YAPAY ZEKA VE VERİ BİLİMİ BÖLÜMÜ

ÖDEV/PROJE BAŞLIĞI

HOG (Histogram of Oriented Gradients)

Hazırlayan:

Esma Elifsu CERİT

Öğrenci Numarası: 220212003

05 Aralık 2025

YÖNLENDİRİLMİŞ GRADYANLARIN HİSTOGRAMI (HOG) TEMELLİ NESNE TANIMA VE SINIFLANDIRMA PERFORMANSI ANALİZİ

Esma Elifsu Cerit

***Özet*—Bu çalışma, bilgisayarla görü ve nesne tanıma alanında devrim yaratan ve Histogram of Oriented Gradients (HOG) özellik tanımlayıcısının teorik temellerini, uygulama adımlarını ve makine öğrenimi sınıflandırıcılarıyla entegrasyonunu detaylıca incelemektedir. 2005 yılında Dalal ve Triggs tarafından geliştirilen HOG, yerel gradyan yönelimlerinin dağılımını yakalayarak nesnelerin şekil ve görünüş bilgisini aydınlatma ve küçük pozisyon değişikliklerine karşı dayanıklı (robust) bir şekilde karakterize eder. Bu makale, HOG algoritmasının Adım 1: Gradyan Hesaplama'dan Adım 5: Blok Normalizasyonu'na kadar olan tüm aşamalarının teorik arka planını açıklamaktadır. Uygulama kısmında ise, gerçek dünya görüntüleri üzerinden çıkarılan HOG özelliklerinin Destek Vektör Makineleri (SVM) gibi güçlü sınıflandırıcılarla kullanılarak başarılı bir ikili sınıflandırma (köpek tespiti) görevinin nasıl gerçekleştirildiği sunulmaktadır. Elde edilen bulgular, HOG'un nesnelerin şekil bilgisini etkili bir şekilde kodlayarak, özellikle katı (rigid) nesnelerin tespitinde yüksek doğruluk ve güvenilirlik sağladığını kanıtlamaktadır.**

***Index Terms*—HOG, Nesne Tanıma, Destek Vektör Makinesi, Gradyan, Özellik Çıkarımı, Bilgisayarla Görü.**

I. GİRİŞ

Bilgisayarla görü (Computer Vision) alanındaki en temel ve zorlu görevlerden biri, dijital bir görüntüdeki nesneleri güvenilir bir şekilde tespit etmek ve tanımaktır. Başarılı bir nesne tanıma sistemi, gürültü, aydınlatma farklılıkları ve geometrik bozulmalar gibi zorlu koşullara karşı dayanıklı, bilgi açısından yoğun özelliklerin çıkarılmasına dayanır. Bu bağlamda, Yönlendirilmiş Gradyanların Histogramı (Histogram of Oriented Gradients - HOG), 2005 yılından bu yana, özellikle insan tespiti (pedestrian detection) olmak üzere, birçok nesne tanıma problemi için altın standart haline gelmiştir.

HOG, bir nesnenin yerel şekil bilgisinin, görüntüdeki yoğunluk gradyanlarının veya kenar yönelimlerinin dağılımı ile etkili bir şekilde temsil edilebileceği temel fikri üzerine kurulmuştur. Bu özellik tanımlayıcısının temel gücü, görüntüdeki yerel kontrast normalizasyonu sayesinde, hem geometrik hem de fotometrik dönüşümlere karşı yüksek düzeyde dayanıklılık sergilemesidir.

Öğrenci Numarası: 220212003
Kurum/Bölüm Bilgisi buraya eklenecektir.

A. Çalışmanın Kapsamı ve Yapısı

Bu makalenin temel amacı, HOG özellik çıkarım yönteminin derinlemesine bir analizini sunmak, adımlarını teorik olarak açıklamak ve son olarak, çıkarılan bu özelliklerin Makine Öğrenimi (ML) teknikleriyle nasıl entegre edileceğini göstermektir. Çalışma, HOG'un sıfırdan implementasyon hedefini taşımakta olup, gradyan hesaplamadan blok normalizasyonuna kadar her aşamayı kapsar.

Bu bağlamda, makale aşağıdaki öğrenme hedeflerini temel almaktadır:

- **Teorik Temeller:** HOG'un gradyan büyüklüğü, yönelimi ve histogram bin'leme prensipleri.
- **Algoritma Adımları:** Ön İşleme, Gradyan Hesabı, Hücre Tanımlama, Yönelim Histogramı Oluşturma ve kritik Blok Normalizasyonu adımlarının incelenmesi.
- **Uygulamalı Entegrasyon:** Çıkarılan HOG özelliklerinin, Lineer Destek Vektör Makinesi (SVM) gibi sınıflandırıcılarla birleştirilerek görüntü sınıflandırma (örneğin köpek/köpek olmayan tespiti) görevine uygulanması.

II. METODOLOJİ: HOG ÖZELLİK ÇIKARICISININ SIFIRDAN İMPLEMENTASYONU

Bu çalışmada, nesnelerin şekil ve görünüş bilgisini kodlamak amacıyla kullanılan Histogram of Oriented Gradients (HOG) özellik tanımlayıcısı, ticari kütüphane fonksiyonlarına bağımlılık olmadan, temel matematiksel prensiplere sadık kalınarak manuel olarak implemente edilmiştir. Bu yaklaşım, algoritmanın temel çalışma mekanizmasının derinlemesine anlaşılmasını sağlamıştır.

A. HOG Hesaplama Adımları

HOG algoritması, görüntüdeki lokal gradyan bilgisine dayanan altı ana adımdan oluşmaktadır. Her bir adım, özelleştirilmiş bir Python fonksiyonu ile gerçekleştirilmiştir.

1) Gradyan Hesaplama (compute_gradients): Özellik çıkarımının ilk adımı, görüntüdeki dikey ve yatay kenar bilgilerini yakalamaktır. compute_gradients fonksiyonu, bu amaçla Sobel filtrelerine benzer şekilde, $[1, 0, 1]$ ve $[1, 0, 1]^\top$ çekirdeklerini kullanarak, sırasıyla yatay ($\mathbf{G}_x$) ve dikey ($\mathbf{G}_y$) gradyan bileşenlerini hesaplamıştır.

Daha sonra, her piksel için Gradyan Büyüklüğü ($\mathbf{M}$) ve Gradyan Açısı (θ) Öklid normu ve arctan2 fonksiyonları kullanılarak türetilmiştir:

$$\mathbf{M} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \quad \text{ve} \quad \theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Gradyan yönelimleri, HOG'un standart uygulamasına uygun olarak, işaretsiz (unsigned) yönlendirmeyi temsil etmek üzere 0° ile 180° arasına normalize edilmiştir.

2) Hücre Histogramı Oluşturma (create_cell_histogram): Görüntü, $C \times C$ boyutunda (varsayılan olarak 8×8 piksel) küçük, örtüşmeyen hücrelere bölünmüştür. create_cell_histogram fonksiyonu, her hücre içindeki piksellerin gradyan yönelimlerini kullanarak bir yönelim histogramı oluşturmuştur. Histogram, 9 bin'e bölünmüş olup, 0° ile 180° arasındaki açılar bin'lere eşit olarak dağıtılmıştır.

3) Blok Normalizasyonu (normalize_block): Aydınlatma ve kontrast farklılıklarına karşı dayanıklılığı artırmak için, komşu hücreler bloklar halinde gruplandırılmıştır (varsayılan 2×2 hücre). normalize_block fonksiyonu, bu bloğun birleştirilmiş histogram vektörünü normalize etmek için L2-Hys (L2-norm, kırpma ve yeniden L2-norm) yöntemini kullanmıştır. Normalizasyon formülü:

$$\mathbf{h}_{\text{norm}} = \frac{\mathbf{h}}{\sqrt{||\mathbf{h}||^2 + \epsilon^2}}$$

Burada $\mathbf{h}$ blok histogram vektörünü, $||\mathbf{h}||$ L2 normunu ve ϵ küçük bir sabiti (10^{-5}) temsil eder.

4) HOG Özellik Vektörü Oluşturma (compute_hog_descriptor): compute_hog_descriptor fonksiyonu, görüntü üzerinde tek hücre kaymalı (sliding window) bir yaklaşımla tüm blokları gezerek, normalize edilmiş blok histogramlarını ardışık olarak birleştirmiş ve nihai HOG özellik vektörünü oluşturmuştur.

B. HOG Görselleştirme (visualize_hog)

HOG'un yerel yönelim bilgisini anlaşılır kılmak için visualize_hog fonksiyonu implement edilmiştir. Bu fonksiyon, her hücrenin histogram bilgisini, hücre merkezinden yayılan ve bin yönelimlerine karşılık gelen oklarla (çizgilerle) göstermiştir.

C. Deneysel Parametre Analizi

İmplementasyonun sağlamlığını ve farklı parametrelerin etkisini değerlendirmek amacıyla, aynı test görüntüsü üzerinde beş farklı parametre seti kullanılarak HOG vektörleri çıkarılmıştır. Sonuçlar Tablo I'de özetlenmiştir:

III. HOG TEMELLİ NESNE TESPİTİ VE SINIFLANDIRMA

Bu bölümde, HOG özellik çıkarımının iki ana görevdeki uygulaması detaylandırılmıştır: Hazır Model ile İnsan Tespiti ve Özel Nesne Sınıflandırması (Köpek). Her iki görev de, HOG özelliklerini Destek Vektör Makineleri (SVM) ile birleştiren temel bir görüntü işleme hattını kullanmıştır.

Tablo I
HOG PARAMETRELERININ ÖZELLIK VEKTÖR BOYUTUNA ETKISI

Test No	Hücre Boyutu (cell)	Blok Boyutu (block)	Bin Sayısı (bins)	Çıkarı
1	(8, 8)	(2, 2)	9	
2	(16, 16)	(2, 2)	9	
3	(8, 8)	(3, 3)	9	
4	(8, 8)	(2, 2)	6	
5	(4, 4)	(2, 2)	9	

Tablo II
ÖZEL NESNE SINIFLANDIRMASI İÇIN KULLANILAN HOG PARAMETRELERI

Parametre	Değer	Açıklama
Orientations	9	Yönelim bin sayısı.
Pixels per Cell	(8, 8)	Lokal alan boyutu.
Cells per Block	(2, 2)	Normalizasyon bloğu boyutu.
Transform	True (sqrt)	Kare Kök Dönüşümü ile aydınlatma dayanıklılığı.
HOG Vektör Boyutu	**3780**	Final özellik vektör boyutu.

A. Problem 2.1: Önceden Eğitilmiş Model ile İnsan Tespiti

OpenCV kütüphanesinin sağladığı, Navneet Dalal ve Bill Triggs tarafından geliştirilen orijinal HOG + Lineer SVM modeli, görüntü içindeki yayaları tespit etmek için kullanılmıştır.

1) OpenCV HOGDedektör Entegrasyonu: Tespit sistemi, OpenCV'nin HOGDescriptor sınıfı kullanılarak oluşturulmuştur.

2) Çakışan Kutuların Yönetimi (Non-Maximum Suppression - NMS): Çok ölçekli tarama sonucunda, aynı nesne üzerinde çok sayıda çakışan tespit kutusu (bounding box) ve buna karşılık gelen güven puanı (confidence score) üretilmiştir. Modelin tespit performansını optimize etmek için Non-Maximum Suppression (NMS) algoritması uygulanmıştır.

B. Problem 2.2: Özel Nesne Sınıflandırması (HOG + SVM)

Kendi seçilen nesne kategorisi olan köpek tespiti/sınıflandırması için özelleştirilmiş bir HOG + Lineer SVM modeli eğitilmiştir.

1) Veri Toplama ve Ön İşleme: Çalışma için Dognondog Dataset kullanılmış ve dengeli bir veri kümesi oluşturulmuştur. Tüm görüntüler, HOG pencere boyutu olan 64×128 piksel boyutuna yeniden ölçeklendirilmiş ve gri tonlamalıya dönüştürülmüştür.

2) HOG Özellik Çıkarımı: Özel nesne sınıflandırması için scikit-image kütüphanesinin hog fonksiyonu kullanılmıştır. Seçilen HOG parametreleri, Tablo II'de gösterilmiştir.

3) Veri Toplama ve Ön İşleme: Çalışma için Dognondog Dataset kullanılmış ve dengeli bir veri kümesi oluşturulmuştur. Tüm görüntüler, HOG pencere boyutu olan 64×128 piksel boyutuna yeniden ölçeklendirilmiş ve gri tonlamalıya dönüştürülmüştür.

4) HOG Özellik Çıkarımı: Özel nesne sınıflandırması için scikit-image kütüphanesinin hog fonksiyonu kullanılmıştır. Seçilen HOG parametreleri, Tablo II'de gösterilmiştir.

Tablo III
SVM VE KNN SINIFLANDIRICI BAŞARIM METRIKLERI ($N = 100$ VERI SETI ÜZERINDE)

Sınıflandırıcı	Doğruluk (Accuracy)	Hassasiyet (Precision)	Geri Çağırma (Recall)	F1-Skoru (F1-score)
Linear SVM	0.7000	0.6944	0.6900	0.6922
KNN (K = 5)	0.7000	0.7500	0.6700	0.6970

5) Sınıflandırma ve Model Eğitimi (SVM/KNN): Toplanan HOG özellik seti, Lineer Destek Vektör Makinesi (LinearSVC) ve K-En Yakın Komşuluk (KNN, K=5) algoritmaları kullanılarak sınıflandırılmıştır. Veri seti, 70% Eğitim ve 30% Test kümelerine ayrılmıştır.

IV. BAŞARIM KARŞILAŞTIRMASI VE TARTIŞMA

Eğitilen iki modelin test kümesi üzerindeki sonuçları Tablo III'te sunulmuştur.

Her iki sınıflandırıcının da $\mathbf{0.70}$'lik aynı Doğruluk oranını elde ettiği gözlemlenmiştir. Ancak, KNN modeli $\mathbf{0.7500}$ ile daha yüksek bir Hassasiyet değeri göstererek, pozitif tahminlerinin (Köpek) daha güvenilir olduğunu işaret etmiştir. Bu sonuçlar, HOG'un görece küçük bir veri seti ($N = 100$) üzerinde dahi nesnenin şekil bilgisini etkili bir şekilde kodlayarak geleneksel makine öğrenimi algoritmalarıyla kabul edilebilir bir başarım sergilediğini göstermiştir.

V. SONUÇ VE TARTIŞMA

Bu çalışma, bilgisayarla görü alanında yaygın olarak kullanılan Histogram of Oriented Gradients (HOG) özellik tanımlayıcısının derinlemesine incelenmesi, sıfırdan implementasyonu ve görüntü sınıflandırma problemlerine uygulanması üzerine odaklanmıştır.

A. HOG İmplementasyonunun Doğrulanması

Çalışmanın ilk aşamasında, HOG algoritmasının temel bileşenleri manuel olarak kodlanmıştır. **Matematiksel Doğrulama** ve **Parametre Analizi** başlıklarında yer alan deneysel testler, HOG'un kenar yönelimlerini başarılı bir şekilde kodladığını ve parametrelerin vektör boyutunu doğrudan etkilediğini kanıtlamıştır.

B. Nesne Sınıflandırma Başarısı

HOG özelliklerinin nesne tespiti ve sınıflandırmadaki pratik etkinliği, **İnsan Tespiti** ve **Özel Nesne Sınıflandırması (Köpek Tespiti)** aşamalarında gösterilmiştir. Basit sınıflandırıcılarla bile 0.70 doğruluk elde edilmesi, HOG'un ayırt edici şekil özelliklerini başarıyla kodladığını kanıtlamaktadır.

C. Genel Değerlendirme ve Gelecek Çalışmalar

HOG; mühendislik, robotik ve gömülü sistemler gibi kaynak kısıtlı ortamlarda hızlı ve güvenilir nesne tespiti ve sınıflandırması için etkinliğini koruyan temel bir görüntü işleme tekniğidir. Gelecek çalışmalarda, SVM hiperparametre optimizasyonu, veri artırımı ve doğrusal olmayan çekirdek denemeleri gibi yaklaşımlar araştırılabilir.

KAYNAKLAR

[1] Preethampaul, "HOG," GitHub repository. [Online]. Available: https://github.com/preethampaul/HOG

[2] E. C. Akdağlı, "Makine Öğrenmesinde Histogram of Gradients (HOG) Algoritması," Medium. [Online]. Available: https://ece-akdagli.medium.com/makine-%C3%B6%C4%9Frenmesinde-histogram-of-gradients-hog-algoritmas%C4%B1-b4eb3c1e69e3

[3] M. F. Toprak, "Histogram of Oriented Gradients (HOG) Algoritması: Makine Öğrenmesi," Medium. [Online]. Available: https://mfatihto.medium.com/histogram-of-oriented-gradients-hog-algoritmas%C4%B1-makine-%C3%B6%C4%9Frenmesi-becfc52b09ae

[4] scikit-image Developers, "scikit-image: Image processing in Python." [Online]. Available: https://scikit-image.org/

[5] scikit-image, "scikit-image/scikit-image," GitHub repository. [Online]. Available: https://github.com/scikit-image/scikit-image

[6] SamPlvs, "Object-detection-via-HOG-SVM," GitHub repository. [Online]. Available: https://github.com/SamPlvs/Object-detection-via-HOG-SVM

[7] A. Akgül, M. Çelebi, "HOG ile CNN Tabanlı Yaya Tespit Sistemleri İçin Performans Analizi," Dergipark. [Online]. Available: https://dergipark.org.tr/en/download/article-file/1913733 (Makale Başlığı)

[8] Elastic, "What is k-nearest neighbors (k-NN)?," Elastic.co. [Online]. Available: https://www.elastic.co/what-is/knn

[9] scikit-learn Developers, "1.4. Support Vector Machines," scikit-learn Documentation. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[10] Murtaza's Workshop - Robotics and AI, "HOG (Histogram of Oriented Gradients) with OpenCV - Python Computer Vision," YouTube. [Online Video]. Available: https://youtu.be/RaaGoB8XnxM?si=vKySDrhAKgksO5f_