor",
    "11" : "Others"

}
```

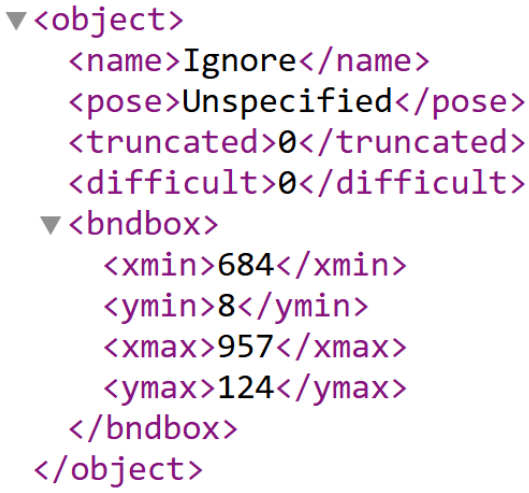
**Resim 3. 4** Sınıflandırma işlemi

Sınıflandırma işlemi bittikten sonra çıktı işleminin taslağı oluşturulmaktadır. Sütunlar ve satırlardaki değerlerin yerleştirilmesi ve anlamlı şekilde okunabilmesi için bir taslak oluşturulmuştur. Taslak her nesnede kullanılacağı için fonksiyon içerisine tanımlanmıştır. Bu taslağın python içerisinde kodlanmış şekli Resim 3.5' de yer almaktadır.

```
def object_string(label, bbox):
    req_str = '''
    <object>
        <name>{}</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>{}</xmin>
            <ymin>{}</ymin>
            <xmax>{}</xmax>
            <ymax>{}</ymax>
        </bndbox>
    </object>
    '''.format(label, bbox[0], bbox[1], bbox[2], bbox[3])
    return req_str
```

**Resim 3. 5** Fonksiyon içerisinde sütunların taslak oluşturulması

Oluşturduğumuz taslağın çıktısı Resim 3.6' da gösterilmiştir.




**Resim 3. 6** Sütunların program çıktısındaki görüntüsü

Fonksiyon oluşturulduktan sonra metin dosyası içerisindeki sütunlarda "os" modülünü kullanarak gerekli değişiklikler yapılmaktadır. Resim ve metin dosyası yazılım içine çekilir ve çıktının dosya özelliklerini ve hangi özelliklere sahip olarak kaydedileceği belirtilmiştir. Resim 3.7' de kodları verilmiştir. Burada amaç metin dosyasındaki sütunları anlamlı veriler halinde genişletilebilir işaretleme dili ".xml" dosyasına çevirmektir.

```
for annotation in annotation_list:

    annotation_path = os.path.join(os.getcwd(), input_ann_folder, annotation)
    xml_annotation = annotation.split('.txt')[0] + '.xml'
    xml_path = os.path.join(os.getcwd(), output_ann_folder, xml_annotation)
    img_file = annotation.split('.txt')[0] + '.jpg'
    img_path = os.path.join(os.getcwd(), input_img_folder, img_file)
    output_img_path = os.path.join(os.getcwd(), output_img_folder, img_file)
    img = cv2.imread(img_path)
```

**Resim 3. 7** Metin belgesi ve img dosyalarının işlenmesi

Elimizdeki verilerin fazla olmasından dolayı, dosyaların daha anlaşılır olması için, işleme alınan her resim dosyasının adı, bilgisayar üzerindeki konumu, uzantısı, ve çözünürlüğü ile ilgili özellikleri tanımlayan, genişletilebilir işaretleme dili(.xml) dosyamızın içerisine tanımlanmıştır. Her bir resim dosyası için Resim 3.8 'deki gibi bir komut satırı atanmıştır.

```
    annotation_string_init = '''
<annotation>
    <folder>annotations</folder>
    <filename>{}</filename>
    <path>{}</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>{}</width>
        <height>{}</height>
        <depth>{}</depth>
    </size>
    <segmented>0</segmented>'''.format(img_file, img_path, img.shape[0], img.shape[1], img.shape[2])
```

**Resim 3. 8** Resim dosyalarının özelliklerini gösteren komutlar

İlgili kod satırının çıktısı Resim 3.9' deki gibidir.

```
▼<annotation>
   <folder>annotations</folder>
   <filename>2 (1).jpg</filename>
   <path>C:\Users\Kaan Yasin KOCAMAN\Downloads\VisDrone-dataset-python-toolkit-master\VisDrone-
   dataset-python-toolkit-master\VisDrone2019-DET-train/images\2 (1).jpg</path>
  ▼<source>
     <database>Unknown</database>
   </source>
  ▼<size>
     <width>540</width>
     <height>960</height>
     <depth>3</depth>
   </size>
   <segmented>0</segmented>
```

**Resim 3. 9** Resim 3.6' da bulunan kodların çıktısı

Metin belgesi dosyalarımızın ve çıktı alacağımız genişletilebilir işaretleme dili(xml) dosyalarımızın gerekli düzenlemeleri yapılmıştır. Bir sonraki işlemimizde metin belgesi içerisinde yer alan sütun ve satırların oluşturduğumuz taslaklara atama işlemi gerçekleştirilmiştir. Burada her satır için işlem yapılacağı için tekrar döngü içerisine alınmaktadır. Klasördeki her dosya döngüye girmiştir. Dosya içerisindeki veriler de bu döngü içerisinde tekrar döngüye alınmıştır. Oluşturduğumuz fonksiyon komutu burada devreye girmektedir. Resim 3.10' de verilerin işlendiği komutlar verilmiştir.

```
file = open(annotation_path, 'r')
lines = file.readlines()
for line in lines:

    new_line = line.strip('\n').split(',')
    new_coords_min = (int(new_line[0]), int(new_line[1]))
    new_coords_max = (int(new_line[0])+int(new_line[2]), int(new_line[1])+int(new_line[3]))
    bbox = (int(new_line[0]), int(new_line[1]), int(new_line[0])+int(new_line[2]), int(new_line[1])+int(new_line[3]))
    label = label_dict.get(new_line[5])
    req_str = object_string(label, bbox)
    annotation_string_init = annotation_string_init + req_str
    cv2.rectangle(img, new_coords_min, new_coords_max, color, thickness)
```

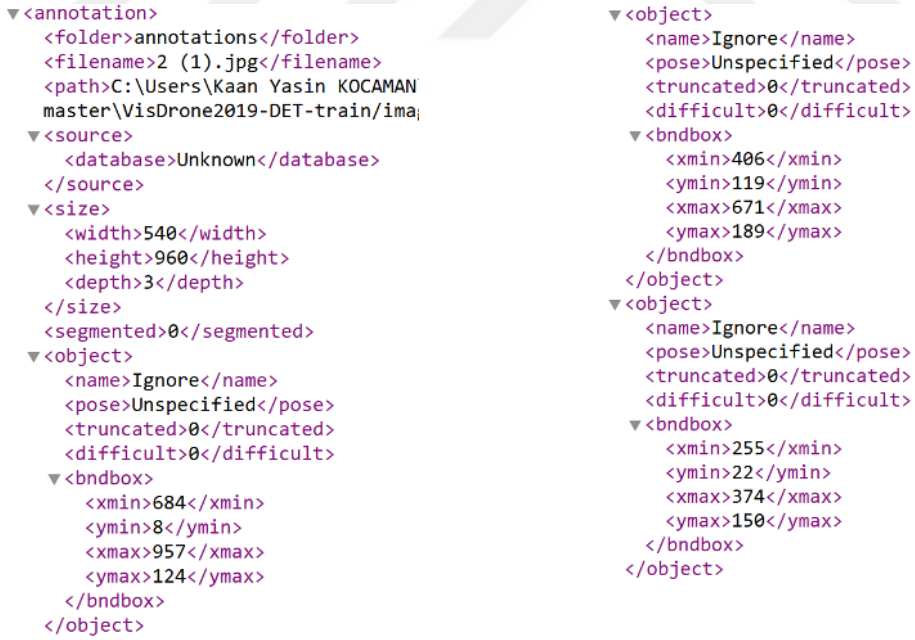
**Resim 3. 10** Metin belgesi verilerinin işlendiği komut dizisi

Metin belgesi içerisinde yer alan verilerin işlenme süreci bittiğinde yeni oluşan dosyalarımızın kaydetme işlemine geçilmiştir. Dosyalarımızın genişletilebilir işaretleme dili(xml) uzantılı kaydolması gerekmektedir. Resim 3.11' de genişletilebilir işaretleme dili(.xml) dosyası olarak kaydetme komutları verilmiştir.

```
annotation_string_final = annotation_string_init + '</annotation>'
f = open(xml_path, 'w')
f.write(annotation_string_final)
f.close()
count += 1
print('[INFO] Completed {} image(s) and annotation(s) pair'.format(count))
```

**Resim 3. 11** Genişletilebilir işaretleme dili (.xml) uzantılı kayıt dizisi

Metin belgesi(.txt) dosyalarımız içerisindeki veriler işlenerek VisDrone veri setinde kırpma işlemi yapmak için kullanıma uygun yapı oluşturulmuştur. Oluşturulan yapının genişletilebilir işaretleme dili(xml) olarak kaydedilme işlemi yapılmıştır. Resim 3.12' de ek açıklamalar (annotations) klasöründeki metin belgesi dosyalarımızın çıktı alındıktan sonraki görüntüsü yer almaktadır.



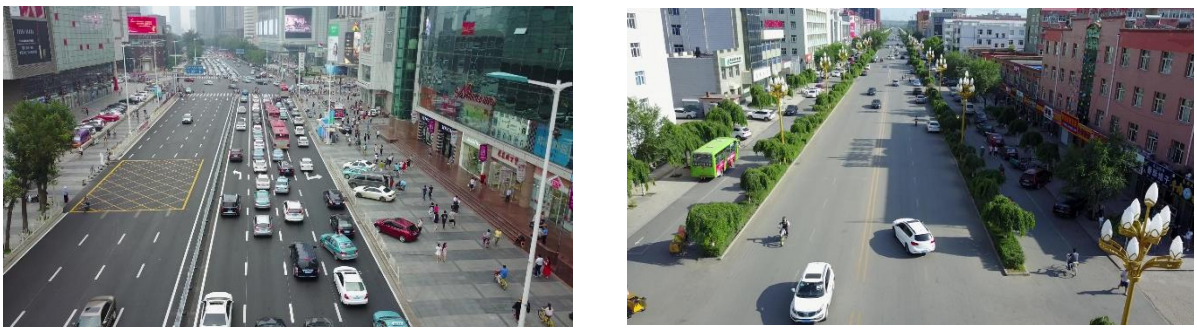

**Resim 3. 12** Ek açıklamalar (annotations) metin belgesi (.txt) dosyasının genişletilebilir işaretlemedili(.xml) olarak çıktısı

```
<object>
  <name>Ignore</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>1</xmin>
    <ymin>3</ymin>
    <xmax>210</xmax>
    <ymax>81</ymax>
  </bndbox>
</object>
<object>
  <name>Car</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>708</xmin>
    <ymin>471</ymin>
    <xmax>782</xmax>
    <ymax>504</ymax>
  </bndbox>
</object>
```

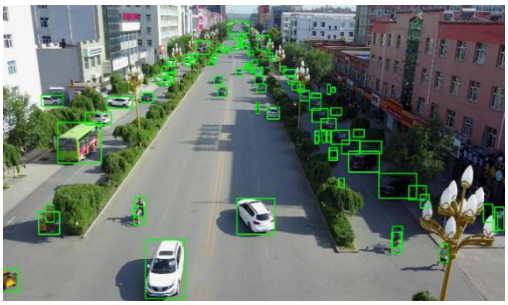
```
<object>
  <name>Car</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>639</xmin>
    <ymin>425</ymin>
    <xmax>700</xmax>
    <ymax>471</ymax>
  </bndbox>
</object>
<object>
  <name>Car</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>594</xmin>
    <ymin>399</ymin>
    <xmax>658</xmax>
    <ymax>450</ymax>
  </bndbox>
</object>
```

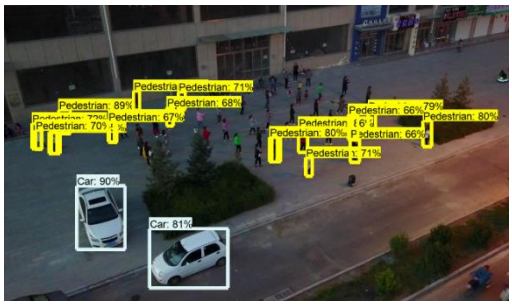

**Resim 3.12(Devam)** Ek açıklamalar (annotations) metin belgesi (.txt) dosyasının genişletilebilir işaretlemedili(.xml) olarak çıktısı

Oluşturulan dosyalar VisDrone veri setinde kullanıma uygun hale gelmiştir. Resim dosyaları içerisindeki nesnelerin tespiti sınıfı ve güvenilirliğini değerlendirmek için kullanılabilir. Resim 3.13' de VisDrone veri setine ait örnek kullanımlar gösterilmiştir. Örnekte veri setinden bazı ham görseller ve nesne tespiti yapılmış görseller bulunmaktadır.



**Resim 3. 13** VisDrone veri seti ham görseller ve ek açıklamalar (annotations) klasöründeki verilerle görseldeki tespit edilmiş nesneler ile sınıflandırılması






**Resim 3.13(Devam)** VisDrone veri seti ham görseller ve ek açıklamalar (annotations) klasöründeki verilerle görseldeki tespit edilmiş nesneler ile sınıflandırılması

Yukarıda ki görselde VisDrone veri setinden nesne tespiti yapılmış örnek görseller verilmiştir. Bu çalışmada VisDrone veri setinde, görsel içerisindeki nesnelerin tespit işlemi yapılmıştır. Bunun yanı sıra bu tespit işleminin, bu çalışmada kullanılabilmesi için nesnelerin sınıflarına göre görselden kırpılarak klasörlendirilme işlemi gerçekleşmiştir. Görsel içindeki belirli sınıflara ait nesneler bu çalışmada veri setini oluşturacaktır. Genişletilebilir işaretleme dili(xml) dosyalarımız veri setindeki nesneleri tespit etmek, kırpmak ve resim dosyası uzantısı ".jpg" olarak kaydetme işlemi için kullanılmıştır. Ek 1' de yazılan kodların tamamı yer almaktadır.

#### 3.1.1.3 Veri Setindeki Resim Dosyalarını Genişletilebilir İşaretleme Dili(.Xml) Dosyalarına Göre Kırpma İşlemi

Çıktı işlemi sonrasında elde edilen genişletilebilir işaretleme dili dosyası VisDrone veri setindeki görsellerle entegre edilerek koordinatlarına göre kırpma ve sınıflandırma işlemi gerçekleştirilmiştir. Her bir görüntü kendine ait xml uzantılı dosyasındaki verilere göre kırpılmıştır. Burada amaç anlamlı bir hale getirilen metin belgesi dosyalarının her satırını nesneler için, her sütununu da nesnelerin tespiti ve sınıflandırılması için kullanmaktır. Bu işlemi yapmak için öncelikle veri setinde yer alan resim dosyalarını girdi olarak yazılan yazılıma tanımlamak, tespit ve kırpma işlemi sonrası elde edilen çıktıyı yine resim dosyası (.jpg) olarak kaydetmek için dosya yolu belirleme işlemlerini tamamlamaktır. Tanımlamalar, genişletilebilir işaretleme dili dosyasındaki etiketleri python programlama dilinin anlayabileceği hale getirmek için değerlere atanması işlemidir. Genişletilebilir

işaretleme dili dosyalarındaki etiketler python diline tanımlanarak etiketler içerisindeki değerler python programlama dilinin anlayacağı değerlere dönüşmektedir. Bu işlemler için öncelikle yazılan yazılıma bilgisayarda girdi dosyalarının bulunduğu klasörün yolu ve yapılan işlemlerden sonra çıktının kaydolacağı klasörün yolunu tanımlamak gerekmektedir. Örnek kodlar Resim 3.14'te verilmiştir.

```
original_file = 'images/' #you images directory
dst = 'results/'
```

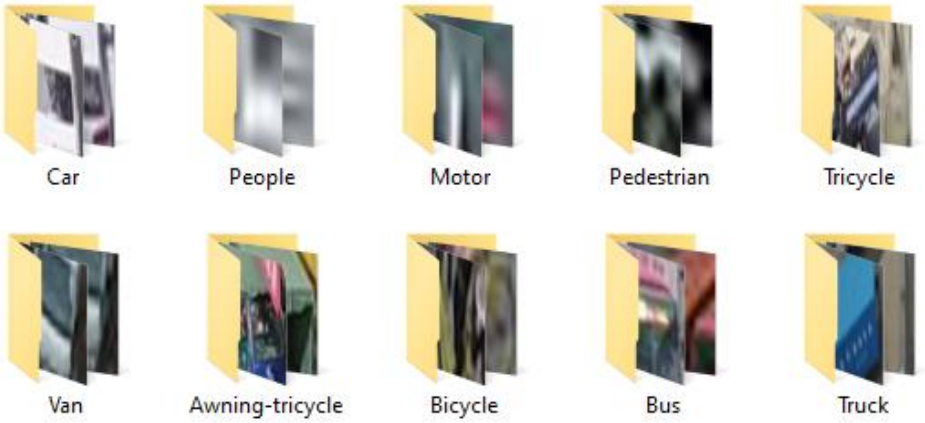
**Resim 3. 14** Veri seti dosyamızın yolu ve çıktı işlemi için tanımladığımız results klasörü yolu

Tanımlama işlemi sonrasında yapılacak olan işlemlerin kontrol edilmesi gerekmektedir. Kontrol için gereken kod satırı eklenmiştir. Burada yapılan işlem sınıflandırma veri setinde yer alan 10 sınıf ve 2 diğer sınıfın klasörünü kontrol etmektir. "check_folder_exists" komutu çıktı olarak elde edilen resmin ait olduğu sınıfı kontrol etmektedir. Eğer klasör mevcut değilse klasör oluşturmak için kullanılmaktadır. Bu işlem her seferinde tekrar edeceği için fonksiyon olarak yazılmıştır. Sonrasında bu fonksiyon döngü içerisine alınacaktır. Resim 3.15' da sınıf klasörlerini kontrol etmek için yazılan kod satırı verilmiştir.

```
def check_folder_exists(path):
    if not os.path.exists(path):
        try:
            os.makedirs(path)
            print ('create ' + path)
        except OSError as e:
            if e.errno != errno.EEXIST:
                raise
```

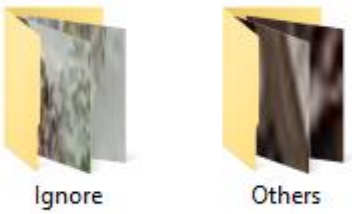
**Resim 3. 15** Sınıf klasörlerini kontrol etmek için yazılan kod satırı

Kontrol işlemi sonrasında var olan klasörlerle ilgili herhangi bir işlem yapılmayacaktır. Klasörün var olmadığı durumlarda kod satırı bu klasörü oluşturmaktadır. Resim 3.16'de veri setinde var olan sınıflar klasörlere ayrılmış şekilde verilmiştir.




**Resim 3. 16** Veri setinde yer alan 10 sınıf

Resim 3.17' de sınıflandırmaya dahil olmayan "tanımsız" ve "diğer sınıfları" yer almaktadır.




**Resim 3. 17** Veri setinde yer alan 2 ayrı "tanımsız" ve "diğer" sınıfı

Klasörlerin oluşturulma işlemi gerçekleşmiştir. Veri setindeki resimlerin içerisinde yer alan nesneler, sınıflandırma işleminden sonra oluşturulan 12 sınıftan ait olduğu sınıfa gönderilecektir. Bu sınıflandırma işlemi için, oluşturulan genişletilebilir işaretleme dili dosyalarının kullanılması gerekmektedir. Her resim dosyası için, ona ait sınıflandırma verileri barındıran bir dosya bulunmaktadır. Bu dosyaların her biri için işlem yapılacaktır. Bu nedenle "for" döngüsü içerisine sırasıyla genişletilebilir işaretleme dili dosyaları işleme alınmıştır. Dosyaların içerisinde bulunan her etiket sırayla işlenmektedir. Resim 3.18'de örnek genişletilebilir işaretleme dosyası içerisinde yer alan etiketler bulunmaktadır.

```
▼<annotation>
   <folder>annotations</folder>
   <filename>2 (1).jpg</filename>
   <path>C:\Users\Kaan Yasin KOCAMAN\Downloads\VisDrone-dataset-python-toolkit-master\VisDrone-dataset-python-toolkit-
   master\VisDrone2019-DET-train/images\2 (1).jpg</path>
```

**Resim 3. 18** Genişletilebilir işaretleme dosyası içerisinde yer alan etiketler

Örnek etiketlerin işlenmesi için gerekli kod satırları Resim 3.19' da verilmiştir. Ek açıklamalar "annotation" klasörü içerisinde yer alan genişletilebilir işaretleme dili dosyalarının, bilgisayar içerisinde yer alan veri yolu belirtilerek dosyalara ulaşılmıştır.

```
seed_arr = []
for xml_file in glob.glob('annotation-xml/*.xml'): #your xml directory
    root = ET.parse(xml_file).getroot()
    filename = root.find('filename').text
```

**Resim 3. 19** Genişletilebilir işaretleme dosyası içerisinde yer alan etiketlerin işlenmesi için kod satırı

Genişletilebilir işaretleme dili dosyaları içerisindeki her etiketin anlamları bulunmaktadır. Dosya içerisinde her resme ait bilgiler ve sonrasında sınıflandırma için yer alan bilgiler bulunmaktadır. Resme ait bilgileri barındıran etiketler Resim 3.20' de verilmiştir.

```
▼<size>
    <width>540</width>
    <height>960</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
```

**Resim 3. 20** Resme ait bilgiler barındıran etiketler

Etiketlerde resmin genişlik ve yükseklik bilgilerinin yer aldığı görülmektedir. Bu etiketlerin tanımlama işlemini Resim 3.21' de yer almaktadır. İşleme alınan her genişletilebilir işaretleme dili dosyası için bu etiketleme işlemi yapılacağından oluşturulan "for" döngüsü içerisine tekrar bir "for" döngüsü oluşturularak resimlerin yükseklik ve genişlik tanımlamaları yapılmıştır. Sınıflandırma işlemi için her verinin kullanılmasına gerek yoktur. Bundan dolayı bazı etiketler tanımlanmamıştır.

```
for type_tag in root.findall('size'):
    width = type_tag.find('width').text
    height = type_tag.find('height').text
```

**Resim 3. 21** Resimlere ait bilgilerin tanımlanması

Her resim için 1 adet genişletilebilir işaretleme dili dosyası bulunmaktadır. Bu dosyanın içerisine bakıldığında, resim içerisinde yer alan nesnelere ait bilgiler verilmektedir. Bu

bilgiler etiketlerden oluşmaktadır. Etiketler resimlere ait bilgiler barındırdığı gibi, içerisinde bulunan nesnelere ait etiketler de bulunmaktadır. Resim 3.22' de nesnelere ait etiketler gösterilmiştir


**Resim 3. 22** Nesnelerin özelliklerine ait etiketler

Etiketlerde resim içinde bulunan nesnelere ait bilgiler görülmektedir. Bu etiketin tanımlanması için Resim 3.23' te bulunan kod satırlarından faydalanılmıştır. Tanımlamalarda nesnenin sınıfı, yatay başlangıç koordinatı, düşey başlangıç koordinatı, yatay bitiş koordinatı ve düşey bitiş koordinatları yer almaktadır. İşleme alınan her nesne için bu etiketleme işlemi yapılacağından, oluşturulan ilk "for" döngüsü içerisine, içindeki "for döngüsünden bağımsız, tekrar bir "for" döngüsü oluşturulmuştur. Döngü içerisine nesnelere ait sınıf ve konumlarına ilişkin tanımlamaları yapılmıştır.

```
    for type_tag in root.findall('object'):
        class_name = type_tag.find('name').text
        xmin = type_tag.find('bndbox/xmin').text
        ymin = type_tag.find('bndbox/ymin').text
        xmax = type_tag.find('bndbox/xmax').text
        ymax = type_tag.find('bndbox/ymax').text
        all_list = [filename, width,height,class_name,xmin, ymin, xmax,ymax]

        seed_arr.append(all_list)

seed_arr.sort()
```

**Resim 3. 23** Nesnelere ait etiketleri tanımlayan kod satırı

Tanımlamalar yapılarak dosya içerisindeki etiketler, python dilinin anlayabileceği değerlere dönüştürülmüştür. Tanımlanan değerlerin fotoğraf üzerinde işlenebilmesi için

etiketlerin içindeki değerler satırlar halinde numaralandırılarak dizin haline getirilmiştir. Her nesne için bu işlem tekrar edeceğinden öncekilerinden bağımsız şekilde bir "for" döngüsü içerisine alınmıştır. Resim 3.24' te bu işlemin kodları verilmiştir.

```
for index, line in enumerate(seed_arr):
    filename = line[0]
    width = line[1]
    height = line[2]
    class_name = line[3]
    xmin = line[4]
    ymin = line[5]
    xmax = line[6]
    ymax = line[7]
```

**Resim 3. 24** Etiketler içindeki değerlerin satırlar halinde numaralandırılarak dizin haline getirilmesi

Dizin haline getirilen değerler resimler üzerinde kırpma işlemi için uygun hale gelmiştir. Kırpma işlemi için resimler klasöründen resim dosyaları sırayla işleme alınmaktadır. Nesneler, genişletilmiş işaretleme dosyası içerisinde yer alan değerlere göre, resim içerisinde konumları belirlenerek sınıflandırma ve kırpma işlemi gerçekleştirilmiştir. Kırpılan nesneler, 64*128 piksel boyutunda resim dosyası (.jpg) olarak ait olduğu sınıfın klasörüne kaydedilmiştir. Resim 3.25'da kırpma ve kaydetme işlemlerinin kod satırları verilmiştir.

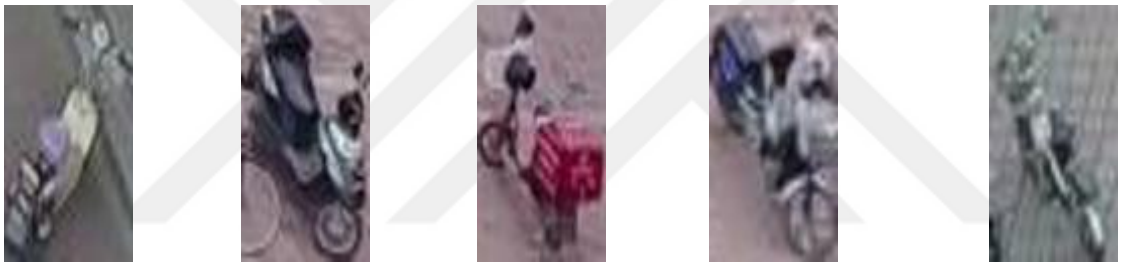
```
load_img_path = os.path.join(original_file, filename)

save_class_path = os.path.join(dst, class_name)
check_folder_exists(save_class_path)
save_img_path = os.path.join(save_class_path, str(index)+'_'+filename)

img = Image.open(load_img_path)
crop_img = img.crop((int(xmin) ,int(ymin) ,int(xmax) ,int(ymax)))
newsize = (64, 128)
im1 = crop_img.resize(newsize)
im1.save(save_img_path, 'JPEG')
print('save ' + save_img_path)
```

**Resim 3. 25** Nesnelerin, resimden kırpma ve kaydedilme işlemi

Çıktı işlemi olarak görsellerdeki nesneler, sınıflandırma yöntemi ile ait olduğu sınıflara göre klasörlere ayrıştırılmıştır. Görüntüler içerisinde tespit edilen nesneler sınıflara ayrılmıştır. Resim 3.26' da sınıf landırma örnekleri verilmiştir.

Arabalar sınıfı

İnsanlar sınıfı



Motosiklet sınıfı

Yaya sınıfı

**Resim 3. 26** VisDrone veri setinde bulunan sınıflara ait görseller

3 tekerlekli bisiklet sınıfı (açık kabin)

Panelvan sınıfı

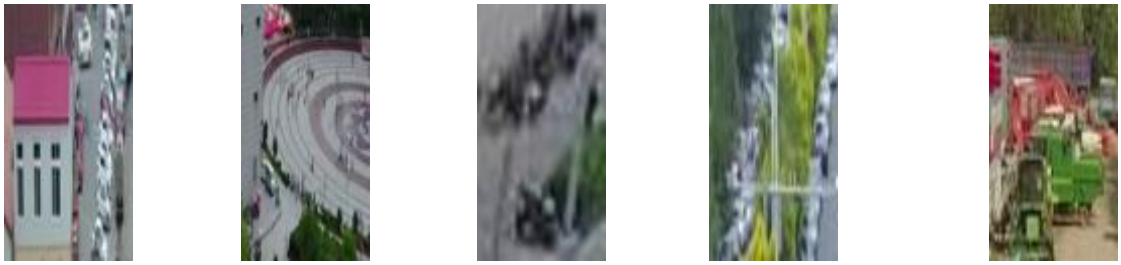
3 tekerlekli bisiklet sınıfı (kapalı kabin)

Bisiklet sınıfı

Otobüs sınıfı

**Resim 3. 26 (Devam)** VisDrone veri setinde bulunan sınıflara ait görseller

Kamyon sınıfı



Tanımsız sınıfı

Diğer sınıfı

**Resim 3. 26 (Devam)** VisDrone veri setinde bulunan sınıflara ait görseller

Sınıflandırma sonucu kullanılacak olan sınıflar seçilip vektör çıkarma işlemi için hazır hale getirilmiştir. Yapılan işlemler için yazılan kodların bütünü Ek 2' de yer almaktadır.

### 3.1.2 MNIST Veri Seti

MNIST veri seti farklı el yazılarıyla yazılmış kyaslama standartlı siyah-beyaz rakamlardan oluşan bir veri setidir. 60000 eğitim ve 10000 test görüntüsünden oluşan MNIST veri setinde her bir görüntü 28*28 piksellerden oluşmaktadır (Toğaçar 2021). Resim 3.27' de MNIST veri setine ait örnek görüntüler yer almaktadır.

**Resim 3. 27** MNIST veri seti örnek görseller

### 3.2 Nesne Algılama Yöntemleri

Nesne tespiti bazı süreçlerden geçerek sonuca ulaşır. Veri girişi, ön işleme, nesne tespiti, öznitelik çıkarımı gibi farkı aşamalar altında çeşitli algoritmalar yardımıyla tamamlanır (Daş vd. 2019).

### 3.3 Veri Girişi

Veri girişi aşamasında görüntü, gereksiz ayrıntılardan ve gürültülerden ayrıştırılmak üzere girdi olarak sisteme yüklenir. Doğa yürüyüşü yapan bir insan görüntüsü veri girişimiz olursa, bu resimdeki insan figürü gereksiz ayrıntılardan ve gürültüden ayıştırmak istenir.

### 3.4 Ön İşleme

Ön işleme aşamasında ise işlenecek olan verinin yapılacak olan işleme uygun hale gelmesi planlanır. Regresyon, kümeleme ve filtreleme gibi bir çok ön işleme yöntemleri mevcuttur. Ön işleme aşamasında bazen bir yöntem yeterli olabilirken, bazı durumlarda birden fazla yöntem peş peşe kullanılması gerekebilir (Daş vd. 2019). Öznitelik çıkarma işlemi görüntünün belirtilen doğrultuda anlamlandırma ve görüntünün sahip olduğu özelliklerin hedeflenen özelliklere sahip olup olmadığı kararı verilir (Şimşek vd. 2019).

### 3.5 Nesne Tespiti

Nesne algılama, bir görüntüde veya bir videoda, nesnelerin konumunu ve sınıfını tespit etmeyi amaçlayan bir görevdir (Tan vd. 2021). Nesne tespiti bilgasayarlı görme görevlerinde önemli rol oynamaktadır. Renk değişimleri ve kareler arası farklar, nesne tespiti için önemli etkenlerdir.

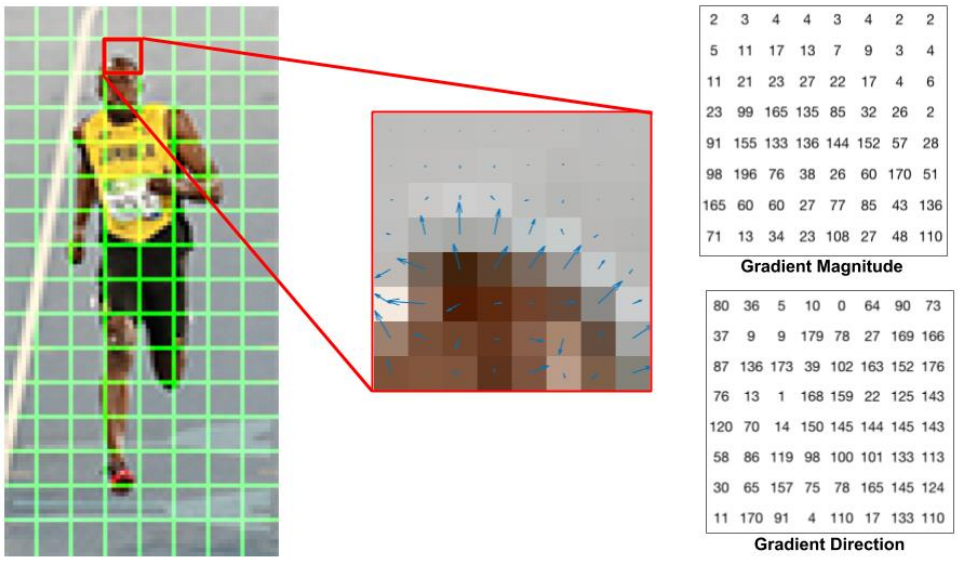
## 3.6 Öznitelik Çıkarımı

Nesne sınıflandırmada aşamalarında öznitelik çıkarımı önemli bir rol oynamaktadır. Girilen verilerin en uygun alt uzayını sınıflandırma performansını arttırmayı amaçlamaktadır (Kocaman ve Gökçe 2022). Nesne tanımada kullanılan öznitelikler literatürde oldukça geniş kapsamlı bir araştırma konusudur. Çünkü nesne tanıma sürekli gelişmektedir (Daş vd. 2019). Girdi olarak verilen görüntü kendine özel özellikler taşımaktadır. Öznitelik çıkarımında tanımlanacak nesnenin değişmez özniteliklerinin belirlenmesi amaçlanır. Literatüre bakıldığında nesne tespit yöntemlerinde, temelde yerel ve global öznitelik çıkarımı olmak üzere iki tip öznitelik çıkarımı mevcuttur. Global öznitelikler görüntüyü bütün olarak alır. Görüntüdeki nesnenin temel özelliklerinden (renk, doku, kenarlık) yola çıkarak nesne ile ilgili tanımlayıcı özellikleri belirler (Çetin 2011). Yerel öznitelikler ise pikseldeki kenar, köşe, eğim gibi özellikleri ele almaktadır. Eğer bir görüntüde insan yüzü olup olmadığı sonucuna varılmak istenirse global, eğer yüz tanıma yapılacaksa yerel öznitelikler kullanılabilir. Bu çalışmada global özniteliklerden HOG, yerel özniteliklerden SIFT ve BRISK öznitelik çıkarma yöntemleri kullanılmıştır.

### 3.6.1 Öznitelik Çıkarma Yöntemleri

#### 3.6.1.1 HOG (Histogram of Oriented Gradients-Yönlendirilmiş Degradelerin Histogramı)

Nesne tanıma ve görüntü işlemede sıklıkla kullanılan bir öznitelik çıkarıcıdır. Girdi olarak verilen görüntünün her bir pikseli kontrol edilerek kenar özelliği olan pikseller ve kenarın yönleri tespit edilir. Tespit işlemlerinin ardından piksellerin gradyanları ve yönleri kullanılarak her bölgenin ayrı ayrı histogramı oluşturulur. Girilen görüntünün en boy oranı HOG için önemlidir. Sabit bir en-boy oranı yapılacak işlemlerin daha kolay uygulanmasını sağlar. En-boy oranı sabit 1:2 şeklinde kullanılmaktadır. Girilen görüntü 8*8 pikseller şeklinde hücrelenir. Bir sonraki aşamada görüntünün yatay ve dikey gradyanları hesaplanır. Hücre içerisinde yoğunluğa göre histogramın yönü ve uzunluğu hesaplanır. Yoğunluğun değişimi yönü değişimin şiddeti uzunluğu belirler.




**Resim 3. 28** HOG öznitelik çıkarma yöntemi

HOG öznitelik çıkarma yöntemi bu çalışmada VisDrone veri seti ve MNIST veri seti üzerinde kullanılmıştır.

#### 3.6.1.1.1 HOG Öznitelik Çıkarıcının VisDrone Veri Setinde Kullanımı

Bu çalışmada HOG özniteliği VisDrone veri seti üzerinde uygulanmıştır. VisDrone veri setinde "insanlar" ve "arabalar" sınıflarından basit rastgele örnekleme ve MNIST veri setinin tüm sınıflarından basit rastgele örnekleme ile alınan veriler HOG öznitelik çıkarıcısına girilmiştir. Öznitelik çıkarıcılarını kullanılır hale getirilmesi için ve bilgisayar içerisinden veri alma, işleme ve saklama işlemleri için gerekli kütüphaneler eklenmiştir. Kütüphaneler yazılımın verimli çalışması içim gereklidir. Kullanıcının kod yazma yükünü hafifletir.

```
import glob
from skimage.io import imread
from skimage.transform import resize
from skimage.feature import hog
import matplotlib.pyplot as plt
from numpy import linalg as LA
```

**Resim 3. 29** HOG öznitelik kullanımı ve diğer işlemler için kullanılan kütüphaneler

Tanımlama işlemi bittiğinde, bilgisayardan öznitelik çıkarıcılarına girecek olan görüntüleri elde etmek için, görüntülerin bilgisayar içerisinde bulunduğu konumun

uzantısı eklenmektedir. Bilgisayarda bulunan dosyaların python tarafından anlaşılabilir olması için "glob" kütüphanesi kullanılır. Yerel bilgisayarda klasörlerin bulunduğu konumlar değişkenlere tanımlanır. Resim 3.30' da örnek kod satırlaerı gösterilmiştir.

```
image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")
```

**Resim 3. 30** Değişkenlere klasörlerin tanımlanması

VisDrone veri setinde öznitelik çıkarıcılarına girecek olan sınıflar "insanlar" ve "araçlar sınıfı olarak belirlenmiştir. Bu iki sınıf işleme girmesi için iki farklı değişkene atanmaktadır. Bu iki sınıf hem kendi aralarında hem de birbirleriyle karşılaştırılmıştır. Karşılaştırma için bir dizi içerisine alınmıştır.

```
listeler=[image_folder_car, image_folder_people]
```

**Resim 3. 31** Dizi içerisine alınan değişkenler

VisDrone veri setinden elde edilen verilerin öznitelik çıkarıcılarına girmesi sonucunda, çıktı olarak elde edilen yeni verilerin saklanması için, yeni değişkenler tanımlanır. Değişkenlere, saklanacak veriler için uzantılar tanımlanır. HOG öznitelik çıkarıcıyla işleme girecek verilerin hem sayısal hem de görsel çıktıları için iki farklı değer tanımlanır. Görsel çıktılar, HOG öznitelik çıkarıcısında işleme giren görüntülerin çıktısını oluşturmaktadır. Sayısal çıktılar, gradyan hesaplamalarının sonucunu oluşturmaktadır. İki farklı sınıf, hem kendi içinde hem de birbirleriyle karşılaştırılacağı için, gradyan hesaplamalarının sonuçları, sayıları ve toplamlarını atamak için değişkenler oluşturulur. Resim 3.32' de değişkenlerin olduğu kod satırları yer almaktadır.

```
dst = 'outputdnm\hog/'
dst2 = 'outputdnm\hog\outputtext/'

toplam=0
a=0
b=0
c=0
```

**Resim 3. 32** Çıktıların saklanacağı uzantıların ve gradyanların atanacağı değişkenler

Tanımlama ve saklama işlemleri için gerekli kodlar yazıldıktan sonra görüntülerin sırasıyla işleme girmesi için döngüler oluşturulmuştur. İç içe döngüler kullanılmıştır. İlk döngü dizi içerisindeki değişkenleri döngü içerisine almaktadır. Bu değişkenlerde, görüntülerin içerisinde bulunduğu klasörlerin konumları bulunmaktadır. İkinci döngü de ise görüntüler döngü içerisine alınır. İç içe döngülerle yazılım, ilk sınıfı HOG özniteliğinde kendi içinde karşılaştırdıktan sonra diğer sınıfla karşılaştırır. Görüntü, HOG öznitelik çıkarıcı için öncelikle, 64*128 piksel oranına uygun hale getirip öznitelik çıkarma işlemine başlamıştır.

```
for liste in listeler:
    for img in liste:
        img = imread(img)
        plt.axis("on")
        resized_img = resize(img, (64*4, 128*4))
```

**Resim 3. 33** İç içe döngüler ve HOG öznitelik çıkarıcısına girecek görüntünün oranlanması

Yeniden boyutlandırılan görüntü, HOG öznitelik çıkarıcısına girdi olarak alınır. Görüntünün gradyanları, gradyan yönleri ve boyutları hesaplanır.

```
fd, hog_image = hog(resized_img, orientations=9,
                    pixels_per_cell=(8, 8),
                    cells_per_block=(2, 2),
                    visualize=True, channel_axis=-1)
```

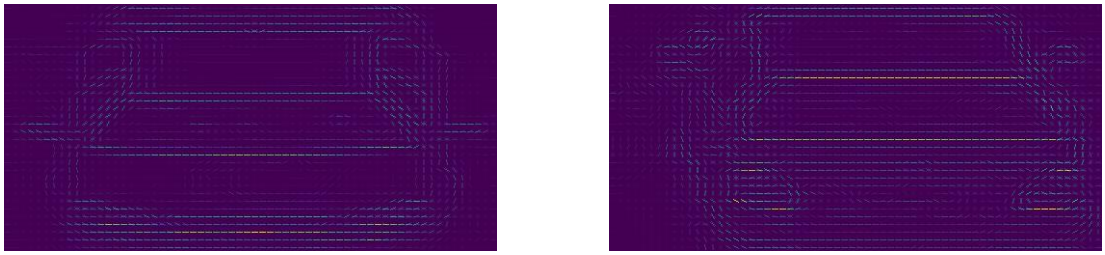
**Resim 3. 34** HOG öznitelik çıkarıcı komutları

Elde edilen yeni görüntünün ekranda görüntülenmesi “imshow”, saklanması için “imsave” komutları kullanılmıştır. “imsave” komutuna sırasıyla görüntünün bilgisayarda saklanacağı konum, dosya adı, dosyanın uzantısı ve saklanılacak dosya şeklinde parametreler eklenir. Döngü içerisinde kaydetme işlemi sürekli olacağından dolayı dosya adı sürekli değişken olarak belirlenir.

```
plt.imshow(hog_image, cmap="gray")
plt.imsave(dst+str(c)+".jpg",hog_image,)
c+=1
```

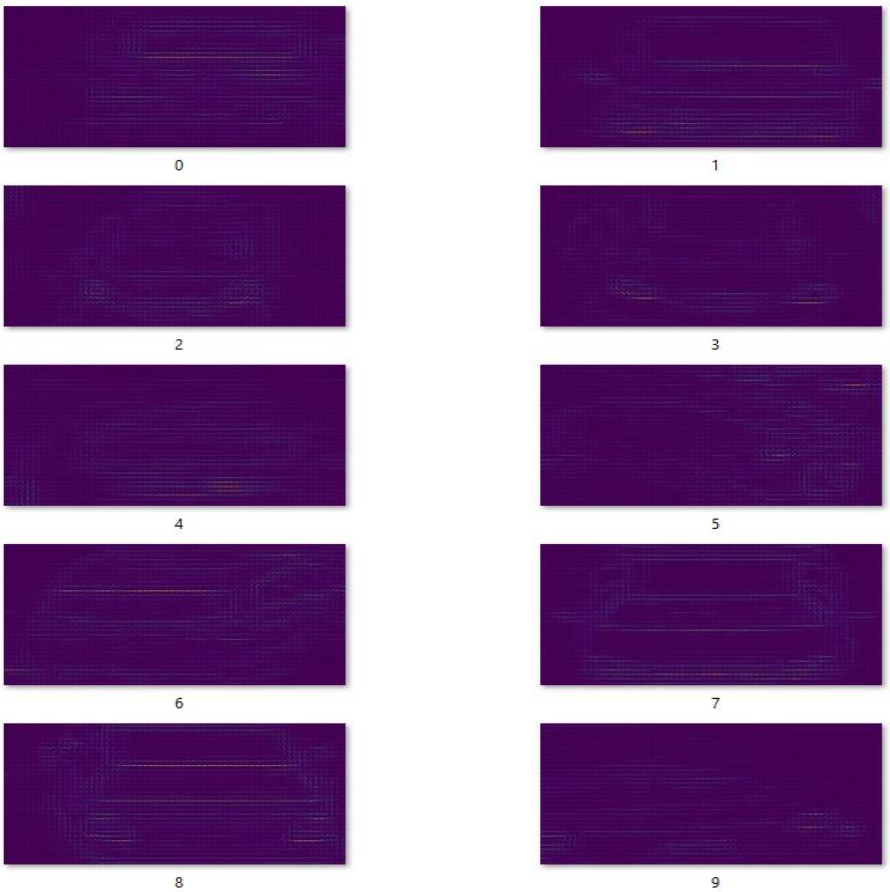
**Resim 3. 35** Elde edilen görüntüyü saklama ve ekranda gösterme komutları

HOG öznitelik çıkarıcılarından elde edilen görüntüler, 64*128 boyutlarında gradyanların yönleri ve uzunlukları Resim 3.36’ da görüldüğü gibidir.



**Resim 3. 36** HOG öznitelik çıkarıcılarından elde edilen görüntü

HOG öznitelik çıkarıcıdan elde edilen görüntüler klasör içerisinde saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir.




**Resim 3. 37** HOG öznitelik çıkarıcıdan elde edilen görntülerin saklanması

Elde edilen veriler doğrultusunda gradyan değerleri ve görseller, var olan döngü içerisinde tekrar iç içe döngü ile işleme alınır. Burada amaç, her bir görüntünün

işlenmesinin yanı sıra, her bir görüntünün birbirleriyle karşılaştırılmasıdır. Bu karşılaştırma sonucu aynı sınıfa ait olan değerlerin "sınıf içi", farklı sınıflara ait olan değerlerin "sınıflar arası" farkının hesaplanması amaçlanmıştır. Her bir öznitelik çıkarıcı için bu işlem uygulanmıştır. Tekrar oluşturduğumuz döngümüzde, bir önceki işlemler tekrarlanmaktadır. Farklı olarak bu komutlar arasında elde edilen sonuçların görüntülenmesi ve saklanması önceki komutlarda elde edildiği için, bu iç içe döngüde görüntülenme ve saklama yer almamıştır. İlk döngüde olduğu gibi aynı işlemler ikinci döngüde de tekrarlanır. Görüntü HOG öznitelik çıkarıcıda işleme alınır.

```
for liste2 in listeler:
    for img2 in liste2:

        img2 = imread(img2)
        resized_img2 = resize(img2, (64*4, 128*4))
        plt.axis("on")
        fd2, hog_image2 = hog(resized_img2, orientations=9,
                              pixels_per_cell=(8, 8),
                              cells_per_block=(2, 2),
                              visualize=True, channel_axis=-1)
```

**Resim 3. 38** İç içe döngü içerisinde görüntünün HOG öznitelik çıkarıcı tarafından işlenmesi

İkinci fotoğrafın öznitelik çıkarma işlemi ile birlikte iki adet görsel çıktı elde edilmiştir. Elde edilen görüntüler aynı sınıfta ise "sınıf içi uzaklık" farklı "sınıfta ise sınıflar arası uzaklık" değerlerini vermektedir. Bu değerlere ulaşmak için elde edilen gradyan değerlerinden matris değerleri oluşturulur. İki görselin matris değerleri arasındaki fark, değerlendirme için kullanılmıştır. Elde edilen matris değerinin normu hesaplanır. Resim 3.39' da bu matris farkı ve norm dönüşümü verilmiştir. Olası bir hata ile karşılaşılması ve çalışan yazılımın aksamasını önlemek için "try, except" parametreleri kullanılmıştır.

```
try:

    tempFd = fd - fd2
    tempNorm1 = LA.norm(tempFd)
    toplam +=tempNorm1
    a+=1
    b+=1
    print(a,tempNorm1,toplam)

except:

    print(a,"no")
```

**Resim 3. 39** Gradyan değerlerinin matrise dönüştürülüp farklarının hesaplanması

HOG öznitelik çıkarıcı ile elde edilen değerler, diğer bütün görsellerle matris farkı bulmak için karşılaştırılır. Bu işlem tamamlandıktan sonra bir sonraki görsel aynı işlemi tekrar eder. Karşılaştırma işlemi bittiğinde elde edilen değerlerin ortalaması alınmaktadır. Görselin kendi sınıfına  ve diğer sınıflara ait bir ortalaması alınmıştır. Kendi sınıfına ait ortalamalar sınıf içi uzaklıkta diğer sınıflara ağit ortalamalar sınıflar arası uzaklıkta kullanılmıştır. Resim 3.40' ta ortalama işleminin kodları verilmiştir.

```
try:

    ortalama=toplam/a

except:
    print("hata")
```

**Resim 3. 40** Görselin kendi ve diğer sınıflara ait ortalama hesaplama işlemi

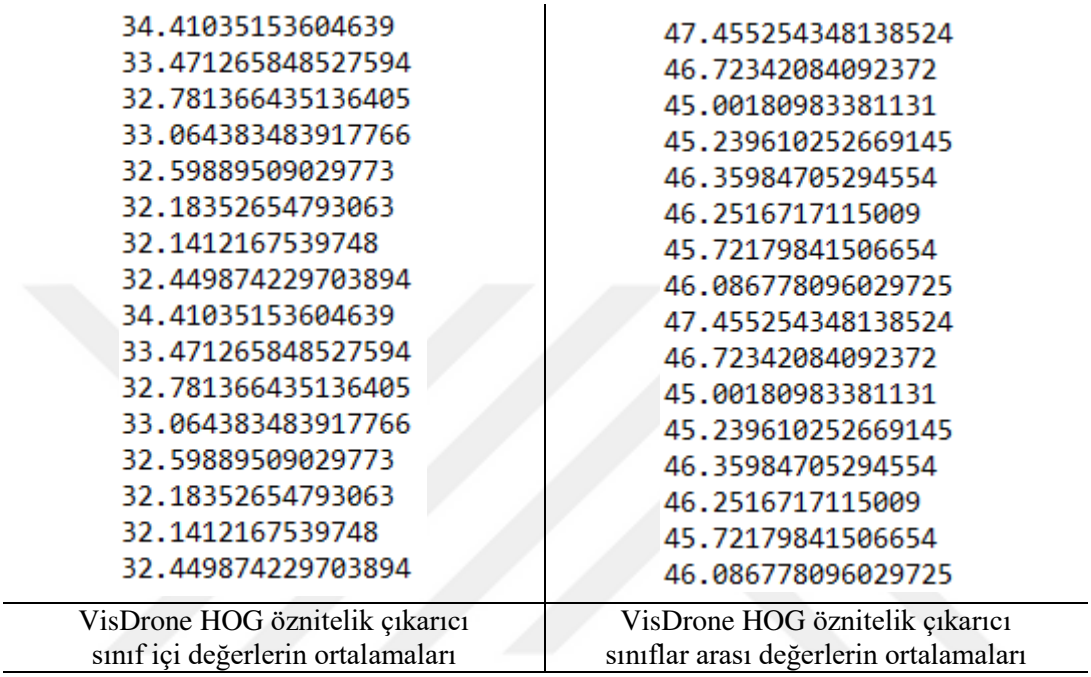
Ortalama değerleri görselden görsele farklılık göstermektedir. her görselin kendine ait değerleri elde edilir. Bu değerler metin belgesi (.txt) uzantılı dosyalarda saklanır. Ortalamalar sınıflara göre ayrılır. Aynı sınıfa ait ortalamalar ile farklı sınıflara ait ortalamalar, farklı dosyalarda saklanır. Bu ayrım "if" komutu ile sağlanmıştır. Resim 3.41' de farklı dosyalarda saklama işlemi gösterilmiştir. Metin belgesi dosyasının içerisine değerler satırlar halinde otomatik olarak atanmaktadır.

```
if liste==liste2:

    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronehogsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()

else:

    print("nesneler arası uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronehogsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

**Resim 3. 41** Ortalamaların ayrılıştırılma ve saklanması için yazılan kodlar

Ortalaması alınan değerler sınıflara göre ayrılıp kaydedilmiştir. kayıt için önce dosyanın bulunduğu konum, saklanacak dosyanın adı ve uzantısı ve yapılacak işlemin

tanımlanması (açma, değiştirme, silme vb.) için gerekli kodlamalar yapılır. Bu çalışmada metin belgesine sürekli veri girişi olacağı için "a" komutu kullanılmıştır. Dosyaya yazma işlemi için "writelines" komutu kullanılmıştır. Elde edilen ortalama değerlerinin dosya içerisinde alt alta eklenebilmesi için "\n" komutu eklenerek resim 3.42' deki gibi değerler elde edilmiştir.




**Resim 3. 42** VisDrone veri setine ait HOG öznitelik çıkarıcı verileri

VisDrone veri seti için hazırlanan HOG öznitelik çıkarma yönteminin kodları Ek 3'te yer almaktadır.

#### 3.6.1.1.2 HOG Öznitelik Çıkarıcının MNIST Veri Setinde Kullanımı

HOG öznitelik çıkarıcı VisDrone veri seti yanı sıra MNIST veri setinde de kullanılmıştır. MNIST veri seti 0-9 arası rakamların farklı el yazılarıyla yazılmış, 28*28 pixel boyutuna sahip görüntülerden oluşmaktadır. MNIST veri setinin, HOG öznitelik çıkarma yöntemi, VisDrone veri setinde kullanılan yönteme benzer komutlardan oluşmaktadır. Aynı komutlar kullanılmıştır. Farklı olarak girdi olarak MNIST veri setinden görseller alınmıştır. Çıktı olarak yine MNIST veri seti farklı bir uzantıda saklanmıştır.

```
image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")
image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")
image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")
image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")
image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")
image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")
image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")
image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")
image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")
image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

**Resim 3. 43** MNIST veri seti dosya uzantılarının değişkenlere atanması

Resim 3.43 'te HOG öznitelik çıkarıcı için, MNIST veri seti girdi olarak kullanılmıştır. 10 sınıftan oluşan MNIST veri setinin her sınıfı HOG öznitelik çıkarıcı ile işleme alınmıştır. Değişkenlere atanan her sınıf , "listler" dizisi içerisinde bir grup haline getirilmiştir. Resim 3.44' te dizi içerisinde değişkenler gösterilmiştir.

```
listeler=[image_folder_zerooo,image_folder_one,image_folder_two,
          image_folder_three,image_folder_four,
         image_folder_five,image_folder_six,image_folder_seven,
         image_folder_eight,image_folder_nine]
```

**Resim 3. 44** Dizi içerisinde MNIST veri seti

VisDrone veri setinde kullanılan komutlar, MNIST veri setinde HOG yöntemi uygulamak için de kullanılmıştır. Döngüler, koşullar, öznitelik çıkarna komutlarında herhangi bir değişiklik yapılmamıştır. Çıktı verilerinin saklanması için, Resim 3.45' te yer alan komutlardan faydalanılmıştır.

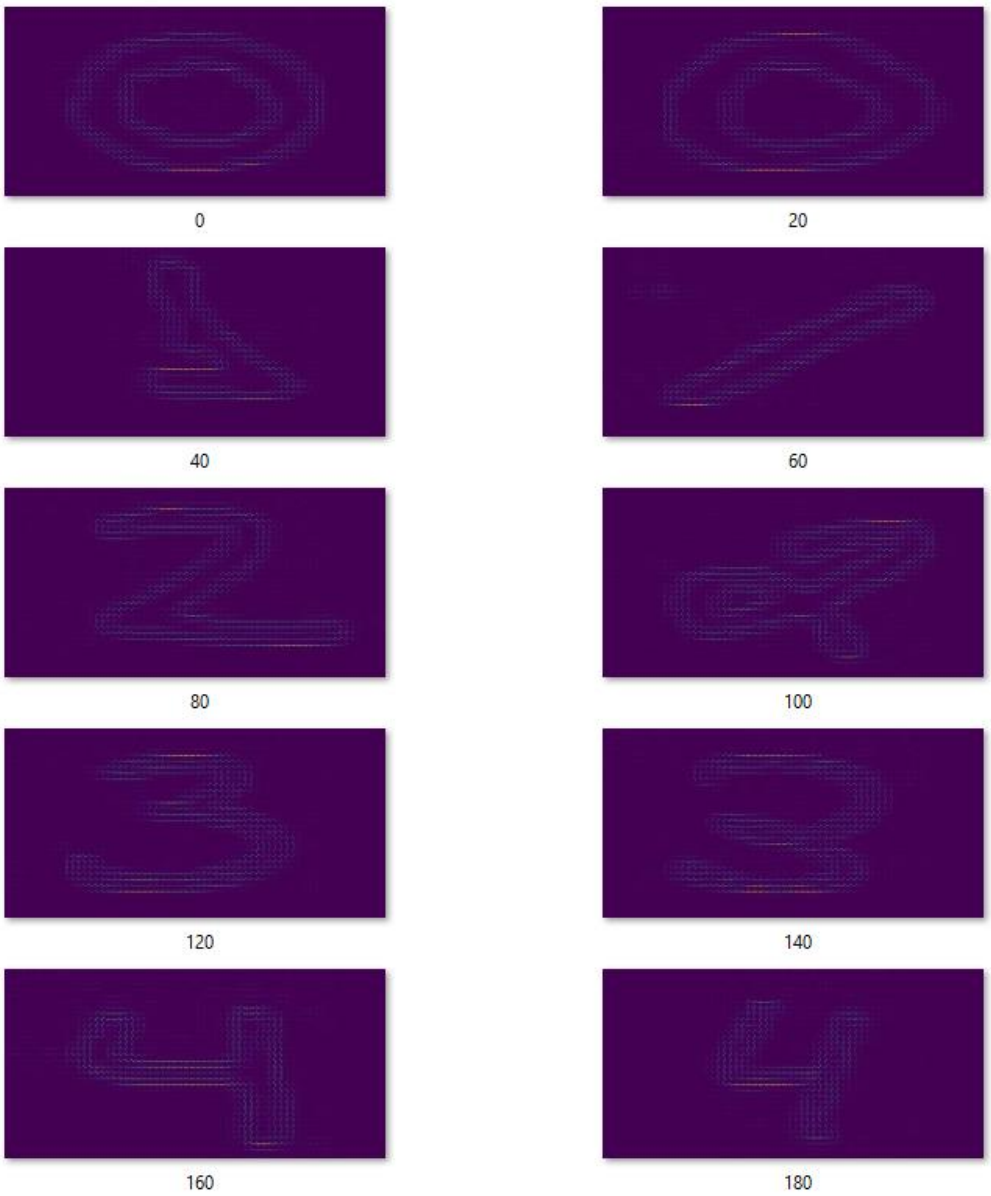
```
dst = 'outputdnm\hog/'
dst2 = 'outputdnm\hog\outputtext/'
```

**Resim 3. 45** MNIST veri seti, HOG yöntemi çıktı dosyalarının saklanması

VisDrone veri setinde olduğu gibi, HOG öznitelik çıkarıcılarından elde edilen görüntüler, 64*128 boyutlarında gradyanların yönleri uzunlukları Resim 3.46' da görüldüğü gibidir.


**Resim 3. 46** HOG öznitelik çıkarıcılarından elde edilen görüntü

VisDrone veri setinde olduğu gibi, HOG öznitelik çıkarıcıdan elde edilen görüntüler yerel bilgisayarda resim 3.47' de görüldüğü gibi klasör içerisinde saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir.




**Resim 3. 47** HOG öznitelik çıkarıcıdan elde edilen görntülerin saklanması

HOG öznitelik çıkarma yöntemi kullanılarak MNIST veri seti işlenmiştir. Görsellerle beraber MNIST veri setinde bulunan 10 sınıf için, sınıf içi ve sınıflar arası uzaklıklar hesaplanmıştır. Aynı sınıfta olan görseller için yapılan hesaplamalar sınıf içi uzaklık olarak tanımlanmıştır. Farklı sınıfta olan görseller için yapılan hesaplamalar sınıflar arası uzaklık olarak tanımlanmıştır.

| | |
|---|---|
| 20.63144854927139<br>20.63144854927139<br>21.30114452712623<br>21.30114452712623<br>23.387149021100594<br>23.387149021100594<br>20.204856378920272<br>20.204856378920272<br>21.80387289629208<br>21.80387289629208<br>19.560093189773504<br>19.560093189773504<br>20.7419226790223<br>20.7419226790223<br>22.708836144521182<br>22.708836144521182 | 44.430127871774864<br>45.8634604228408<br>42.30227324769959<br>45.44769457238323<br>44.50712062142378<br>43.12816942566903<br>45.425650572638695<br>45.311854372080106<br>44.65085207820685<br>45.64006910547971<br>46.51296861775633<br>44.27676766320771<br>47.74446637192311<br>44.773074642375<br>44.63068019213184<br>47.28061986371236 |
| MNIST HOG öznitelik çıkarıcı sınıf içi değerlerin ortalamaları | MNIST HOG öznitelik çıkarıcı sınıflar arası değerlerin ortalamaları |

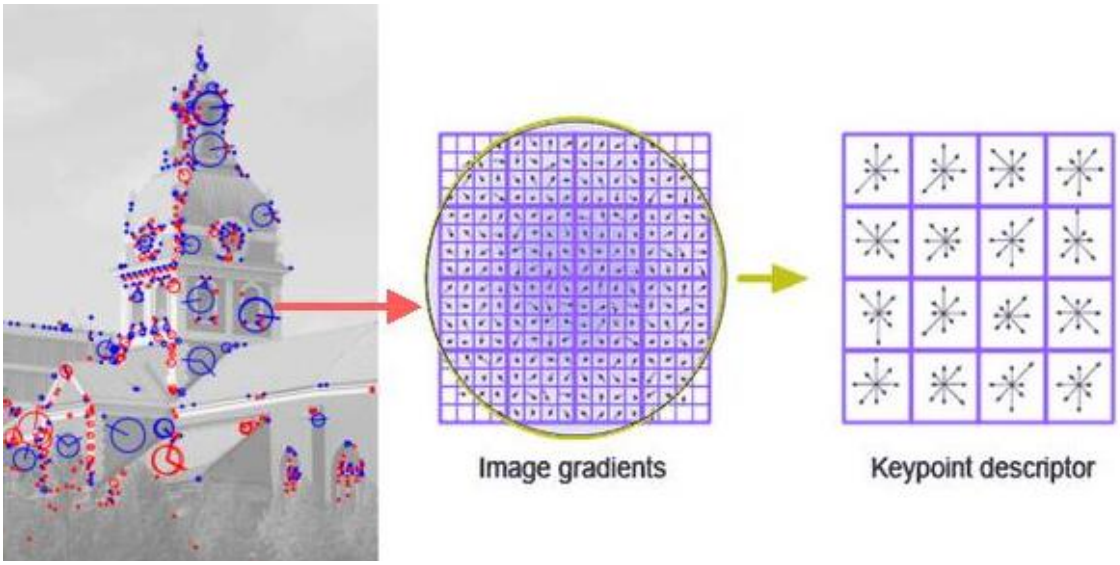
**Resim 3. 48** MNIST veri setine ait HOG öznitelik çıkarıcı değerleri

Resim 3.48' de MNIST veri setinin sınıf içi ve sınıflar arası uzaklıklarının hesaplamaları verilmiştir. MNIST veri seti için hazırlanan HOG öznitelik çıkarma yönteminin kodları Ek 4'te yer almaktadır.

#### 3.6.1.2 SIFT (Scale Invarıant Feature Transform – Ölçek Değişmez Unsur Dönüşümü)

SIFT David Lowe tarafından geliştirilen yerel öznitelik çıkarma yöntemidir. Bu yöntem görüntünün anahtar noktalarını belirler ve tanımlayıcılarını hesaplayarak nesne tanıma işlemi gerçekleştirir (Tosun 2019). SIFT öznitelik çıkarma yönteminde öncelikle görüntünün ölçek alan filtrelemesi yapılır. Ölçek alan filtrelemesi, anahtar nokta belirlemek için kullanılır. Görüntünün farklı ölçeklerinde belirlenen noktaların, istikrarlı olması durumunda anahtar nokta olarak işaretlenir. Sonraki aşamada anahtar noktaların

filtreleme işlemi yapılır. Anahtar noktaların bazıları gürültü ve ışıktan çabuk etkilenir. Bu noktaların temizlenmesi gerekmektedir. Anahtar noktalar belirlendikten sonra görüntünün döndürülmesi durumunda noktaların özelliklerinin korunması için her bir anahtar noktaya yön ataması yapılır. Anahtar noktanın etrafındaki 16*16'lık komşulları alınarak bir bölge oluşturulur. Bu bölge 4*4 boyutlarında 16 alt bloğa ayrılır. Her alt blok için 8 bölmeli oryantasyon histogramı oluşturulur. Toplam olarak 128 bin değer oluşturulmuş olur. Resim 3.49'da SIFT öznitelik çıkarma yöntemi kullanılan örnek görsel verilmiştir.




**Resim 3. 49** SIFT öznitelik çıkarıcının görselde kullanılması

### 3.6.1.2.1 SIFT Öznitelik Çıkarıcının VisDrone Veri Setinde Kullanımı

Bu çalışmada SIFT özniteliği hem VisDrone veri seti hem de MNIST veri seti üzerinde uygulanmıştır. VisDrone veri setinde "insanlar" ve "arabalar" sınıflarından basit rastgele örnekleme ve MNIST veri setinin tüm sınıflarından basit rastgele örnekleme ile alınan veriler SIFT öznitelik çıkarıcısına girilmiştir. Öznitelik çıkarıcılarını kullanılır hale getirilmesi için ve bilgisayar içerisinden veri alma, işleme ve saklama işlemleri için gerekli kütüphaneler eklenmiştir.

```
import cv2
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
import numpy as np
import matplotlib as inline
from cv2 import xfeatures2d
from scipy.spatial import distance
from numpy import linalg as LA
import glob
```

**Resim 3. 50** SIFT öznitelik kullanımı ve diğer işlemler için kullanılan kütüphaneler

Tanımlama işlemi bittiğinde, bilgisayardan öznitelik çıkarıcılarına girecek olan görüntüleri elde etmek için, görüntülerin bilgisayar içerisinde bulunduğu konumun uzantısı eklenmektedir. Bilgisayarda bulunan dosyaların python tarafından anlaşılabilir olması için "glob" kütüphanesi kullanılır. Yerel bilgisayarda klasörlerin bulunduğu konumlar değişkenlere tanımlanır.

```
image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")
```

**Resim 3. 51** Değişkenlere klasörlerin tanımlanması

VisDrone veri setinde öznitelik çıkarıcılarına girecek olan sınıflar "insanlar" ve "araçlar sınıfı olarak belirlenmiştir. Bu iki sınıf işleme girmesi için iki farklı değişkene atanmaktadır. Bu iki sınıf hem kendi aralarında hem de birbirleriyle karşılaştırılmıştır. Karşılaştırma için bir dizi içerisine alınmıştır.

```
listeler=[image_folder_car, image_folder_people]
```

**Resim 3. 52** Dizi içerisine alınan değişkenler

VisDrone veri setinden elde edilen verilerin öznitelik çıkarıcılarına girmesi sonucunda, çıktı olarak elde edilen yeni verilerin saklanması için, yeni değişkenler tanımlanır. Değişkenlere, saklanacak veriler için uzantılar tanımlanır. SIFT öznitelik çıkarıcıyla işleme girecek verilerin hem sayısal hem de görsel çıktıları için iki farklı değer tanımlanır. Görsel çıktılar, SIFT öznitelik çıkarıcısında işleme giren görüntülerin çıktısını oluşturmaktadır. Sayısal çıktılar, anahtar nokta hesaplamalarının sonucunu oluşturmaktadır. İki farklı sınıf, hem kendi içinde hem de birbirleriyle karşılaştırılacağı

için, anahtar nokta hesaplamalarının sonuçları, sayıları ve toplamlarını atamak için değişkenler oluşturulur.

```
dst = 'outputdnm/sift/'
dst2 = 'outputdnm\sift\outputtext/'

toplam=0
a=0
b=0
c=0
```

**Resim 3. 53** Çıktıların saklanacağı uzantıların ve anahtar noktaların atanacağı değişkenler

Tanımlama ve saklama işlemleri için gerekli kodlar yazıldıktan sonra görüntülerin sırasıyla işleme girmesi için döngüler oluşturulmuştur. İç içe döngüler kullanılmıştır. İlk döngü dizi içerisindeki değişkenleri döngü içerisine almaktadır. Bu değişkenlerde, görüntülerin içerisinde bulunduğu klasörlerin konumları bulunmaktadır. İkinci döngü de ise görüntüler döngü içerisine alınır. İç içe döngülerle yazılım, ilk sınıfı SIFT özniteliğinde kendi içinde karşılaştırdıktan sonra diğer sınıfla karşılaştırır. Görüntü, SIFT öznitelik çıkarıcı için uygun hale getirilip öznitelik çıkarma işlemine başlanmıştır.

```
for liste in listeler:
    for img in liste:
        print(liste)
        img = cv.imread(img)
        gray= cv.cvtColor(img,cv.COLOR_BGR2GRAY)
        sift = cv.SIFT_create()
```

**Resim 3. 54** İç içe döngüler ve SIFT öznitelik çıkarıcısına girecek görüntünün oranlanması

Yeniden boyutlandırılan görüntü, SIFT öznitelik çıkarıcısına girdi olarak alınır. Görüntünün ölçek alan ve anahtar noktaları hesaplanır.

```
kp, des = sift.detectAndCompute(gray,None)
img=cv.drawKeypoints(gray,kp,img,
                     flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
plt.imshow(img, cmap="gray")
plt.imsave(dst+str(c)+".jpg",img,)#cmap="gray")
c+=1
```

**Resim 3. 55** SIFT öznitelik çıkarıcı komutları

Elde edilen yeni görüntünün ekranda görüntülenmesi için “imshow”, saklanması için “imsave” komutları kullanılmıştır. “imsave” komutuna sırasıyla görüntünün bilgisayarda saklanacağı konum, dosya adı, dosyanın uzantısı ve saklanılacak dosya şeklinde parametreler eklenir. Döngü içerisinde kaydetme işlemi sürekli olacağından dolayı dosya adı sürekli değişken olarak belirlenir.

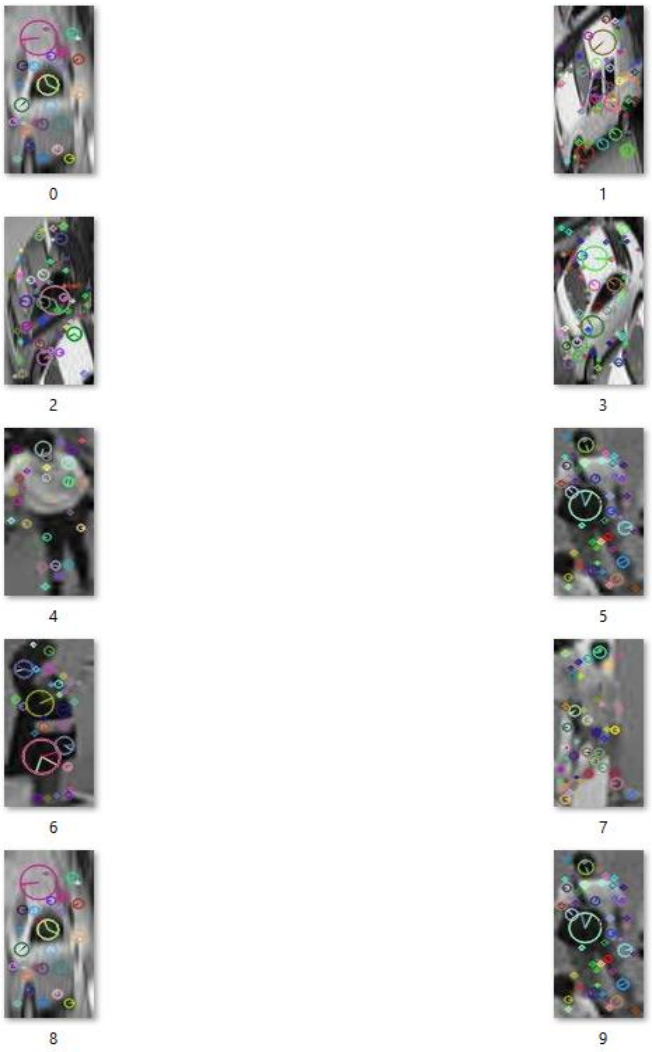
```
plt.imshow(img, cmap="gray")
plt.imsave(dst+str(c)+".jpg",img,)#cmap="gray")
c+=1
```

**Resim 3. 56** Elde edilen görüntüyü saklama ve ekranda gösterme komutları

SIFT öznitelik çıkarıcılarından elde edilen görüntüler, anahtar noktalar ve yönelimleri Resim 3.57’ de görüldüğü gibidir.

**Resim 3. 57** SIFT öznitelik çıkarıcılarından elde edilen görüntü

SIFT öznitelik çıkarıcıdan elde edilen görüntüler klasör içerisinde saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir.




**Resim 3. 58** SIFT öznitelik çıkarıcıdan elde edilen görntülerin saklanması

Elde edilen veriler doğrultusunda gradyan değerleri ve görseller, var olan döngü içerisinde tekrar iç içe döngü ile işleme alınır. Burada amaç, her bir görüntünün işlenmesinin yanı sıra, her bir görüntünün birbirleriyle karşılaştırılmasıdır. Bu karşılaştırma sonucu aynı sınıfa ait olan değerlerin "sınıf içi" farklı sınıflara ait olan değerlerin "sınıflar arası" farkının hesaplanması amaçlanmıştır. Her bir öznitelik çıkarıcı için bu işlem uygulanmıştır. Tekrar oluşturduğumuz döngümüzde, bir önceki işlemler tekrarlanmaktadır. Farklı olarak bu komutlar arasında elde edilen sonuçların görüntülenmesi ve saklanması önceki komutlarda elde edildiği için, bu iç içe döngüde görüntülenme ve saklama yer almamıştır. İlk döngüde olduğu gibi aynı işlemler ikinci döngüde de tekrarlanır. Görüntü SIFT öznitelik çıkarıcıda işleme alınır.

```
for liste2 in listeler:
    for img2 in liste2:
        img2 = cv.imread(img2)
        gray2= cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
        sift2 = cv.SIFT_create()
        kp2, des2 = sift2.detectAndCompute(gray2,None)
        img2=cv.drawKeypoints(gray2,
                              kp2,img2,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

**Resim 3. 59** İç içe döngü içerisinde görüntünün SIFT öznitelik çıkarıcı tarafından işlenmesi

İkinci fotoğrafın öznitelik çıkarma işlemi ile birlikte iki adet görsel çıktı elde edilmiştir. Elde edilen görüntüler aynı sınıfta ise "sınıf içi uzaklık", farklı "sınıfta ise sınıflar arası uzaklık" değerlerini vermektedir. Bu değerlere ulaşmak için elde edilen gradyan değerlerinden matris değerleri oluşturulur. İki görselin matris değerleri arasındaki farkı alınımıştır. Bu farkın normu değerlendirme için kullanılmıştır. Resim 3.60' da bu matris hesaplaması verilmiştir. Olası bir hata ile karşılaşılması ve çalışan yazılımın aksamasını önlemek için "try, except" parametreleri kullanılmıştır.

```
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)

try:
    matches = bf.match(des,des2)

    matched_img = cv2.drawMatches(img, kp, img2, kp2,
                                  matches[:50], img2, flags=2)
    tempFd=len(kp)-len(kp2)
    tempNorm1=LA.norm(tempFd)
    tempFd2=len(img)-len(img2)
    tempNorm2=LA.norm(tempFd2)
    toplam +=tempNorm1
    a+=1
    print(a,tempNorm1,toplam)
except:
    print(a,"no")
```

**Resim 3. 60** Gradyan değerlerinin matrise dönüştürülüp farklarının hesaplanması

SIFT öznitelik çıkarıcı ile elde edilen değerler, diğer bütün görsellerle matris farkını bulmak için karşılaştırılır. Bu işlem tamamlandıktan sonra bir sonraki görsel aynı işlemi tekrar eder. Karşılaştırma işlemi bittiğinde elde edilen değerlerin ortalaması alınmaktadır. Görselin kendi sınıfına ve diğer sınıflara ait bir ortalaması alınmıştır. Kendi sınıfına ait ortalamalar sınıf içi uzaklıkta diğer sınıflara ait ortalamalar sınıflar arası uzaklıkta kullanılmıştır. Resim 3.61' de ortalama işleminin kodları verilmiştir.

```
try:

    ortalama=toplam/a

except:
    print("hata")
```

**Resim 3. 61** Görselin kendi ve diğer sınıflara ait ortalama hesaplama işlemi

Ortalama değerleri görselden görsele farklılık göstermektedir. her görselin kendine ait değerleri elde edilir. Bu değerler metin belgesi (.txt) uzantılı dosyalarda saklanır. Ortalamalar sınıflara göre ayrılır. Aynı sınıfa ait ortalamalar ile farklı sınıflara ait ortalamalar, farklı dosyalarda saklanır. Bu ayrım "if" komutu ile sağlanmıştır. Resim 3.62' de farklı dosyalarda saklama işlemi gösterilmiştir. Metin belgesi dosyasının içerisine değerler satırlar halinde otomatik olarak atanmaktadır.

```
if liste==liste2:
    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronesiftsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()

else:

    print("nesneler arası uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronesiftsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

**Resim 3. 62** Ortalamaların ayrılıştırılma ve saklanması için yazılan kodlar

Ortalaması alınan değerler sınıflara göre ayrılıp kaydedilmiştir. Kayıt için önce dosyanın bulunduğu konum, saklanacak dosyanın adı, uzantısı ve yapılacak işlemin tanımlanması (açma, değiştirme, silme vb.) için gerekli kodlamalar yapılır. Bu çalışmada metin belgesine sürekli veri girişi olacağı için "a" komutu kullanılmıştır. Dosyaya yazma işlemi için "writelines" komutu kullanılmıştır. Elde edilen ortalama değerlerinin dosya içerisinde alt alta eklenebilmesi için "\n" komutu eklenerek Resim 3.63' te görüldüğü gibi değerler elde edilmiştir.

| | |
|---|---|
| 31.75<br>11.75<br>11.75<br>12.25<br>10.75<br>14.25<br>10.75<br>10.75 | 14.75<br>25.25<br>28.25<br>29.25<br>29.75<br>17.25<br>29.75<br>20.75 |
| VisDrone SIFT öznitelik çıkarıcı sınıf içi değerlerin ortalamaları | VisDrone SIFT öznitelik çıkarıcı sınıflar arası değerlerin ortalamaları |

**Resim 3. 63** VisDrone veri setine ait SIFT öznitelik çıkarıcıdan elde edilen değerler

Ek 5' te yer alan kodlarda VisDrone veri seti için kullanılan SIFT öznitelik çıkarma yönteminin kodları yer almaktadır.

#### 3.6.1.2.2 SIFT Öznitelik Çıkarıcının MNIST Veri Setinde Kullanımı

SIFT öznitelik çıkarıcı, VisDrone veri seti yanı sıra MNIST veri setinde de kullanılmıştır. MNIST veri seti 0-9 arası rakamların farklı el yazılarıyla yazılmış, 28*28 pixel boyutuna sahip görüntülerden oluşmaktadır. MNIST veri setinin, SIFT öznitelik çıkarma yöntemi, VisDrone veri setinde kullanılan yönteme benzer komutlardan oluşmaktadır. Aynı komutlar kullanılmıştır. Farklı olarak MNIST veri setinden görseller girdi olarak kullanılmıştır. Çıktı olarak yine MNIST veri seti farklı bir uzantıda saklanmıştır.

```
image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")
image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")
image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")
image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")
image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")
image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")
image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")
image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")
image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")
image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

**Resim 3. 64** MNIST veri seti dosya uzantılarının değişkenlere atanması

Resim 3.64' te SIFT öznitelik çıkarıcı için, MNIST veri seti girdi olarak kullanılmıştır. 10 sınıftan oluşan MNIST veri setinin her sınıfı SIFT öznitelik çıkarıcı ile işleme alınmıştır. Değişkenlere atanan her sınıf , "listler" dizisi içerisinde bir grup haline getirilmiştir. Resim 3.65' te dizi içerisinde değişkenler gösterilmiştir.

```
listeler=[image_folder_zerooo,image_folder_one,image_folder_two,
          image_folder_three,image_folder_four,
         image_folder_five,image_folder_six,image_folder_seven,
         image_folder_eight,image_folder_nine]
```

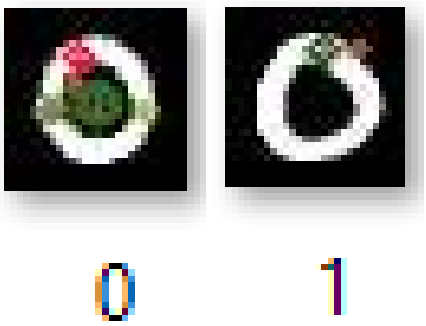
**Resim 3. 65** Dizi içerisinde MNIST veri seti

VisDrone veri setinde kullanılan komutlar, MNIST veri setinde SIFT yöntemi uygulamak için de kullanılmıştır. Döngüler, koşullar, öznitelik çıkarna komutlarında herhangi bir değişiklik yapılmamıştır. Çıktı verilerinin saklanması için, Resim 3.66' da yer alan komutlardan faydalanılmıştır.

```
dst = 'outputdnm/sift/'
dst2 = 'outputdnm\sift\outputtext/'
```

**Resim 3. 66** MNIST veri seti, SIFT yöntemi çıktı dosyalarının saklanması

VisDrone veri setinde olduğu gibi, SIFT öznitelik çıkarıcılarından elde edilen görüntüler, 64*128 boyutlarında gradyanların yönleri ve uzunlukları Resim 3.67' de görüldüğü gibidir.




**Resim 3. 67** SIFT öznitelik çıkarıcılarından elde edilen görüntü

VisDrone veri setinde olduğu gibi, SIFT öznitelik çıkarıcıdan elde edilen görüntüler klasör içerisinde saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir. HOG öznitelik çıkarıcıdan elde edilen görüntüler klasör içerisinde saklanmaktadır.


**Resim 3. 68** SIFT öznitelik çıkarıcıdan elde edilen görntülerin saklanması

SIFT öznitelik çıkarma yöntemi kullanılarak MNIST veri seti işlenmiştir. Görsellerle beraber MNIST veri setinde bulunan 10 sınıf için, sınıf içi ve sınıflar arası uzaklıklar hesaplanmıştır. Aynı sınıfta olan görseller için yapılan hesaplamalar sınıf içi uzaklık olarak tanımlanmıştır. Farklı sınıfta olan görseller için yapılan hesaplamalar sınıflar arası uzaklık olarak tanımlanmıştır.

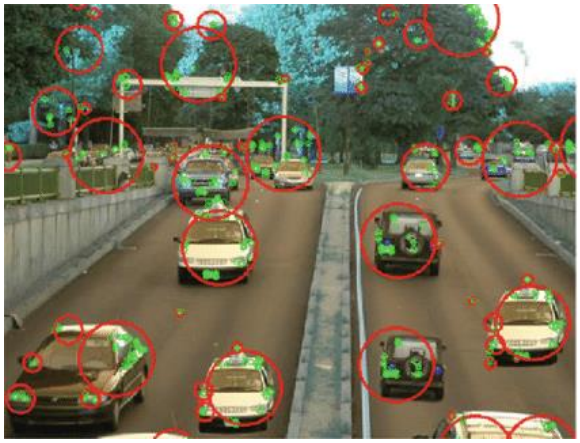
| | |
|---|---|
| 4.5<br>2.0<br>3.5<br>2.5<br>7.0<br>3.5<br>2.0<br>2.5<br>0.0<br>1.5<br>4.5 | 4.0<br>3.5<br>6.5<br>7.0<br>5.5<br>3.0<br>3.5<br>4.0<br>2.5<br>5.0<br>9.5 |
| MNIST SIFT öznitelik çıkarıcı sınıf içi değerlerin ortalamaları | MNIST SIFT öznitelik çıkarıcı sınıflar arası değerlerin ortalamaları |

**Resim 3. 69** MNIST veri setine ait SIFT öznitelik çıkarıcıdan elde edilen değerler

Resim 3.69'da MNIST veri setinin sınıf içi ve sınıflar arası uzaklıklarının hesaplamaları verilmiştir. Ek 6' da yer alan kodlarda, MNIST veri seti için, SIFT öznitelik çıkarma yönteminin kodları yer almaktadır.

#### 3.6.1.3 ORB (Oriented FAST ve Rotated BRIEF / Yönlendirilmiş FAST ve Döndürülmüş BRIEF)

ORB öznitelik çıkarma yöntemi fast ve brief tanımlayıcıların birleştirilip harmanlanmasıyla geliştirilen bir öznitelik çıkarma yöntemidir. Düşük işlem gücü ve yüksek performans sağlayan alternatif bir öznitelik çıkarma yöntemidir. (Rublee vd. 2011). ORB, görüntüde keskin piksel değişimlerini algılar ve bu noktaları anahtar nokta olarak belirler. Bu belirlenen noktalardan nesnenin köşe noktaları belirlenir ve nesne tespiti yapılır (Üncü vd. 2021). Resim 3.70' te anahtar noktaları belirlenmiş bir görüntü örneği verilmiştir.



**Resim 3. 70** ORB anahtar nokta tespiti

Görselde görüldüğü gibi ORB öznitelik çıkarıcısı görseldeki piksellerin keskin farklılıklarını kullanarak yeşil daire içine alır. bu noktaları kullanarak nesne tespiti yapar.

#### 3.6.1.3.1 ORB Öznitelik Çıkarıcının VisDrone Veri Setinde Kullanımı

ORB özniteliği hem VisDrone veri seti hem de MNIST veri seti üzerinde uygulanmıştır. Diğer öznitelik çıkarma yöntemlerinde olduğu gibi ORB öznitelik çıkarma yöntemi VisDrone veri setinde "insanlar" ve "arabalar" sınıflarından basit rastgele örnekleme ve MNIST veri setinin tüm sınıflarından basit rastgele örnekleme ile alınan veriler ORB öznitelik çıkarıcısına girilmiştir. Öznitelik çıkarıcılarını kullanılır hale getirilmesi için ve bilgisayar içerisinden veri alma, işleme ve saklama işlemleri için gerekli kütüphaneler eklenmiştir.

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from numpy import linalg as LA
import cv2
import numpy as np
import glob
```

**Resim 3. 71** ORB öznitelik kullanımı ve diğer işlemler için kullanılan kütüphaneler

Tanımlama işlemi bittiğinde, bilgisayardan öznitelik çıkarıcılarına girecek olan görüntüleri elde etmek için, görüntülerin bilgisayar içerisinde bulunduğu konumun uzantısı eklenmektedir. Bilgisayarda bulunan dosyaların python tarafından anlaşılabilir olması için "glob" kütüphanesi kullanılır. Yerel bilgisayarda klasörlerin bulunduğu konumlar değişkenlere tanımlanır.

```
image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")
```

**Resim 3. 72** Değişkenlere klasörlerin tanımlanması

VisDrone veri setinde öznitelik çıkarıcılarına girecek olan sınıflar "insanlar" ve "araçlar sınıfı olarak belirlenmiştir. Bu iki sınıf işleme girmesi için iki farklı değişkene atanmaktadır. Bu iki sınıf hem kendi aralarında hem de birbirleriyle karşılaştırılmıştır. Karşılaştırma için bir dizi içerisine alınmıştır.

```
listeler=[image_folder_car, image_folder_people]
```

**Resim 3. 73** Dizi içerisine alınan değişkenler

VisDrone veri setinden elde edilen verilerin öznitelik çıkarıcılarına girmesi sonucunda, çıktı olarak elde edilen yeni verilerin saklanması için, yeni değişkenler tanımlanır. Değişkenlere, saklanacak veriler için uzantılar tanımlanır. ORB öznitelik çıkarıcıyla işleme girecek verilerin hem sayısal hem de görsel çıktıları için iki farklı değer tanımlanır. Görsel çıktılar, ORB öznitelik çıkarıcısında işleme giren görüntülerin çıktısını oluşturmaktadır. Sayısal çıktılar, anahtar nokta hesaplamalarının sonucunu oluşturmaktadır. İki farklı sınıf, hem kendi içinde hem de birbirleriyle karşılaştırılacağı için, anahtar nokta hesaplamalarının sonuçları, sayıları ve toplamlarını atamak için değişkenler oluşturulur.

```
dst = 'outputdnm\orb/'
dst2 = 'outputdnm\orb\outputtext/'

toplam=0
a=0
b=0
c=0
```

**Resim 3. 74** Çıktıların saklanacağı uzantıların ve anahtar noktaların atanacağı değişkenler

Tanımlama ve saklama işlemleri için gerekli kodlar yazıldıktan sonra görüntülerin sırasıyla işleme girmesi için döngüler oluşturulmuştur. İç içe döngüler kullanılmıştır. İlk döngü dizi içerisindeki değişkenleri döngü içerisine almaktadır. Bu değişkenlerde, görüntülerin içerisinde bulunduğu klasörlerin konumları bulunmaktadır. İkinci döngü ise görüntüler döngü içerisine alınır. İç içe döngülerle yazılım, ilk sınıfı ORB özniteliğinde kendi içinde karşılaştırdıktan sonra diğer sınıfla karşılaştırır. Görüntü, ORB öznitelik çıkarıcı için uygun hale getirip öznitelik çıkarma işlemine başlamıştır.

```
for liste in listeler:
    for img in liste:
        print(liste)
        img=cv2.imread(img, cv2.IMREAD_GRAYSCALE)
        orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
```

**Resim 3. 75** İç içe döngüler ve ORB öznitelik çıkarıcısına girecek görüntünün oranlanması

Yeniden boyutlandırılan görüntü, ORB öznitelik çıkarıcısına girdi olarak alınır. Görüntünün ölçek alan ve anahtar noktaları hesaplanır.

```
kp1, des1 = orbb.detectAndCompute(img, None)
brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches=brute_force.match(des1)
matches=sorted(matches, key=lambda x:x.distance)
matching_results=cv2.drawKeypoints(img,
                    kp1,cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

**Resim 3. 76** ORB öznitelik çıkarıcı komutları

Elde edilen yeni görüntünün ekranda görüntülenmesi “imshow”, saklanması için “imsave” komutları kullanılmıştır. “imsave” komutuna sırasıyla görüntünün bilgisayarda saklanacağı konum, dosya adı, dosyanın uzantısı ve saklanılacak dosya şeklinde parametreler eklenir. Döngü içerisinde kaydetme işlemi sürekli olacağından dolayı dosya adı sürekli değişken olarak belirlenir.

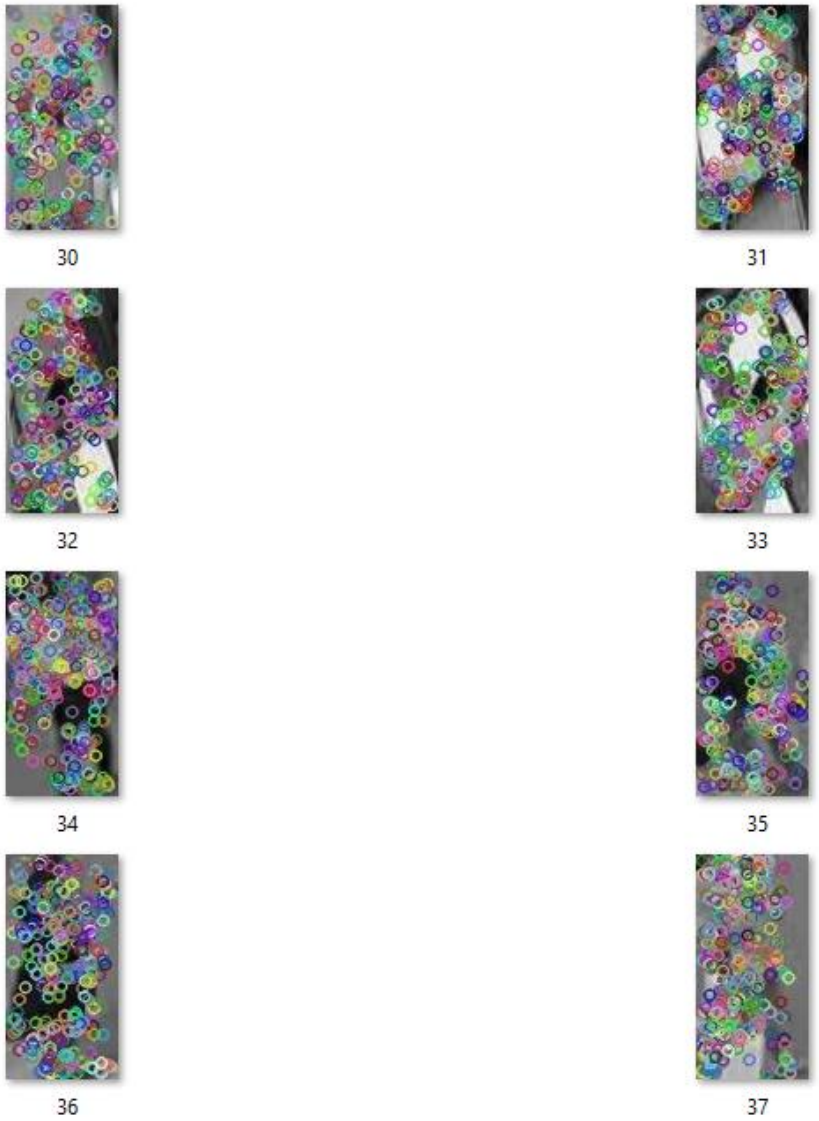
```
plt.axis("on")
plt.imsave(dst+str(c)+".jpg",matching_results)
c+=1
```

**Resim 3. 77** Elde edilen görüntüyü saklama ve ekranda gösterme komutları

ORB öznitelik çıkarıcılarından elde edilen görüntüler, anahtar noktalar ve yönelimleri Resim 3.78’ de görüldüğü gibidir.

**Resim 3. 78** ORB öznitelik çıkarıcılarından elde edilen görüntü

ORB öznitelik çıkarıcıdan elde edilen görüntüler yerel bilgisayardaki klasörlerde, resim 3.79' da görüldüğü gibi saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir.




**Resim 3. 79** ORB öznitelik çıkarıcıdan elde edilen görntülerin saklanması

Elde edilen veriler doğrultusunda gradyan değerleri ve görseller, var olan döngü içerisinde tekrar iç içe döngü ile işleme alınır. Burada amaç, her bir görüntünün işlenmesinin yanı sıra, her bir görüntünün birbirleriyle karşılaştırılmasıdır. Bu karşılaştırma sonucu aynı sınıfa ait olan değerlerin "sınıf içi", farklı sınıflara ait olan değerlerin "sınıflar arası" farkının hesaplanması amaçlanmıştır. Her bir öznitelik çıkarıcı için bu işlem uygulanmıştır. Tekrar oluşturduğumuz döngümüzde, bir önceki işlemler

tekrarlanmaktadır. Farklı olarak bu komutlar arasında elde edilen sonuçların görüntülenmesi ve saklanması önceki komutlarda elde edildiği için, bu iç içe döngüde görüntülenme ve saklama yer almamıştır. İlk döngüde olduğu gibi aynı işlemler ikinci döngüde de tekrarlanır. Görüntü ORB öznitelik çıkarıcıda işleme alınır.

```
for liste2 in listeler:
    print(liste2)
    for img2 in liste2:
        img2=cv2.imread(img2, cv2.IMREAD_GRAYSCALE)
        orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
        kp2, des2 = orbb.detectAndCompute(img2, None)
        brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches=brute_force.match(des1, des2)
        matches=sorted(matches, key=lambda x:x.distance)
```

**Resim 3. 80** İç içe döngü içerisinde görüntünün ORB öznitelik çıkarıcı tarafından işlenmesi

İkinci fotoğrafın öznitelik çıkarma işlemi ile birlikte iki adet görsel çıktı elde edilmiştir. Elde edilen görüntüler aynı sınıfta ise "sınıf içi uzaklık" farklı sınıfta ise "sınıflar arası uzaklık" değerlerini vermektedir. Bu değerlere ulaşmak için elde edilen gradyan değerlerinden matris değerleri oluşturulur. İki görselin matris değerleri arasındaki fark elde edilmiştir. elde edilen değerim normu alınıp değerlendirme için kullanılmıştır. Resim 3.81' de bu matris hesaplaması ve norm dönüşümü verilmiştir. Olası bir hata ile karşılaşılması ve çalışan yazılımın aksamasını önlemek için "try, except" parametreleri kullanılmıştır.

```
matching_results=cv2.drawMatches(img, kp1, img2,
                                 kp2, matches, None)

try:

    tempFd=len(des1)-len(des2)
    tempNorm1=LA.norm(tempFd)

    tempFd2=len(img)-len(img2)
    tempNorm2=LA.norm(tempFd2)
    toplam +=tempNorm1
    a+=1
    print(a,tempNorm1,toplam)

except:
    print(a,"no")
```

**Resim 3. 81** Gradyan değerlerinin matrise dönüştürülüp farklarının hesaplanması

ORB öznitelik çıkarıcı ile elde edilen değerler, diğer bütün görsellerle matris farkı bulmak için karşılaştırılır. Bu işlem tamamlandıktan sonra bir sonraki görsel aynı işlemi tekrar eder. Karşılaştırma işlemi bittiğinde elde edilen değerlerin ortalaması alınmaktadır. Görselin kendi sınıfına  ve diğer sınıflara ait bir ortalaması alınmıştır. Kendi sınıfına ait ortalamalar sınıf içi uzaklıkta diğer sınıflara ağit ortalamalar sınıflar arası uzaklıkta kullanılmıştır. Resim 3.82' de ortalama işleminin kodları verilmiştir.

```
try:

    ortalama=toplam/a

except:
    print("hata")
```

**Resim 3. 82** Görselin kendi ve diğer sınıflara ait ortalama hesaplama işlemi

Ortalama değerleri görselden görsele farklılık göstermektedir. her görselin kendine ait değerleri elde edilir. Bu değerler metin belgesi (.txt) uzantılı dosyalarda saklanır. Ortalamalar sınıflara göre ayrılır. Aynı sınıfa ait ortalamalar ile farklı sınıflara ait ortalamalar, farklı dosyalarda saklanır. Bu ayrım "if" komutu ile sağlanmıştır. Resim 3.83' te farklı dosyalarda saklama işlemi gösterilmiştir. Metin belgesi dosyasının içerisine değerler satırlar halinde otomatik olarak atanmaktadır.

```
if liste==liste2:
    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"visdroneorbsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
else:
    print("nesneler arası uzaklık",liste,ortalama)
    dosya=open(dst2+"visdroneorbsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

**Resim 3. 83** Ortalamaların ayrılıştırılma ve saklanması için yazılan kodlar

Ortalaması alınan değerler sınıflara göre ayrılıp kaydedilmiştir. kayıt için önce dosyanın bulunduğu konum, saklanacak dosyanın adı, uzantısı ve yapılacak işlemin tanımlanması (açma, değiştirme, silme vb.) için gerekli kodlamalar yapılır. Bu çalışmada metin belgesine sürekli veri girişi olacağı için "a" komutu kullanılmıştır. Dosyaya yazma işlemi

için "writelines" komutu kullanılmıştır. Elde edilen ortalama değerlerinin dosya içerisinde alt alta eklenebilmesi için "\n" komutu eklenerek Resim 3.84' te olduğu gibi değerler elde edilmiştir.

| | |
|---|---|
| 0.75<br>1.25<br>0.75<br>0.75<br>3.25<br>4.25<br>5.75<br>3.25 | 3.25<br>3.25<br>3.75<br>3.75<br>3.25<br>5.25<br>4.75<br>0.75 |
| VisDrone ORB öznitelik çıkarıcı sınıf içi değerlerin ortalamaları | VisDrone ORB öznitelik çıkarıcı sınıflar arası değerlerin ortalamaları |

**Resim 3. 84** VisDrone veri setine ait ORB öznitelik çıkarıcıdan elde edilen değerler

ORB öznitelik çıkarma yöntemi için yazılan kodların tamamı Ek 7' de yer almaktadır. kodlar VisDrone veri seti için kullanılmıştır.

### 3.6.1.3.2 ORB Öznitelik Çıkarıcının MNIST Veri Setinde Kullanımı

SIFT öznitelik çıkarıcı, VisDrone veri seti yanı sıra MNIST veri setinde de kullanılmıştır. MNIST veri seti 0-9 arası rakamların farklı el yazılarıyla yazılmış, 28*28 pixel boyutuna sahip görüntülerden oluşmaktadır. MNIST veri setinin, ORB öznitelik çıkarma yöntemi, VisDrone veri setinde kullanılan yönteme benzer komutlardan oluşmaktadır. Aynı komutlar kullanılmıştır. Farklı olarak girdi olarak MNIST veri setinden görseller alınmıştır. Çıktı olarak yine MNIST veri seti farklı bir uzantıda saklanmıştır.

```
image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")
image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")
image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")
image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")
image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")
image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")
image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")
image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")
image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")
image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

**Resim 3. 85** MNIST veri seti dosya uzantılarının değişkenlere atanması

Resim 3.85’ te SIFT öznitelik çıkarıcı için, MNIST veri seti kullanılmıştır. 10 sınıftan oluşan MNIST veri setinin her sınıfı ORB öznitelik çıkarıcı ile işleme alınmıştır. Değişkenlere atanan her sınıf , “listler” dizisi içerisinde bir grup haline getirilmiştir. Resim 3.86’ da dizi içerisinde değişkenler gösterilmiştir.

```
listeler=[image_folder_zerooo,image_folder_one,image_folder_two,
          image_folder_three,image_folder_four,
         image_folder_five,image_folder_six,image_folder_seven,
         image_folder_eight,image_folder_nine]
```

**Resim 3. 86** Dizi içerisinde MNIST veri seti

VisDrone veri setinde kullanılan komutlar, MNIST veri setinde SIFT yöntemi uygulamak için de kullanılmıştır. Döngüler, koşullar, öznitelik çıkarna komutlarında herhangi bir değişiklik yapılmamıştır. Çıktı verilerinin saklanması için, Resim 3.87’ de görülen komutlardan faydalanılmıştır.

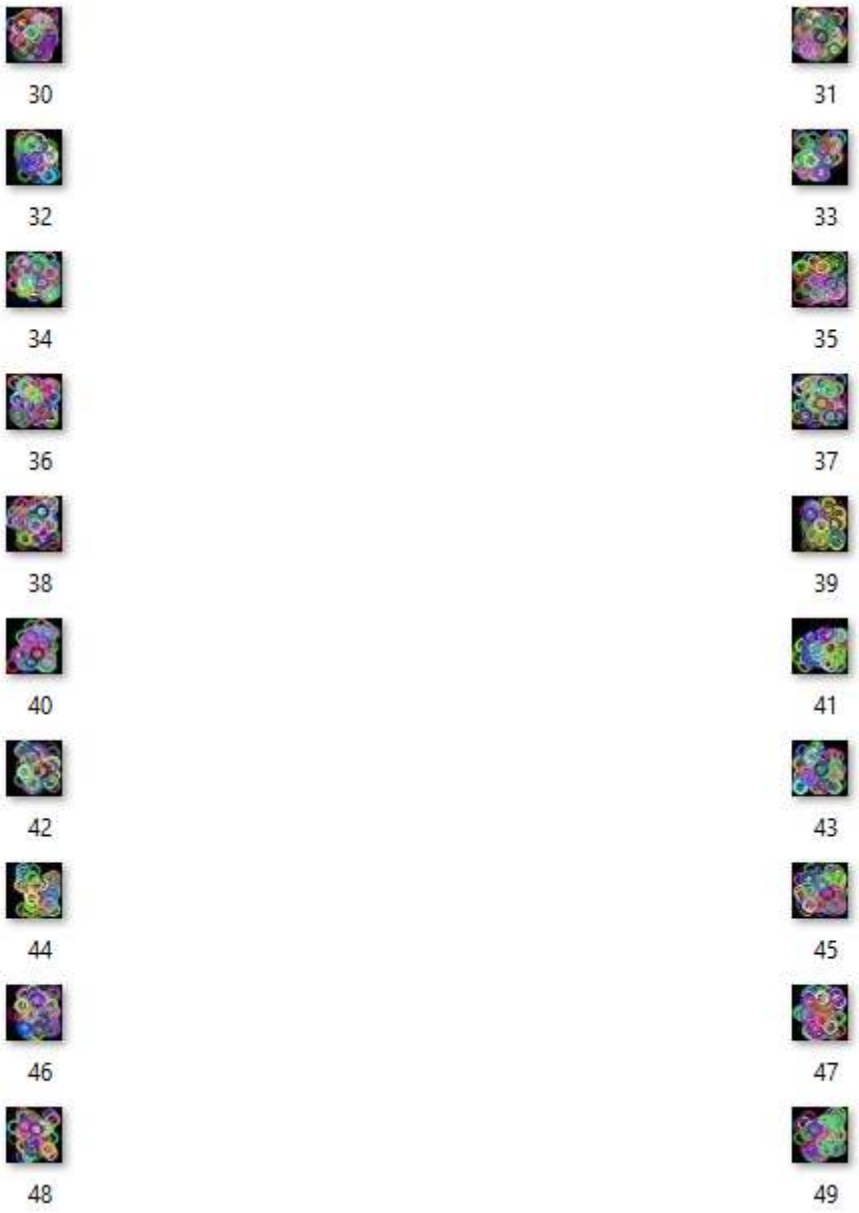
```
dst = 'outputdnm/orb/'
dst2 = 'outputdnm\orb\outputtext/'
```

**Resim 3. 87** MNIST veri seti, ORB yöntemi çıktı dosyalarının saklanması

VisDrone veri setinde olduğu gibi, ORB öznitelik çıkarıcılarından elde edilen görüntüler, 64*128 boyutlarında anahtar nolktalar Resim 3.88’ de görüldüğü gibidir.

**Resim 3. 88** ORB öznitelik çıkarıcılarından elde edilen görüntü

VisDrone veri setinde olduğu gibi, ORB öznitelik çıkarıcıdan elde edilen görüntüler Resim 3.89’ da görüldüğü gibi klasör içerisinde saklanmaktadır. Her bir görüntü öznitelik çıkarıcılardan elde edilmektedir.




**Resim 3. 89** ORB öznitelik çıkarıcıdan elde edilen görntülerin saklanması

ORB öznitelik çıkarma yöntemi kullanılarak MNIST veri seti işlenmiştir. Görsellerle beraber MNIST veri setinde bulunan 10 sınıf için, sınıf içi ve sınıflar arası uzaklıklar hesaplanmıştır. Aynı sınıfta olan görseller için yapılan hesaplamalar sınıf içi uzaklık olarak tanımlanmıştır. Farklı sınıfta olan görseller için yapılan hesaplamalar sınıflar arası uzaklık olarak tanımlanmıştır.

| | |
|---|---|
| 4.0<br>4.0<br>6.5<br>6.5<br>3.0<br>3.0<br>5.0<br>5.0<br>5.5<br>5.5 | 24.0<br>17.5<br>11.0<br>25.0<br>25.5<br>19.5<br>19.5<br>15.0<br>23.5<br>24.0 |
| MNIST ORB öznitelik çıkarıcı sınıf içi değerlerin ortalamaları | MNIST ORB öznitelik çıkarıcı sınıflar arası değerlerin ortalamaları |

**Resim 3. 90** MNIST veri setine ait ORB öznitelik çıkarıcıdan elde edilen değerler

Resim 3.90' da MNIST veri setinin sınıf içi ve sınıflar arası uzaklıklarının hesaplamaları verilmiştir. ORB öznitelik çıkarma yöntemi için yazılan kodların tamamı Ek 8' de yer almaktadır. kodlar MNIST veri seti için kullanılmıştır.

### 3.7 Fisher Ayırt Edici (Fisher's Linear Discriminant)

İngiliz istatikçi Ronald A. Fisher tarafından geliştirilen Fisher ayırtacı, sınıflandırma problemlerinde yaygın olarak kullanılan bir yöntemdir. Literatürde Fisher'ın Doğrusal Ayırt Edicisi (Fisher's Linear Discriminant) olarak bilinmektedir (Fisher 1936). Fisher ayırtacı sınıflar arasındaki farkı en iyi şekilde temsil etmeyi amaçlamaktadır. Bu amaç doğrultusunda sınıflama işlemi yapılırken sınıf içi ve sınıflar arası farkın oranına bakılmaktadır (Çakmak 2015). Temel prensip olarak aynı sınıftan olan verilerin arasındaki farkın en aza indirilmesi hedeflenir ve benzerlikler birbirine daha yakın olur. Bu duruma sınıf içi dağılımın minimizasyonu denmektedir. Farklı sınıftaki verilerde durum tam tersine dönmektedir. Farklı sınıflarda sınıflar arası verilerin birbirine en uzak olanı belirlenmeye çalışılır. Bu duruma da sınıflar arası dağılımın maksimizasyonu denir. En yakın sınıf içi uzaklık ve en uzak sınıflar arası uzaklık oranına bakılarak elde edilen verilerdir. Fisher ayırtacı formülü aşağıda verilmiştir.

$$\text{Öznitelik Kalitesi} = \frac{Sınıflar\ Arası\ Uzaklık\ Ortalaması}{Sınıf\ İçi\ Uzaklık\ Ortalaması} \tag{3.7}$$

Formül: Öznitelik kalitesi formulü

# 4. BULGULAR

VisDrone veri setinde üç popüler öznitelik çıkarma yöntemlerinden elde edilen sonuçlar Fisher ayırtacına girmiştir. Oranlama yöntemi olarak sınıflar arası ve sınıf içi uzaklıklar ele alınmıştır. Aynı sınıfta olan öznitelikler kendi içlerinde ve sonrasında farklı sınıfların öznitelikleri ile karşılaştırılmıştır. Farklı veri setlerinde farklı öznitelik çıkarıcıların öznitelik kalitesinin yüksek olması, bu alanda teknolojinin gelişime açık olduğunu göstermektedir. VisDrone veri setinde arabalar ve insanlar olarak iki sınıf ele alınmıştır. Sınıf içi uzaklık olarak "insan – insan" ve "araba – araba" sınıfları kullanılmıştır. Sınıflar arası uzaklıkta "insan-araba" ve "araba-insan" sınıfları kullanılmıştır. Üç öznitelik çıkarma yöntemine uygulanan yöntemde en yüksek performansı veren öznitelik en iyi öznitelik olarak belirlenmiştir.Elde edilen sonuçlar tablolar halinde verilmiştir.

| Uzaklıklar | Sınıf İçi Uzaklık | | Sınıflar Arası Uzaklık | | | |
|---|---|---|---|---|---|---|
| Sınıflar / Öznitelikler | Araba1-Araba2 | İnsan1-İnsan2 | Araba1- İnsan1 | Araba1– İnsan2 | İnsan1- Araba1 | İnsan1- Araba2 |
| HOG | 42,438 | 40,033 | 43,325 | 44,219 | 43,435 | 43,970 |
| SIFT | 27,725 | 19,584 | 22,677 | 28,623 | 28,417 | 24,734 |
| ORB | 8,211 | 7,286 | 7,945 | 7,963 | 8,929 | 6,527 |

**Tablo 4.1** VisDrone veri setinden elde edilen sınıf içi ve sınıflar arası değerler

Tablo 3.1' de, girdi olarak girilen görüntülerin, her öznitelik çıkarıcı için sonuçları ortalamalar halinde verilmiştir. Elde edilen sonuçlara göre, arabalar sınıfı kendi içinde HOG öznitelik çıkarıcısından ortalama 42,438 sonucu elde edilmiştir. Aynı öznitelik çıkarıcıda İnsanlar sınıfı ortalaması 40,033 olarak elde edilmiştir. HOG öznitelik çıkarıcısında sınıflar arası uzaklıkların ortalaması sırasıyla 43,325 – 44,219 – 43,435 – 43,970 şeklindedir. SIFT öznitelik çıkarıcısında ise arabalar sınıfı 27,725 sonucunu vermiştir. İnsanlar sınıfından 19,584 sonucu elde edilmiştir. Sınıflar arası uzaklıklar ise sırasıyla 22,677 – 28,623 – 28,417 – 24,734 sonuçları elde edilmiştir. ORB öznitelik çıkarıcısı arabalar sınıfında 8,211 sonucunu vermiştir. İnsanlar sınıfında 7,286 sonucu elde edilmiştir. Sınıflar arası uzaklıkların ortalaması ise ORB öznitelik çıkarıcısında sırasıyla 7,945 – 7,963 – 8,929 – 6,527 sonuçları elde edilmiştir. Öznitelik çıkarıcılardan

elde edilen veriler öznitelik kalitesini belirlemek için kullanılmıştır. Sınıflar arası uzakların ortalaması alınarak, sınıf içi uzaklıkların ortalamasına bölünmüştür. Elde edilen sonuçlar özniteliklerin kalitesini vermektedir. Tablo 3.2' de ortalamalar verilmiştir.

| Öznitelikler / Sınıflar | HOG | SIFT | ORB |
|---|---|---|---|
| Sınıflar Arası Uzaklık | 43,740 | 26,110 | 7,840 |
| Sınıf İçi Uzaklık | 41,240 | 23,65 | 7,750 |
| Ortalama | 1,060 | 1,103 | 1,011 |

**Tablo 4.2** VisDrone veri setine ait öznitelik kalitesi hesaplamaları

Tablo 3.2' den elde edilen sonuçlara göre VisDrone veri setinde, HOG öznitelik çıkarıcıdan 1,060 öznitelik kalitesi, SIFT öznitelik çıkarıcıdan 1,103 öznitelik kalitesi, ORB öznitelik çıkarıcıdan 1,011 öznitelik kalitesi sonucuna varılmıştır. VisDrone veri setinde yapılan işlemler MNIST veri seti içinde uygulanmıştır. MNIST veri setinde on adet sınıf işleme alınmıştır. Sıfırdan dokuza kadar olan her rakam birer sınıf olarak ele alınmıştır. Bu durumda her sınıf önce kendi içinde uzaklıkları alınarak sınıf içi uzaklıkları, daha sonrasında diğer dokuz sınıf ile mesafeleri karşılaştırılarak sınıflar arası uzaklıkları oranlanmıştır. "sıfır – sıfır", "bir-bir", "iki-iki" sınıf içi uzaklık, "sıfır-bir", "sıfır-iki", "sıfır-üç", "bir-sıfır", "bir-iki", "bir-üç" şeklinde sınıflar arası uzaklıklar hesaplanmıştır. Tablo 3.3' te sınıf içi uzaklıklardan elde edilen sonuçlar gösterilmiştir.

| Sınıf İçi Uzaklıklar | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sınıflar / Öznitelikler | 0-0 | 1-1 | 2-2 | 3-3 | 4-4 | 5-5 | 6-6 | 7-7 | 8-8 | 9-9 |
| HOG | 34,590 | 33,817 | 35,200 | 34,935 | 36,311 | 34,816 | 33,946 | 34,829 | 35,108 | 32,705 |
| SIFT | 2,080 | 2,880 | 4,520 | 4,960 | 7,040 | 4,160 | 2,560 | 3,360 | 4,320 | 2,400 |
| ORB | 15,840 | 9,760 | 8,000 | 5,440 | 7,040 | 4,000 | 11,040 | 9,600 | 6,400 | 5,920 |

**Tablo 4.3** MNIST veri setine ait sınıf içi değerler

Sınıf içi uzaklıkların MNIST veri setindeki öznitelik kalitesi ele alınmıştır. Tablodan elde edilen verilere göre HOG öznitelik çıkarıcı 32 ile 37 arasında değerler almıştır. SIFT öznitelik çıkarıcı 2 ile 8 arasında değerler alımştır. ORB öznitelik çıkarıcı 4 ile 16 arasında değerler almıştır. Elde edilen değerlerin ortalaması sınıf içi uzaklık olarak kabul edilmiştir. Sınıf içi uzaklıkların öznitelik kalitesi belirlenmesinin ardından sınıflar arası uzaklıların öznitelik kalitesi belirlenmiştir. Tablo 3.4' te sınıflar arası uzaklıkların ortalama öznitelik kaliteleri verilmiştir. On sınıfın her biri kendi aralarında uzaklıkları hesaplanmış ve tablo şeklinde verilmiştir.

| Sınıflar Arası Uzaklıklar | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sınıflar / Öznitelikler | 0-1 | 0-2 | 0-3 | 0-4 | 0-5 | 0-6 | 0-7 | 0-8 | 0-9 |
| HOG | 43,513 | 44,047 | 44,127 | 45,272 | 44,205 | 43,314 | 44,533 | 44,805 | 43,454 |
| SIFT | 4,640 | 3,600 | 4,160 | 6,240 | 4,920 | 3,200 | 3,680 | 4,040 | 3,480 |
| ORB | 19,840 | 16,600 | 14,480 | 16,320 | 17,960 | 15,240 | 17,480 | 13,160 | 17,200 |
| | 1-0 | 1-2 | 1-3 | 1-4 | 1-5 | 1-6 | 1-7 | 1-8 | 1-9 |
| HOG | 43,512 | 42,862 | 43,804 | 44,250 | 43,838 | 42,569 | 43,533 | 44,323 | 42,501 |
| SIFT | 4,640 | 6,700 | 5,600 | 6,800 | 3,960 | 3,280 | 3,600 | 5,800 | 2,920 |
| ORB | 19,840 | 10,600 | 13,760 | 10,160 | 8,040 | 14,360 | 10,760 | 19,000 | 9,120 |
| | 2-0 | 2-1 | 2-3 | 2-4 | 2-5 | 2-6 | 2-7 | 2-8 | 2-9 |
| HOG | 44,047 | 42,862 | 44,298 | 45,008 | 44,763 | 43,683 | 44,693 | 44,901 | 43,770 |
| SIFT | 4,840 | 6,440 | 5,720 | 7,320 | 6,560 | 5,480 | 5,720 | 5,680 | 5,680 |
| ORB | 16,600 | 10,600 | 8,760 | 7,720 | 7,680 | 10,960 | 9,280 | 13,040 | 7,800 |
| | 3-0 | 3-1 | 3-2 | 3-4 | 3-5 | 3-6 | 3-7 | 3-8 | 3-9 |
| HOG | 44,128 | 43,804 | 44,298 | 45,295 | 44,690 | 43,894 | 44,902 | 44,359 | 43,962 |
| SIFT | 4,160 | 5,600 | 4,800 | 6,880 | 5,720 | 4,640 | 4,960 | 5,240 | 4,840 |
| ORB | 14,480 | 13,760 | 8,760 | 8,480 | 11,000 | 9,480 | 10,760 | 8,280 | 9,600 |

**Tablo 4.3** MNIST veri setine ait sınıflar arası değerler

| | 4-0 | 4-1 | 4-2 | 4-3 | 4-5 | 4-6 | 4-7 | 4-8 | 4-9 |
|---|---|---|---|---|---|---|---|---|---|
| HOG | 45,272 | 44,250 | 45,008 | 45,295 | 45,245 | 44,766 | 44,856 | 45,546 | 43,590 |
| SIFT | 6,240 | 6,800 | 6,600 | 6,880 | 6,920 | 6,240 | 6,560 | 6,520 | 6,200 |
| ORB | 16,320 | 10,160 | 7,720 | 8,480 | 7,000 | 10,600 | 8,840 | 12,840 | 7,120 |
| | 5-0 | 5-1 | 5-2 | 5-3 | 5-4 | 5-6 | 5-7 | 5-8 | 5-9 |
| HOG | 44,205 | 43,838 | 44,763 | 44,690 | 45,245 | 43,877 | 44,927 | 45,284 | 43,568 |
| SIFT | 4,920 | 3,960 | 6,700 | 5,720 | 6,920 | 4,040 | 4,200 | 6,320 | 3,840 |
| ORB | 17,960 | 8,040 | 7,680 | 11,000 | 7,000 | 11,520 | 8,240 | 16,800 | 5,400 |
| | 6-0 | 6-1 | 6-2 | 6-3 | 6-4 | 6-5 | 6-7 | 6-8 | 6-9 |
| HOG | 43,315 | 42,568 | 43,683 | 43,894 | 44,766 | 43,877 | 44,367 | 44,730 | 43,161 |
| SIFT | 3,200 | 3,280 | 5,000 | 4,640 | 6,240 | 4,040 | 3,120 | 4,680 | 2,600 |
| ORB | 15,240 | 14,360 | 10,960 | 9,480 | 10,600 | 11,520 | 12,240 | 10,960 | 11,000 |
| | 7-0 | 7-1 | 7-2 | 7-3 | 7-4 | 7-5 | 7-6 | 7-8 | 7-9 |
| HOG | 44,533 | 43,533 | 44,693 | 44,902 | 44,856 | 44,927 | 44,368 | 44,329 | 42,552 |
| SIFT | 3,680 | 3,600 | 5,400 | 4,960 | 6,560 | 4,200 | 3,120 | 5,240 | 3,080 |
| ORB | 17,480 | 10,760 | 9,280 | 10,760 | 8,840 | 8,240 | 12,240 | 14,960 | 8,680 |
| | 8-0 | 8-1 | 8-2 | 8-3 | 8-4 | 8-5 | 8-6 | 8-7 | 8-9 |
| HOG | 44,804 | 44,323 | 44,900 | 44,359 | 45,546 | 45,284 | 44,729 | 44,329 | 43,522 |
| SIFT | 4,040 | 5,800 | 4,400 | 5,240 | 6,520 | 6,320 | 4,680 | 5,240 | 4,800 |
| ORB | 13,160 | 19,000 | 13,040 | 8,280 | 12,840 | 16,800 | 10,960 | 14,960 | 14,920 |
| | 9-0 | 9-1 | 9-2 | 9-3 | 9-4 | 9-5 | 9-6 | 9-7 | 9-8 |
| HOG | 43,454 | 42,501 | 43,770 | 43,961 | 43,590 | 43,568 | 43,161 | 42,552 | 43,522 |
| SIFT | 3,480 | 2,920 | 5,400 | 4,840 | 6,200 | 3,840 | 2,600 | 3,080 | 4,800 |
| ORB | 17,200 | 9,120 | 7,800 | 9,600 | 7,120 | 5,400 | 11,000 | 8,680 | 14,920 |

**Tablo 4.4 (Devam)** MNIST veri setine ait sınıflar arası değerler

Sınıflar arası uzaklıkların değerleri üç öznitelik çıkarıcı için ayrı ayrı hesaplanmıştır. Fisher ayırt edici için sınıflar arası uzaklıların ortalaması alınmıştır. Sınıflar arası uzaklıkların ortalamasının sınıf içi uzakların ortalamasına bölümünden elde edilen değer

öznitelik kalitesini belirlemiştir. Tablo 3.5’ te elde edilen ortalamalar üç öznitelik içinde verilmiştir.

| Öznitelikler / Sınıflar | HOG | SIFT | ORB |
|---|---|---|---|
| Sınıflar Arası Uzaklık | 44,16638 | 4,977333 | 11,734 |
| Sınıf İçi Uzaklık | 34,62574 | 3,828 | 8,304 |
| Ortalama | 1,275 | 1,300 | 1,413 |

**Tablo 4.4** MNIST veri setine ait öznitelik kaliteleri

# 5. TARTIŞMA ve SONUÇ

Nesne tanıma ve görüntü işleme, bilgisayarlı görü alanında önemli bir teknolojidir. Nesne tespiti ve görüntü işleme için çeşitli yöntemler kullanılmaktadır. Renk, doku, keskinlik ve arka plan çıkarma gibi görüntünün ayırt edici özellikleri nesne tespiti için önemli rol oynamaktadır. Gelişen teknoloji ile nesne tanıma yöntemleri sürekli üzerine koyarak ilerlemektedir. Bu çalışmada kullanılan farklı veri setleri ve farklı öznitelik çıkarıcılar, nesne tanıma yöntemlerinin farklı girdilerde değişiklik gösterdiğini ve uygun nesne tanıma yönteminin kullanım alanına göre değiştiğini göstermektedir. Teknolojinin gelişmesiyle birlikte nesne tanıma görevleri önemli başarılar elde etmiştir. Ancak nesne tanıma hala tam anlamıyla gelişimini tamamlamamıştır. Yeni teknolojiler, bilgisayarla görü işlemlerinde yenilikleri beraberinde getirecektir. Yapay zekanın gelişimi, nesne tanıma görevlerinde önemli olan işlem hızı ve öznitelik kalitesi gibi performansı etkileyen faktörlerin, daha verimli çalışmasına ve istikrarlı sonuçlar elde edilmesine katkı sağlayacaktır. Bu çalışmada nesne tanımada kullanılan 3 popüler öznitelik, 2 veri setinden basit rastgele örneklem yöntemi ile alınan görüntülerle karşılaştırılmıştır. Bu karşılaştıma sonuçları tablo 4.1 ‘ de verilmiştir.

Tabloda elde edilen verilere göre MNIST veri setinde HOG öznitelik çıkarıcının öznitelik kalitesi 1,275 olarak belirlenmiştir. SIFT öznitelik çıkarıcının öznitelik kalitesi 1,300 olarak belirlenmiştir. ORB öznitelik çıkarıcının öznitelik kalitesi ise 1,413 olarak belirlenmiştir. VisDrone veri setinde HOG öznitelik çıkarıcının öznitelik kalitesi 1,060 olarak belirlenmiştir. SIFT öznitelik çıkarıcının öznitelik kalitesi 1,103 olarak belirlenmiştir. ORB öznitelik çıkarıcının öznitelik kalitesi ise 1,011 olarak belirlenmiştir. Her iki veri seti üç öznitelik çıkarıcı ile test edilmiş sonuçlar tablolar halinde verilmiştir. Tablo 4.1’ de HOG, SIFT ve ORB öznitelik çıkarıcılarının VisDrone ve MNIST veri setlerine ait öznitelik kaliteleri verilmiştir.

| Öznitelik Kaliteleri | | | |
|---|---|---|---|
| | HOG | SIFT | ORB |
| VisDrone | 1,060 | 1,103 | 1,011 |
| MNIST | 1,276 | 1,300 | 1,413 |

**Tablo 5.1** Veri setlerine ait öznitelik kaliteleri

Tablodan elde edilen verilere göre VisDrone veri setinde 1,103 öznitelik kalitesi ile en yüksek performansı SIFT öznitelik çıkarıcı vermiştir. MNIST veri setinde 1,413 öznitelik kalitesi ile en iyi performans ORB öznitelik çıkarıcıya aittir.

Tablodan elde edilen sonuçlara göre öznitelikler farklı veri setlerinde farklı sonuçlar verebilmektedir. Öznitelik kalitesi girdi olarak verilen görsellere göre farklılık göstermektedir. Farklılıklar, gelecek çalışmalarda kullanılacak veri setine göre, bu 3 öznitelikten en verimli olanı seçmede ve kullanmada literatüre katkı sağlayacaktır. Çalışmada yer alan öznitelik çıkarma yöntemlerinin performansını, girdi olarak kullanılan görsellerin özellikleri etkilemektedir. Görsellerin görüntü kalitesi, bu performansı etkileyen faktörlerdendir. Örneklem olarak kullanılan farklı veri setleri öznitelik kalitesini etkilemiştir. Farklı veri setlerinin kullanılması çalışmanın daha objektif sonuçlar elde edilmesinde önemli olmuştur. Görüntü işleme ve nesne tanıma yöntemlerinde doğru öznitelik çıkarma, işlem verimliliğini ve doğruluğunu artırır. Gelecek çalışmalarda, kullanılacak öznitelik çıkarma yöntemlerinin sayısı arttırılarak daha geniş kapsamlı bir çalışma gerçekleştirilebilir. Kullanılan özniteliklerlerin yanı sıra, örneklem olarak kullanılan veri setlerinin sayısı arttırılabilir veya bu çalışmada kullanılan veri setleri içerisinde yer alan görsellerin sayısı arttırılabilir. VisDrone veri setinde görsellere nesne sınıflama işlemi uygulanmıştır. Sınıflama yönteminde bazı görsellerin görüntü kalitesinde bozulmalar fazladır. Bu durum çalışmayı olumsuz etkilemektedir. Bu durum göz önüne alındığında, gelecek çalışmalarda öznitelik çıkarma işlemi öncesinde, görüntü kalitesinde aşırı bozukluk olan görsellerin ayrıştırılması için bir yazılım kullanılması daha verimli sonuçlar elde edilmesine katkı sağlayacaktır. Nesne tanıma alanında kullanılan yöntemlerin karşılaştırılması, yeni algoritmaların geliştirilmesine temel oluşturacaktır. Gelecekteki akademik çalışmalarda, farklı öznitelik çıkarma yöntemlerinin performans analizleri üzerine yapılan bu araştırma, gelecekte yapılacak çalışmalara referans sağlayacaktır.

## 6. KAYNAKLAR


Abduljabbar M M, 2022, Certain Smoothing Techniques and Their Applications to İmage Processing, Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans tezi, 40, Isparta.

Adıyaman E, Er A O, 2024, İşlenmesi Zor Malzemelerin Tornalanması İçin En Uygun İşleme Sıcaklıklarının Makine Öğrenmesi İle Belirlenmesi. İmalat Teknolojileri ve Uygulamaları, 5(1), 46-64.

Akçay S, 2023, Digital Twin of UAV: İmage Processing İn Different Virtual Environments, OSTİM Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 80, Ankara.

Alhurmuzi O N I, 2023, Fatigue Detection Of Drivers Using İmage Processing And Artificialintelligence Techniques, Altınbaş Üniversitesi Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, 50, İstanbul.

Almahdi I, 2023, CNN GoogleNet ve AlexNet Mimari Diyabetik Retinopati Görüntü İşleme ve Sınıflandırma İçin Derin Öğrenme, İstanbul Gelişim Üniversitesi, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi, 79, İstanbul.

Altekin F, 2021, Görüntü İşleme Teknikleri ve Evrişimsel Sinir Ağları Kullanarak Yüz İfadesinden Duygu Tespiti, Tekirdağ Namık Kemal Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 140, Tekirdağ,

Aydın Y, Akar F, 2017, Yüz Tanıma Sistemlerinde Yerel Özniteliklerin Kullanılması (Using Local Features in Face Recognition Systems) . Akıllı Sistemlerde Yenilikler ve Uygulamalar (ASYU) Konferansı (pp.53). Antalya, Turkey

Başer B Ö, Yangın M, Sarıdaş E S, 2021, Makine Öğrenmesi Teknikleriyle Diyabet Hastalığının Sınıflandırılması, Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü Dergisi, 25(1), 112-120.

Behbudova S, 2023, Makine Öğrenmesi Algoritmaları İle Kardiyovasküler Hastalıkların Tahmin Edilmesi, İstanbul Üniversitesi-Cerrahpaşa, Lisansüstü Eğitim Enstitüsü, Yüksek Lisans Tezi , 73, İstanbul.

Cao Y, He Z, Wang L, Wang W, Yuan Y, Zhang D, vd. 2021, VisDrone-DET2021: The vision meets drone object detection challenge results, In Proceedings of the IEEE/CVF International conference on computer vision, 2847-2854.

Çakmak Z, 2015, Kümeleme Analizinde Geçerlilik Problemi Ve Kümeleme Sonuçlarının Değerlendirmesi, Dumlupınar Üniversitesi Sosyal Bilimler Dergisi, 3, 187-205.

Çetin F H, 2011, Bir Görüntüdeki Nesnenin Başka Bir Görüntüde Bulunması, Başkent Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 163, Ankara.

Dalkıran İ, Ozan M, 2022, Derin Öğrenme Teknikleri Kullanılarak Borsadaki Hisse Değerlerinin Tahmin Edilmesi, Avrupa Bilim ve Teknoloji Dergisi, 39, 143-148.

Daş R, Polat B, Tuna G, 2019, Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi, Fırat Üniversitesi Mühendislik Bilimleri Dergisi , 31 (2) , 571-581.

Emir B, 2021, Oküler Hastalıkların Sınıflandırılmasında Derin Konvolüsyonel Sinir Ağı Modeli, Eskişehir Osmangazi Üniversitesi, Sağlık Bilimleri Enstitüsü, Doktora Tezi, 220, Eskişehir.

Erbir M A, 2023, Derin Öğrenme Tabanlı Görüntü İşleme ile Nesne Tanıma Yöntemlerinde Başarım Oranı Artırma, Kırıkkale Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 109, Kırıkkale.

Erhandı B, 2020, Derin Öğrenme İle Metin Özetleme, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 65, Sakarya

Fisher R A, 1936, The Use Of Multiple Measurements in Taxonomic Problems. Annals Of Eugenics, 7(2), 179-188.

Gauen K, Dailey R, Laiman J, Zi Y, Asokan N, Lu Y H, vd. 2017, Comparison of Visual Datasets For Machine Learning, in 2017, IEEE International Conference On Information Reuse And Integration, 346-355.

Girshick R, Donahue J, Darrell T, Malik J, 2016, RegionBased Convolutional Networks for Accurate Object Detection and Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 142 – 158.

Güler H, 2021, Denetimli Öğrenme Yöntemleri Kullanılarak Bir Ses Olay Tespit Sisteminin Gerçekleştirilmesi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü,Yüksek Lisans Tezi, 74, Elazığ.

Güzel B, Okatan E, 2022, Yapay Zekânın Değıştırdığı Dınamıkler, 1, Eğitim Yayınevi, 199-224, Konya.

Hanbay K, Üzen H, 2017, Nesne Tespit ve Takip Metotları: Kapsamlı Bir Derleme, Türk Doğa ve Fen Dergisi, 6(2), 40-49.

Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, vd. 2017, Speed/Accuracy Trade-Offs For Modern Convolutional Object Detectors. In Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition, 7310-7311.

Jalil A J, Serttaş S, 2022, Yolo ve SSD ile Çeşitli Koşullarda Nesne Algılaması İçin Makine Öğrenimi Yaklaşımı, In 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies, 872-875.

Karakuş P, Karabörk H, 2014, Surf Algoritması Kullanılarak Uzaktan Algılama Görüntülerinin Geometrik Kaydı, V. Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Sempozyumu, 14-17 Ekim, İstanbul.

Kavuncu S K, 2018, Makine Öğrenmesi ve Derin Öğrenme: Nesne Tanıma Uygulaması, Kırıkkale Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 170, Kırıkkale.

Kaya V, Tuncer S, Baran A, 2020, Derin Öğrenme Yöntemleri Kullanılarak Nesne Tanıma, International Science and Technology Conference, Lefkoşa, 277-287,

Kaynar O, Tuna M F, Görmez, Y, Deveci M A, 2017, Makine Öğrenmesi Yöntemleriyle Müşteri Kaybı Analizi, Cumhuriyet Üniversitesi İktisadi ve İdari Bilimler Dergisi, 18(1), 1-14.

Kocaman K Y, Gökçe C O, 2022, Nesne Tanımada Kullanılan İki Popüler Özniteliğin Karşılaştırılması, Avrupa Bilim ve Teknoloji Dergisi, 45, 85-87.

Koç E, Çalışkan S, Yazıcıoğlu, S A, Demirci U, Kuş Z, 2018, Yapay Sinir Ağları, Kelime Vektörleri ve Derin Öğrenme Uygulamaları, Fatih Sultan Mehmet Vakıf Üniversitesi Aluteam, 1, 1-63

Kuşkapan E, Çodur M K, Çodur M Y, 2022, Türkiye'deki Demiryolu Enerji Tüketiminin Yapay Sinir Ağlari İle Tahmin Edilmesi, Konya Journal of Engineering Sciences, 10(1), 72-84.

Kutuk S, 2012, Tissue Density Classification in Mammographic Images Using Local Features, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 75, İstanbul.

Lecun Y, Bottou L, Bengio Y, Haffner P, 1998, Gradient-Based Learning Applied To Document Recognition, Proceedings of The Ieee, 86(11), 2278-2324.

Lowe D G, 1999, Object Recognition From Local Scale-İnvariant Features, In Proceedings Of The Seventh IEEE İnternational Conference On Computer Vision, 2, 1150-1157.

Öztemel E, 2003, Yapay Sinir Ağları, Papatya Yayınevi, İstanbul, 231.

Partal T, Kahya E, Cığızoğlu K, 2011, Yağış Verilerinin Yapay Sinir Ağları ve Dalgacık Dönüşümü Yöntemleri İle Tahmini, İTÜ Dergisi, 7, 3.

Rublee E, Rabaud V, Konolige K, Bradski G, 2011, Orb: An Efficient Alternative to Sıft or Surf, 2011 International Conference On Computer Vision, 2564-2571.

Sarı Ö F, 2024, Nohut Tohumlarının Makine Öğrenmesi Yöntemleri İle Sınıflandırılması, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 63, Konya.

Shakir M H S, 2022, Bazı Düzgünleştirme Teknikleri ve Görtüntü İşleme Uygulamaları, Çankırı Karatekin Üniveresitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 53, Çankırı.

Şeker E, 2020, Yapay Zeka Tekniklerinin/Uygulamalarının Siber Savunmada Kullanımı, Uluslararası Bilgi Güvenliği Mühendisliği Dergisi, 6(2), 108-115.

O'Shea K, Nash R, 2015, An İntroduction To Convolutional Neural Networks, Arxiv Preprint, Arxiv:1511.08458.

Şimşek E, Özyer B, Tümüklü Özyer G, 2019, Fotokapan Görüntülerinde Yerel Öznitelikler ile Nesne Tespiti, Gümüşhane Üniversitesi Fen Bilimleri Dergisi, 9 (4), 633-644.

Tan F G, Yüksel A S, Aydemir E, Ersoy M, 2021, Derin Öğrenme Teknikleri İle Nesne Tespiti ve Takibi Üzerine Bir İnceleme, Avrupa Bilim ve Teknoloji Dergisi, 25, 159-171.

Toğaçar M, 2021, Harflerden Oluşan Genişletilmiş MNIST Veri Kümesinin Derin Öğrenme Tabanlı Tasarlanmış Sinir Ağı Modeli ile Sınıflandırılması, Çukurova Üniversitesi Mühendislik Fakültesi Dergisi, 36(3), 681-690.

Tosun U, 2019, Comparative Analysis Of The Feature Extraction Performance Of Augmented Reality Algorithms, Cumhuriyet Science Journal, 40(4), 958-966.

Tüfekçi M, Karpat F, 2019, Derin Öğrenme Mimarilerinden Konvolüsyonel Sinir Ağları (CNN) Üzerinde Görüntü İşleme-Sınıflandırma Kabiliyetininin Arttırılmasına Yönelik Yapılan Çalışmaların İncelenmesi, In International Conference on Human-Computer Interaction, Optimization and Robotic Applications, 28-31.

Üncü İ, S, Kayakuş M, Çetinkol S, 2021, ORB Yöntemi İle Oy Tespiti Ve Sayımını Gerçekleştiren Sistemin Tasarımı, International Journal Of Technological Sciences, 13(2), 50-56.

Yanık H, 2022, Görüntü İşleme ile Piksel Çakıştırmaya Dayalı Mesafe Tespit Yöntemi, Tokat Gaziosmanpaşa Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 90, Tokat.

Yıldırım B, Cagıl G, 2020, Bir Montaj Parçasının Derin Öğrenme ve Görüntü İşleme İle Tespiti, Journal Of Intelligent Systems: Theory And Applications, 3(2), 31-37.

Yılmaz İ C, 2022, Gerçek Zamanlı Görüntü İşleme İçin Ultra Hızlı Desen Tabanlı Çizgi Dedektörü, Çukurova Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 96, Adana

Zaidi S S A, Ansari M S, Aslam A, Kanwal N, Asghar M, Lee B, 2022, A Survey Of Modern Deep Learning Based Object Detection Models, Digital Signal Processing, 126, 103514.

Zhiqiang W, Jun L, 2017, A Review Of Object Detection Based On Convolutional Neural Network, In 2017 36th Chinese Control Conference, Temmuz 2017, Çin, 11104-11109.

Zhu P, Wen L, Du D, Bian X, Ling H, Hu Q, vd. 2018, Visdrone-Det2018: The Vision Meets Drone Object Detection İn İmage Challenge Results, In Proceedings Of The European Conference On Computer Vision, 0-0.

Zhu P, Wen L, Du D, 2022, Detection And Tracking Meet Drones Challenge, Transactions on Pattern Analysis & Machine Intelligence, 44, 7380-7399.

# EKLER

**Ek 1.** Txt Change Xml Sınıflandırma

```
import cv2
import os

input_img_folder = 'VisDrone2019-DET-train/images'
input_ann_folder = 'VisDrone2019-DET-train/annotations'
output_ann_folder = 'VisDrone2019-DET-train/annotations_new'
output_img_folder = 'VisDrone2019-DET-train/images_new'

os.makedirs(output_img_folder, exist_ok=True)
os.makedirs(output_ann_folder, exist_ok=True)



image_list = os.listdir(input_img_folder)
annotation_list = os.listdir(input_ann_folder)

label_dict = {
    "0" : "Ignore",
    "1" : "Pedestrian",
    "2" : "People",
    "3" : "Bicycle",
    "4" : "Car",
    "5" : "Van",
    "6" : "Truck",
    "7" : "Tricycle",
    "8" : "Awning-tricycle",
    "9" : "Bus",
    "10" : "Motor",
    "11" : "Others"
}
```

**EK 1. (Devam)** Txt Change Xml Sınıflandırma

```
thickness = 2
color = (255,0,0)
count = 0

def object_string(label, bbox):
    req_str = '''
    <object>
        <name>{}</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>{}</xmin>
            <ymin>{}</ymin>
            <xmax>{}</xmax>
            <ymax>{}</ymax>
        </bndbox>
    </object>
    '''.format(label, bbox[0], bbox[1], bbox[2], bbox[3])
    return req_str

for annotation in annotation_list:

    annotation_path = os.path.join(os.getcwd(), input_ann_folder, annotation)
    xml_annotation = annotation.split('.txt')[0] + '.xml'
    xml_path = os.path.join(os.getcwd(), output_ann_folder, xml_annotation)
    img_file = annotation.split('.txt')[0] + '.jpg'
    img_path = os.path.join(os.getcwd(), input_img_folder, img_file)
    output_img_path = os.path.join(os.getcwd(), output_img_folder, img_file)
    img = cv2.imread(img_path)
    annotation_string_init = '''
```

**EK 1. (Devam)** Txt Change Xml Sınıflandırma

```
<annotation>
    <folder>annotations</folder>
    <filename>{}</filename>
    <path>{}</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>{}</width>
        <height>{}</height>
        <depth>{}</depth>
    </size>
    <segmented>0</segmented>'''.format(img_file, img_path, img.shape[0], img.shape[1], img.shape[2])

    file = open(annotation_path, 'r')
    lines = file.readlines()
    for line in lines:

        new_line = line.strip('\n').split(',')
        new_coords_min = (int(new_line[0]), int(new_line[1]))
        new_coords_max = (int(new_line[0])+int(new_line[2]), int(new_line[1])+int(new_line[3]))
        bbox = (int(new_line[0]), int(new_line[1]), int(new_line[0])+int(new_line[2]), int(new_line[1])+int(new_line[3]))
        label = label_dict.get(new_line[5])
        req_str = object_string(label, bbox)
        annotation_string_init = annotation_string_init + req_str
        cv2.rectangle(img, new_coords_min, new_coords_max, color, thickness)

    #resim=img[new_coords_max: new_coords_min, new_coords_max:bbox]
```

**EK 1. (Devam)** Txt Change Xml Sınıflandırma

```
cv2.imwrite(output_img_path, img)
annotation_string_final = annotation_string_init + '</annotation>'
f = open(xml_path, 'w')
f.write(annotation_string_final)
f.close()
count += 1
print('[INFO] Completed {} image(s) and annotation(s) pair'.format(count))
```

**EK 2.** Annotatin Xml Dosyasından Alınan Verilerle Resimleri Kırparak Klasörleme

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr  7 16:03:46 2023

@author: Kaan Yasin KOCAMAN
"""

from PIL import Image
import ast
import os
import cv2
import os
import glob
import xml.etree.ElementTree as ET

original_file = 'images/' #you images directory
dst = 'resultsss/'


def check_folder_exists(path):
    if not os.path.exists(path):
        try:
            os.makedirs(path)
            print ('create ' + path)
        except OSError as e:
            if e.errno != errno.EEXIST:
                raise




seed_arr = []
for xml_file in glob.glob('annotation-xml/*.xml'): #your xml directory
```

**EK 2. (Devam)** Annotatin Xml Dosyasından Alınan Verilerle Resimleri Kırparak Klasörleme

```
    root = ET.parse(xml_file).getroot()
    filename = root.find('filename').text

    for type_tag in root.findall('size'):
        #file_name = type_tag.find('filename').text
        width = type_tag.find('width').text
        height = type_tag.find('height').text

    for type_tag in root.findall('object'):
        class_name = type_tag.find('name').text
        xmin = type_tag.find('bndbox/xmin').text
        ymin = type_tag.find('bndbox/ymin').text
        xmax = type_tag.find('bndbox/xmax').text
        ymax = type_tag.find('bndbox/ymax').text
        all_list = [filename, width,height,class_name,xmin, ymin, xmax,ymax]

        seed_arr.append(all_list)

seed_arr.sort()
#print(str(len(seed_arr)))
#print(str(seed_arr))


for index, line in enumerate(seed_arr):
    filename = line[0]
    width = line[1]
    height = line[2]
    class_name = line[3]
    xmin = line[4]
    ymin = line[5]
```

```
    xmax = line[6]
```

**EK 2. (Devam)** Annotatin Xml Dosyasından Alınan Verilerle Resimleri Kırparak Klasörleme

```
    ymax = line[7]


#print(len(class_name))



    load_img_path = os.path.join(original_file, filename)
    #save img path

#save img path----------
    save_class_path = os.path.join(dst, class_name)
    check_folder_exists(save_class_path)
    save_img_path = os.path.join(save_class_path, str(index)+'_'+filename)

    img = Image.open(load_img_path)
    crop_img = img.crop((int(xmin) ,int(ymin) ,int(xmax) ,int(ymax)))
    newsize = (64, 128)
    im1 = crop_img.resize(newsize)
    im1.save(save_img_path, 'JPEG')
    print('save ' + save_img_path)
```

**Ek 3.** HOG Features Öznitelik Çıkarımı (VisDrone)

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jul  7 15:49:51 2024

@author: Kaan Yasin KOCAMAN
"""

import glob
from skimage.io import imread
from skimage.transform import resize
from skimage.feature import HOG
import matplotlib.pyplot as plt
from numpy import linalg as LA

image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")

listeler=[image_folder_car, image_folder_people]
listeler2=[image_folder_car, image_folder_people]

dst = 'outputdnm\hog/'
```

```
dst2 = 'outputdnm\hog\outputtext/'
toplam=0
```

**Ek 3. (Devam)** HOG Features Öznitelik Çıkarımı (visDrone)

```
a=0
b=0
c=0

for liste in listeler:
    for img in liste:
        print(liste)
        img = imread(img)
        plt.axis("on")

        resized_img = resize(img, (64*4, 128*4))
        plt.axis("on")

        fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True, channel_axis=-1)
        plt.axis("on")
        plt.imshow(hog_image, cmap="gray")
        plt.imsave(dst+str(c)+".jpg",hog_image,)#cmap="gray")
        c+=1
        for liste2 in listeler:
            print(liste2)
```

```
for img2 in liste2:
    img2 = imread(img2)
```

**Ek 3. (Devam)** HOG Features Öznitelik Çıkarımı (visDrone)

```
    plt.axis("on")
    resized_img2 = resize(img2, (64*4, 128*4))
    plt.axis("on")

    fd2, hog_image2 = hog(resized_img2, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualize=True, channel_axis=-1)
    plt.axis("on")

    try:

        tempFd = fd - fd2
        tempNorm1 = LA.norm(tempFd)
        toplam +=tempNorm1
        a+=1
        b+=1
        print(a,tempNorm1,toplam)
    except:
        print(a,"no")
try:

    ortalama=toplam/a
```

```
except:
    print("hata
```

**Ek 3. (Devam)** HOG Features Öznitelik Çıkarımı (visDrone)

```
print("nesneler içi uzaklık",liste,ortalama)
dosya=open(dst2+"visdronehog.txt","a")
dosya.writelines("sinif "+str(ortalama)+"\n")
dosya.close()

if liste==liste2:

    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronehogsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
else:
    print("nesneler arası uzaklık",liste,ortalama)
    dosya=open(dst2+"visdronehogsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()

a=0
ortalama=0
toplam=0
```

**Ek 4.** HOG Features Öznitelik Çıkarımı (MNIST)

```
# -*- coding: utf-8 -*-

"""

Spyder Editor


This is a temporary script file.

"""

import glob

from skimage.io import imread

from skimage.transform import resize

from skimage.feature import hog

import matplotlib.pyplot as plt

from numpy import linalg as LA


image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")

image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")

image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")

image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")

image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")

image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")

image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")

image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")

image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")
```

```
image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

**Ek 4. (Devam)** HOG Features Öznitelik Çıkarımı (MNIST)

```
listeler=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_three
,image_folder_four,

image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

listeler2=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_thre
e,image_folder_four,

image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

toplam=0

a=0

b=0

dst = 'outputdnm\hog/'

dst2 = 'outputdnm\hog\outputtext/'

for liste in listeler:

    for img in liste:

        print(liste)

        img = imread(img)

        plt.axis("on")

        resized_img = resize(img, (64*4, 128*4))

        plt.axis("on")
```

```
    fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True)

    plt.axis("on")
```

**Ek 4. (Devam)** HOG Features Öznitelik Çıkarımı (MNIST)

```
    plt.imshow(hog_image, cmap="gray")

    plt.imsave(dst+str(b)+".jpg",hog_image, cmap="gray")


    for liste2 in listeler:
        print(liste2)
        for img2 in liste2:
            img2 = imread(img2)
            plt.axis("on")
            resized_img2 = resize(img2, (64*4, 128*4))
            plt.axis("on")
            fd2, hog_image2 = hog(resized_img2, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True)
            plt.axis("on")


            try:


                tempFd = fd - fd2
                tempNorm1 = LA.norm(tempFd)
                toplam +=tempNorm1
                a+=1
                b+=1
```

```
            print(a,tempNorm1,toplam)


        except:
```

**Ek 4. (Devam)** HOG Features Öznitelik Çıkarımı (MNIST)

```
            print(a,"no")


    try:


        ortalama=toplam/a


    except:


        print("hata")


    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"mnisthog.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()


    if liste==liste2:


        print("nesneler içi uzaklık",liste,ortalama)
        dosya=open(dst2+"mnisthogsi.txt","a")
        dosya.writelines("sinif "+str(ortalama)+"\n")
```

```
        dosya.close()


    else:
```

**Ek 4. (Devam)** HOG Features Öznitelik Çıkarımı (MNIST)

```
        print("nesneler arası uzaklık",liste,ortalama)
        dosya=open(dst2+"mnisthogsa.txt","a")
        dosya.writelines("sinif "+str(ortalama)+"\n")
        dosya.close()


    a=0
    ortalama=0
    toplam=0
```

**Ek 5.** SIFT Features Öznitelik Çıkarımı (VisDrone)

```
import cv2
import cv2 as cv
from matplotlib import pyplot as plt
from numpy import linalg as LA
import glob

image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")

listeler=[image_folder_car, image_folder_people]
listeler2=[image_folder_car, image_folder_people]
dst = 'outputdnm/sift/'
dst2 = 'outputdnm\sift\outputtext/'
toplam=0
a=0
b=0
c=0

for liste in listeler:
    for img in liste:
```

```
        print(liste)
        img = cv.imread(img)
```

**Ek 5. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
        gray= cv.cvtColor(img,cv.COLOR_BGR2GRAY)
        sift = cv.SIFT_create()
        kp = sift.detect(gray,None)
        img=cv.drawKeypoints(gray,kp,img)

img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        sift = cv.SIFT_create()
        kp, des = sift.detectAndCompute(gray,None)
        plt.imshow(img, cmap="gray")
        plt.imsave(dst+str(c)+".jpg",img,)#cmap="gray")
        c+=1


        for liste2 in listeler:
            print(liste2)
            for img2 in liste2:


                img2 = cv.imread(img2)
                gray2= cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
                sift2 = cv.SIFT_create()
                kp2 = sift.detect(gray,None)
```

```
            img2=cv.drawKeypoints(gray2,kp2,img2)
            img2=cv.drawKeypoints(gray2,kp2,img2,
```

**Ek 5. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
            sift2 = cv.SIFT_create()
            kp2, des2 = sift2.detectAndCompute(gray2,None)
            bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)

            try:

               matches = bf.match(des,des2)
               matched_img = cv2.drawMatches(img, kp, img2, kp2, matches[:50], img2,
flags=2)
               tempFd=len(kp)-len(kp2)
               tempNorm1=LA.norm(tempFd)
               tempFd2=len(img)-len(img2)
               tempNorm2=LA.norm(tempFd2)
               toplam +=tempNorm1
               a+=1
               print(a,tempNorm1,toplam)

            except:
```

```
            print(a,"no")


        try:
```

**Ek 5. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
            ortalama=toplam/a


        except:


            print("hata")


        print("nesneler içi uzaklık",liste,ortalama)
        dosya=open(dst2+"visdronesift.txt","a")
        dosya.writelines("sinif "+str(ortalama)+"\n")
        dosya.close()


        if liste==liste2:


            print("nesneler içi uzaklık",liste,ortalama)
            dosya=open(dst2+"visdronesiftsi.txt","a")
            dosya.writelines("sinif "+str(ortalama)+"\n")
            dosya.close()


        else:
```

```
        print("nesneler arası uzaklık",liste,ortalama)
        dosya=open(dst2+"visdronesiftsa.txt","a")
```

**Ek 5. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
        dosya.writelines("sinif "+str(ortalama)+"\n")
        dosya.close()


    a=0
    ortalama=0
    toplam=0
```

**EK 6.** SIFT Features Öznitelik Çıkarımı (MNIST)

```
# -*- coding: utf-8 -*-

"""

Created on Sun Jul 21 13:19:58 2024


@author: Kaan Yasin KOCAMAN

"""


import cv2

import glob

from numpy import linalg as LA

import cv2 as cv

from matplotlib import pyplot as plt



image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")

image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")

image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")

image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")

image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")

image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")

image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")
```

```
image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")

image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")

image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

**EK 6. (Devam)** SIFT features öznitelik çıkarımı (MNIST)

```
folder_dokuz=glob.glob("trainingSet\9/*.jpg")


listeler=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_three
,image_folder_four,

image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

listeler2=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_thre
e,image_folder_four,

image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

dst = 'outputdnm/sift/'

dst2 = 'outputdnm\sift\outputtext/'

toplam=0

a=0

b=0

c=0


for liste in listeler:
```

```
    for img in liste:
        print(liste)
        img = cv.imread(img)
```

**EK 6. (Devam)** SIFT features öznitelik çıkarımı (MNIST)

```
        gray= cv.cvtColor(img,cv.COLOR_BGR2GRAY)
        sift = cv.SIFT_create()
        kp = sift.detect(gray,None)
        img=cv.drawKeypoints(gray,kp,img)

img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        sift = cv.SIFT_create()
        kp, des = sift.detectAndCompute(gray,None)
        plt.imshow(img, cmap="gray")
        plt.imsave(dst+str(c)+".jpg",img,)#cmap="gray")
        c+=1


        for liste2 in listeler:
            print(liste2)
            for img2 in liste2:
                img2 = cv.imread(img2)
                gray2= cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
                sift2 = cv.SIFT_create()
                kp2 = sift.detect(gray,None)
                img2=cv.drawKeypoints(gray2,kp2,img2)
```

```
            img2=cv.drawKeypoints(gray2,kp2,img2,

flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

**EK 6. (Devam)** SIFT features öznitelik çıkarımı (MNIST)

```
            sift2 = cv.SIFT_create()
            kp2, des2 = sift2.detectAndCompute(gray2,None)
            bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)

            try:

               matches = bf.match(des,des2)
               matched_img = cv2.drawMatches(img, kp, img2, kp2, matches[:50], img2,
flags=2)
               tempFd=len(kp)-len(kp2)
               tempNorm1=LA.norm(tempFd)
               tempFd2=len(img)-len(img2)
               tempNorm2=LA.norm(tempFd2)
               toplam +=tempNorm1
               a+=1
               print(a,tempNorm1,toplam)

            except:

               print(a,"no")
```

```
try:
```

**EK 6. (Devam)** SIFT features öznitelik çıkarımı (MNIST)

```
    ortalama=toplam/a


except:
    print("hata")



print("nesneler içi uzaklık",liste,ortalama)
dosya=open(dst2+"mnistsift.txt","a")
dosya.writelines("sinif "+str(ortalama)+"\n")
dosya.close()

if liste==liste2:

    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"mnistsiftsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()

else:
    print("nesneler arası uzaklık",liste,ortalama)
```

```
    dosya=open(dst2+"mnistsiftsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

**EK 6. (Devam)** SIFT features öznitelik çıkarımı (MNIST)

```
a=0
ortalama=0
toplam=0
```

**Ek 7.** ORB Features Öznitelik Çıkarımı (VisDrone)

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 22 22:40:59 2024

@author: Kaan Yasin KOCAMAN
"""
import matplotlib.pyplot as plt
from numpy import linalg as LA
import cv2
import glob

image_folder_car = glob.glob("results\denemee\AArac/*.jpg")
image_folder_people= glob.glob("results\denemee\insan/*.jpg")

listeler=[image_folder_car, image_folder_people]
listeler2=[image_folder_car, image_folder_people]
dst = 'outputdnm\orb/'
dst2 = 'outputdnm\orb\outputtext/'
toplam=0
a=0
```

```
b=0
c=30
```

**Ek 7. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
for liste in listeler:
    for img in liste:
        print(liste)
        img=cv2.imread(img, cv2.IMREAD_GRAYSCALE)
        orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
        kp1, des1 = orbb.detectAndCompute(img, None)
        brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches=brute_force.match(des1)
        matches=sorted(matches, key=lambda x:x.distance)

matching_results=cv2.drawKeypoints(img,kp1,cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        plt.axis("on")
        plt.imsave(dst+str(c)+".jpg",matching_results)  #cmap="gray")
        c+=1

        for liste2 in listeler:
            print(liste2)
            for img2 in liste2:
                img2=cv2.imread(img2, cv2.IMREAD_GRAYSCALE)
                orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
```

```
        kp2, des2 = orbb.detectAndCompute(img2, None)
        brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches=brute_force.match(des1, des2)
```

**Ek 7. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
        matches=sorted(matches, key=lambda x:x.distance)
        matching_results=cv2.drawMatches(img, kp1, img2, kp2, matches, None)

        try:

            tempFd=len(des1)-len(des2)
            tempNorm1=LA.norm(tempFd)
            tempFd2=len(img)-len(img2)
            tempNorm2=LA.norm(tempFd2)
            toplam +=tempNorm1
            a+=1
            print(a,tempNorm1,toplam)

        except:

            print(a,"no")

    try:

        ortalama=toplam/a
```

```
except:
    print("hata")
```

**Ek 7. (Devam)** SIFT Features Öznitelik Çıkarımı (visDrone)

```
print("nesneler içi uzaklık",liste,ortalama)
dosya=open(dst2+"visdroneorb.txt","a")
dosya.writelines("sinif "+str(ortalama)+"\n")
dosya.close()

if liste==liste2:

    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"visdroneorbsi.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()

else:

    print("nesneler arası uzaklık",liste,ortalama)
    dosya=open(dst2+"visdroneorbsa.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

```
    a=0
    ortalama=0
    toplam=0
```

**Ek 8.** ORB Features Öznitelik Çıkarımı (MNIST)

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 22 22:40:59 2024

@author: Kaan Yasin KOCAMAN
"""
import matplotlib.pyplot as plt
from numpy import linalg as LA
import cv2
import glob
image_folder_zerooo = glob.glob("trainingSet\yyy2\zeroo/*.jpg")
image_folder_one = glob.glob("trainingSet\yyy2\one/*.jpg")
image_folder_two= glob.glob("trainingSet\yyy2\TWO/*.jpg")
image_folder_three = glob.glob("trainingSet\yyy2\THRE/*.jpg")
image_folder_four = glob.glob("trainingSet\yyy2\drt/*.jpg")
image_folder_five = glob.glob("trainingSet\yyy2\BS/*.jpg")
image_folder_six = glob.glob("trainingSet\yyy2\six/*.jpg")
image_folder_seven= glob.glob("trainingSet\yyy2\seven/*.jpg")
image_folder_eight = glob.glob("trainingSet\yyy2\eight/*.jpg")
image_folder_nine = glob.glob("trainingSet\yyy2\dkz/*.jpg")
```

```
listeler=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_three
,image_folder_four,
```

**Ek 8. (Devam)** SIFT Features Öznitelik Çıkarımı (MNIST)

```
image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

listeler2=[image_folder_zerooo,image_folder_one,image_folder_two,image_folder_thre
e,image_folder_four,


image_folder_five,image_folder_six,image_folder_seven,image_folder_eight,image_fol
der_nine]

dst = 'outputdnm/orb/'

dst2 = 'outputdnm\orb\outputtext/'

toplam=0

a=0

b=0

c=30



for liste in listeler:
    for img in liste:
        print(liste)
        img=cv2.imread(img, cv2.IMREAD_GRAYSCALE)
        orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
        kp1, des1 = orbb.detectAndCompute(img, None)
        brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
```

```
matches=brute_force.match(des1)
matches=sorted(matches, key=lambda x:x.distance)
```

**Ek 8. (Devam)** SIFT Features Öznitelik Çıkarımı (MNIST)

```
matching_results=cv2.drawKeypoints(img,kp1,cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
    plt.axis("on")
    plt.imsave(dst+str(c)+".jpg",matching_results)  #cmap="gray")
    c+=1

    for liste2 in listeler:
        print(liste2)
        for img2 in liste2:
            img2=cv2.imread(img2, cv2.IMREAD_GRAYSCALE)
            orbb=cv2.ORB_create(fastThreshold=0, edgeThreshold=0)
            kp2, des2 = orbb.detectAndCompute(img2, None)
            brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            matches=brute_force.match(des1, des2)
            matches=sorted(matches, key=lambda x:x.distance)
            matching_results=cv2.drawMatches(img, kp1, img2, kp2, matches, None)

            try:

                tempFd=len(des1)-len(des2)
```

```
            tempNorm1=LA.norm(tempFd)
            tempFd2=len(img)-len(img2)
            tempNorm2=LA.norm(tempFd2)
```

**Ek 8. (Devam)** SIFT Features Öznitelik Çıkarımı (MNIST)

```
            toplam +=tempNorm1
            a+=1
            print(a,tempNorm1,toplam)

        except:

            print(a,"no")
    try:

        ortalama=toplam/a

    except:

        print("hata")

    print("nesneler içi uzaklık",liste,ortalama)
    dosya=open(dst2+"mnistorb.txt","a")
    dosya.writelines("sinif "+str(ortalama)+"\n")
    dosya.close()
```

```
    if liste==liste2:


        print("nesneler içi uzaklık",liste,ortalama)
```

**Ek 8. (Devam)** SIFT Features Öznitelik Çıkarımı (MNIST)

```
        dosya=open(dst2+"mnistorbsi.txt","a")
        dosya.writelines("sinif "+str(ortalama)+"\n")
        dosya.close()


    else:


        print("nesneler arası uzaklık",liste,ortalama)
        dosya=open(dst2+"mnistorbsa.txt","a")
        dosya.writelines("sinif "+str(ortalama)+"\n")
        dosya.close()


    a=0
    ortalama=0
    toplam=0
```