T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL RESİMLERDEKİ YAYALARIN TESPİTİ

YUSUF ENGİN TETİK

DOKTORA TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
HABERLEŞME PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. BÜLENT BOLAT

İSTANBUL, 2014

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL RESİMLERDEKİ YAYALARIN TESPİTİ

Yusuf Engin TETİK tarafından hazırlanan tez çalışması 14.02.2014 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Bülent BOLAT

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Bülent BOLAT

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. İlker BAYRAM

İstanbul Teknik Üniversitesi _____

Prof. Dr. Tülay YILDIRIM

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Nihan KAHRAMAN

Yıldız Teknik Üniversitesi _____

Prof. Dr. Ece Olcay GÜNEŞ

İstanbul Teknik Üniversitesi _____

ÖNSÖZ

Bu tezi benden desteklerini hiç esirgemeyen sevgili anneme ve babama adıyorum.

Mart, 2014

Yusuf Engin Tetik

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

ε	ağırlıklı hata miktarı
$h(.)$	sınıflandırıcı fonksiyon
β	sınıflandırıcının ağırlığı
rsk	referans sınır kutusu
tsk	tespit sınır kutusu
e	sınıflama hatası
w	örnek ağırlığı
t	iterasyon
j	sınıflandırıcı indeksi
i	örnek indeksi
x	öznitelik değeri
y	sınıflama sonucu

KISALTMA LİSTESİ

AKD	Ayrık Kosinüs Dönüşümü
DET	Detection Error Tradeoff
DN	Doğru Negatif
DP	Doğru Pozitif
FPGA	Field Programmable Gate Array
FPPW	False Positive Per Window
HIKSVM	Histogram Intersection Kernel based Support Vector Machines
HN	Hatalı Negatif
HOG	Histogram of Oriented Gradients
HP	Hatalı Pozitif
HPO	Hatalı Pozitif Oranı
IEEE	Institute of Electrical and Electronics Engineers
INRIA	Institut National de Recherche en Informatique et en Automatique
kNN	k-Nearest Neigbors algorithm
KO	Kaçırma Oranı
LIDAR	Light Detection and Ranging
NICTA	National Information and Communications Technology Australia
PASCAL	Pattern Analysis, Statistical Modeling and Computational Learning
REC	Receiver Operator Characteristics
SIFT	Scale Invariant Feature Transformation
SVM	Support Vector Machines
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

SAYISAL RESİMLERDEKİ YAYALARIN TESPİTİ

Yusuf Engin TETİK

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Doktora Tezi

Tez Danışmanı: Yrd. Doç. Dr. Bülent BOLAT

Nesne tespiti bilgisayarla görü çalışmalarındaki temel çalışma konularından biridir. Çeşitli görüntü işleme ve işlemsel zekâ algoritmalarının birlikte kullanılmalarını gerektirir. Bu tezde nesne tespitindeki en zor konulardan biri olan yaya tespiti incelenmiştir.

Yaya tespiti uygulamalarında akıllı araçların önlerine çıkabilecek yayaları önceden tespit edip sürücülerini uyarmaları ya da çarpmayı engelleyecek önlemleri almaları beklenmektedir. Görme için çeşitli sensörler ya da kameralar kullanılabilir. Kameralı sistemlerde aracın önüne yerleştirilecek tek bir kamera ya da stereo görüntü elde etmek amacıyla iki kamera kullanılabilir. Aracın hareketinden dolayı kamera görüntüsünün sürekli titremesi akan görüntüler arasındaki hareket bilgisinden yararlanmayı zorlaştırmaktadır. Bu duruma bir çözüm olarak yayaların kameradan alınan durağan görüntüler üzerinde aranması önerilmektedir. Bu çalışmada, tek bir kameradan alınan durağan görüntülerdeki yayaları tespit edebilecek bir yaya tespit sistemi üzerinde durulmuştur.

Bu tezde yapılan çalışmalar sonucunda yayaların tespitinde olduğu gibi başka nesnelerin tespitinde de kullanılabilecek genel bir nesne tespit sistemi oluşturulmuştur. Bunun yanısıra, basit sınıflandırıcıları biraraya getirerek daha güçlü bir sınıflandırıcı oluşturulması probleminin hata matrisi adı verilen bir matris üzerinden çözülebilecek bir optimizasyon problemi olduğu gösterilmiştir. Bu problemin cebirsel çözümü

incelenerek Adaboost'un bu problem için en iyi çözümü bulamasa bile en iyiye yeterince yakın bir çözüm bulabildiği de gösterilmiştir. Bu çözüm, sınıflandırıcıların sınıflama sonuçlarının yanısıra bu sınıflandırıcıların ne derece güvenilir olduğunu gösteren bir güvenilirlik skoru kullanmalarını sağlayarak geliştirilmiştir.

Yaya tespit sisteminde yukarıda değinilen Adaboost yapısının yanı sıra resmin entropisini kullanan yeni bir öznitelik bulma yöntemi geliştirilmiş ve bu yöntemin diğer öznitelik bulma yöntemleriyle birlikte kullanıldığında tespit doğruluğunu arttırdığı gösterilmiştir. Ayrıca, işlemsel karmaşası fazla olan entropik özniteliklerin hızlı bir şekilde hesaplanabilmesi için yeni bir yöntem önerilmiştir.

Anahtar Kelimeler: Bilgisayarla görü, nesne tespiti, görüntü işleme, örüntü tanıma, yapay zekâ, yaya tespiti

ABSTRACT

PEDESTRIAN DETECTION IN DIGITAL PICTURES

Yusuf Engin TETİK

Department of Electronics and Communications Engineering

PhD. Thesis

Advisor: Assoc. Prof. Dr. Bülent BOLAT

Object detection is one of the fundamental subjects in computer vision studies. It requires collaboration of many image processing and computational intelligence algorithms. In this study, pedestrian detection, one of the most challenging tasks in object detection, is investigated.

In pedestrian detection applications, intelligent vehicles detect pedestrians at their path and warn drivers or take precautions to prevent collisions. A variety of sensors or cameras can be used for this purpose. In these systems, a camera is attached to the front panel of the vehicle, or two cameras can be used to have a stereo vision. Mobility of the vehicles makes it difficult to make use of the motion information taken from consecutive images since the video taken from these cameras is tilting continuously. As a solution to this problem, the detection of pedestrians in still images taken from these cameras is proposed. Thus, in this study a pedestrian detection system that uses still images taken from a single camera is investigated.

As a result of this thesis, a general object detection system has been created which can be used in detection of other objects as well as pedestrians. Besides, it is shown that the problem of constructing powerful classifiers by using simple ones is an optimization problem which can be solved by using a matrix that is called as error matrix. The algebraical solution to the problem is investigated and it is also shown that Adaboost proposes good enough solutions, but does not guarantee the best one. This

solution is improved by using confidence scores which show the confidence degree of each classification made by the simple classifiers as well as the result of the classification.

A novel feature extraction method is proposed which uses image entropy and it is shown that detection accuracy is improved dramatically when it is used with other feature detection methods. Furthermore, since the computational cost of calculating entropic features is quite high, a method is proposed to compute the entropic features rapidly.

Keywords: Computer vision, object detection, image processing, pattern recognition, artificial intelligence, pedestrian detection

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Literatürde, yaya tespiti probleminin çözümüne dair farklı yaklaşımlar bulunmaktadır. Bu yaklaşımlardan bazıları stereo görüntülerin kullanımını gerektirir. Örneğin Zhao ve Thorpe [1], yaya tespitinde kullandıkları stereo görüntüleri elde etmek için ikili bir kamera sistemi kullanmışlardır. Resimleri, nesne adaylarını barındıran alt resimlere bölütlemiş ve bir yapay sinir ağına (YSA) beslemişlerdir.

Broggi vd. [2] ise monoküler (tek kameradan çekilmiş) görüntü üzerinde kaba bir tespit yaptıktan sonra stereo resmi kullanarak uzaklık hakkında bilgi vermeye çalışmışlardır. [3]'de stereo gece görüşü kameraları kullanarak bir yaya tespit sistemi geliştirilmiştir. Wang ve arkadaşları ise [4], parçacık filtreleme ve ortalama kayma olarak adlandırdıkları algoritmalarını kullanarak yayaları izleyen bir metod geliştirmişlerdir.

[5]'de insanların şekil ve hareketlerini tespit eden stereo bir görüş sistemi geliştirilmiştir. Bunların yanısıra sadece monoküler görüntüyle çözüm üreten yaklaşımlar da oldukça popülerdir [6],[7],[8],[9],[10].

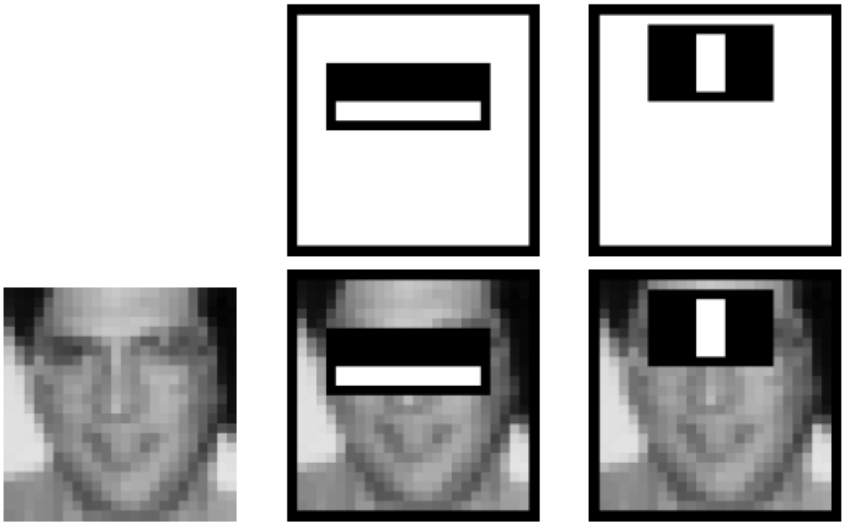
Güncel çalışmaların çoğu, ışığın insan gözü tarafından farkedilebilir spektrumu üzerinde yaya tespiti yapmaya çalışmaktadır [6], [7], [11], [12]. Fakat farklı bilgi kaynaklarını kullanarak yaya tespiti yapmaya çalışan sistemler de bulunmaktadır. Örneğin, Dai vd. [13] kızılötesi alıcılar kullanarak yaya tespiti yapmışlardır. [5]'de sadece gece görüşü kamerası kullanılmıştır. Premebida vd. ise [14] bir günışığı kamerası ve LIDARdan oluşan bir alıcı birleşimi denemişlerdir.

Yaya tespiti için çok sayıda farklı sınıflandırıcı denenmiştir. Örneğin, Zhao ve Thorpe [1] 3 katmanlı ileri beslemeli bir YSA kullanmışlardır. [15]'de hipoteze dayalı bölütleme üzerinde ipucu çıkarımı yapan bir sınıflandırıcı denenmiştir. [4]'de, parçacık filtresi ve ortalama kayma algoritmaları birleştirilerek bir sınıflandırıcı oluşturulmuştur. Cao vd. [8], destek vektör makinalarını hem öznitelik çıkarımı hem de sınıflandırma için kullanmışlardır. Liu vd. [3], geliştirdikleri yaya tespit sisteminde Realboost algoritmasını sınıflandırıcı olarak kullanmayı denemişlerdir.

Viola ve Jones'un [16] hızlı yaya tespiti için Adaboost'u kullandıkları yaklaşıma dayanan yaya tespit sistemleri halen en etkili yaya tespit sistemleri arasındadır [7], [11]. En hızlı yaya tespit sistemlerinden biri olduğu iddia edilen [17] Dollar vd.'nin kanal öznitelikleri kullananan tespit sistemi [18], büyük oranda Viola ve Jones'un çalışmasını [16] temel alır. Dollar vd. [18] bu çalışmalarında Viola'nın yaklaşımını çeşitli görüntü kanallarına uygulayarak elde ettiği sonuçları harmanlar ve nihai tespit sonucunu verir. Bu tespit yaklaşımında kullanılan teknikler [19], [20], [21] çalışmalarında iyileştirilerek tespit başarısı arttırılmıştır. Aşağıda literatürde en çok referans edilen yaya tespit sistemleri hakkında bilgi verilmektedir.

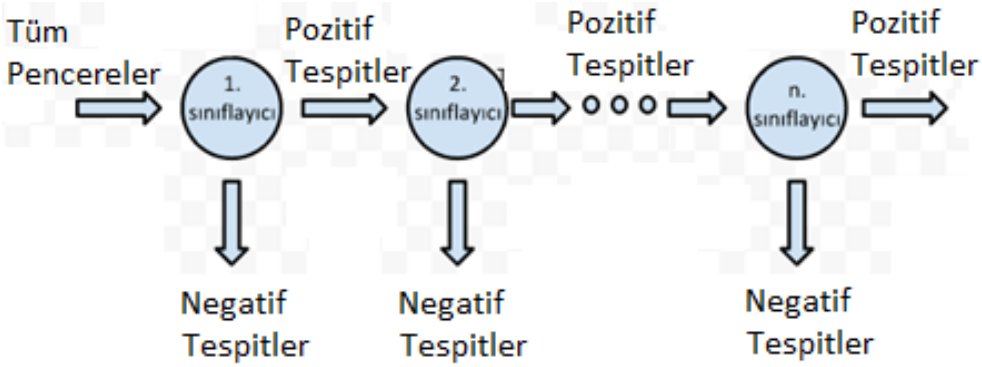
Viola ve Jones (VJ) Yöntemi: Viola ve Jones'un basit öznitelikleri kullanan hızlı bir nesne tespit yöntemi olarak önerdikleri [16] ve bunu yüz tespitine uyguladıkları yöntemleri oldukça popüler olmuş ve kendi adlarıyla anılır hale gelmiştir. Viola ve Jones, ilk olarak dikdörtgen farklarını yeni bir öznitelik olarak önermişler, bu basit özniteliklerin hızlı bir şekilde hesaplanabilmesi için integral resim yöntemini kullanmışlardır. Adaboost sınıflama algoritmasını, bu basit özniteliklerden oluşan güçlü bir sınıflandırıcı yapabilmek için kullanmışlardır.

Dikdörtgen fark öznitelikleri, Papageorgiou vd. [22] tarafından önerilen Haar dalgacıklarına benzemektedir. Buna göre çok sayıda dikdörtgen şablonları oluşturulur. Bu şablonlar daha küçük dörtgenlerden oluşmaktadır ve şablonu oluşturan bu dörtgenler içinde kalan piksel toplamları şablonun öngördüğü şekilde birbirinden çıkarılarak bir öznitelik bulunmuş olur. Bu yöntem nesne üzerinde koyulukları birbirine göre değişmeyen bölgeler bulunduğu varsayımına dayanır. Örneğin yüz tespitinde kaşların alınlardan daha koyu olması ya da gözler arasındaki boşluğun gözlerden daha açık olması gibi (Şekil 1.1).



Şekil 1. 1 Dikdörtgen şablonları kullanan yüz tespit sisteminde en ayırt edici olarak bulunan dikdörtgen fark öznitelikleri [16]

Son olarak farklı sayıdaki özniteliklerden oluşan Adaboost sınıflandırıcıları arka arkaya bağlayarak çok daha hızlı bir şekilde nesne tespiti yapmayı önermişlerdir. Buna göre, ilk baştaki Adaboost sınıflandırıcısı oldukça az sayıda (3 ya da 5 gibi) öznitelik kullanır. Bu ilk sınıflandırıcı negatif örneklerin büyük çoğunluğunu eler, fakat pozitif olarak tespit ettiği örneklerde hatalı pozitif oranı oldukça yüksektir. Bu hatalı pozitifleri elemek içinse hemen arkasındaki daha fazla sayıda öznitelikten oluşan bir Adaboost sınıflandırıcısına ilk sınıflandırıcı tarafından pozitif olarak tespit edilen örnekler verilir. İstenilen başarı oranına ulaşılıncaya kadar daha yüksek sayıda öznitelik kullanan Adaboost sınıflandırıcılarının eklenmesine devam edilir (Şekil 1.2).

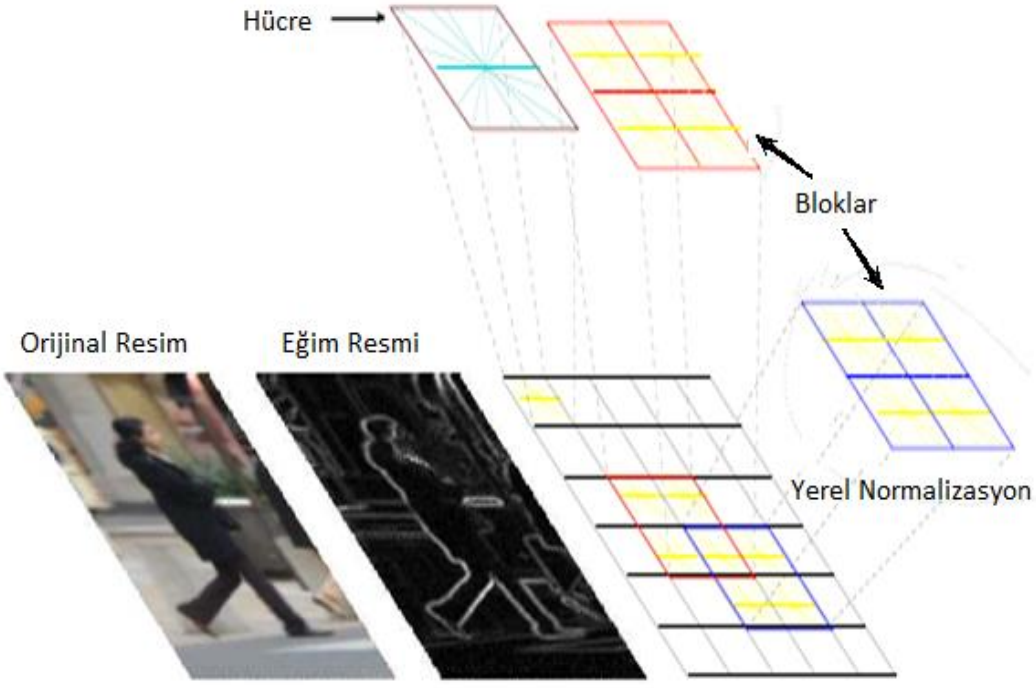


Şekil 1. 2 Sınıflandırıcıların arka arkaya sıralanmış olarak kullanılması [16]

Yönlü Eğimlerin Sıklık Grafiği: Yönlü eğimler resme ait karakteristik özellikleri ifade eden öznitelik bulma yöntemlerinden biridir. Kenar yönlü sıklık grafiklerine [23] ve

ölçeğe göre değişmeyen öznitelik dönüşümlerine (Scale Invarian Feature Transformation, SIFT) [24] benzeyen bu özniteliklerin yaya tespitinde kullanımı Dalal ve Triggs [25] tarafından önerilmiştir. Dalal ve Triggs elde ettikleri bu öznitelikleri sınıflandırmak için doğrusal destek vektör makinalarını (SVM) kullanmışlardır.

Bu yönteme göre öncelikle resme (-1, 0, 1) gibi bir maske uygulanarak eğimler bulunur. Resim küçük boyuttaki üst üste binmeyen hücre denilen piksel bölgelerine ayrılır. Her bir hücre için 1 boyutlu eğim yön histogramları hesaplanır. Hücredeki her piksel için eğim 9 eğim yönünden (0-40, 41-80, ... ,321-360) uygun olanına atanır ve her piksel eğiminin büyüklüğü ile orantılı olarak atandığı yöne oy verir. Hücreler gruplanarak bloklar oluşturulur. Böylece her hücrenin histogramı blok içerisindeki diğer hücrelerin eğim enerjilerine bakılarak normalize edilir. Resim üzerindeki her bloktan alınan histogram vektörleri toplanarak sınıflamada kullanılacak olan öznitelik vektörü bulunmuş olur (Şekil 1.3).



Şekil 1. 3 Yönlü eğimlerin sıklık grafiği yöntemiyle öznitelik çıkarımı [25]

Literatürde bu yaklaşımı temel alarak geliştirilen birçok yaya tespit sistemi bulunmaktadır [26], [27], [28], [29]. Hahnle vd. [30] bu yaklaşımın FPGA tabanlı gerçekleştirimini yaparak yüksek çözünürlüklü resimlerde gerçek zamanlı çalışan bir yaya tespit sistemi elde etmişlerdir. Wei vd. [31] yine Dalal ve Triggs'in yaklaşımını

temel alarak kullanılan (Histogram Oriented Gradients, HOG) özniteliklere daha fazla bilgi eklemenin yolunu bulmuş ve EHOG (Enhanced HOG) dedikleri gelişmiş HOG öznitelikleri elde etmişler. Bu öznitelikleri HIKSVM denilen bir tür SVM'le sınıflandırarak daha iyi tespit sonuçları veren bir yaya tespit sistemi elde etmişlerdir.

Şekilcik Yöntemi: Şekilcik (Shapelet) yöntemi, pozitif ve negatif örnek resimler arasındaki bölgesel şekil farklarını kullanmayı öngörür. Örneğin resmin üst bölgesinde baş ve omuzların oluşturduğu ters omega şeklini yayanın varlığına dair önemli bir kanıt sayar (Şekil 1.4). Pozitif ve negatif örnekleri ayrıştırmayı sağlayan bu şekillere şekilcik adı verilir. Bu yöntemde [32], kayan pencere yaklaşımı ve Adaboost sınıflandırıcısı kullanılmaktadır. Şekilcik yöntemi, kenarcık (edgelet) yöntemine [33] benzemektedir.

Şekil 1. 4 En üstteki satır pozitif örnek resimleri göstermektedir. Ortaki satırda örneklerde kullanılan 3 bölge gösterilmiş, en alltaki satır ise bu bölgelerden elde edilen şekilcikler gösterilmiştir [32].

Yao ve Deng [34], iki aşamalı bir yaya tespit sistemi tasarlamış ve ilk aşamada şekilcik yöntemi kullanarak yayanın bütünü bulunmuş, ikinci aşamada bu bütün üzerinde Haar-benzeri öznitelikleri kullanarak kol, bacak, kafa, vb. alt nesneleri aramış ve bulunan nesnenin yaya olduğunu doğrulamışlardır. Yonghzi vd. [35] ise tersi bir yaklaşımı uygulayarak önce Haar-benzeri öznitelikleri kullanarak yaya olabilecek bölgeleri hızlıca tespit ederek ilgi bölgelerinin belirlenmesini sağlamış ve son aşamada bu bölgelerde yaya olup olmadığını doğrulamak için şekilcik yöntemini kullanmışlardır. Hong vd. [36] ise yine Haar benzeri öznitelikleri kullanan bir sınıflandırıcı ile bir şekilcik sınıflandırıcısından aldıkları sonuçları harmanlayarak tespit yapan bir sistem önermektedirler.

1.2 Tezin Amacı

Bilgisayarların gördükleri nesneleri insanlar kadar iyi tanımaları, bilgisayarla görü çalışmalarının en temel amaçlarından biridir. Bilgisayarlar insanlardan daha keskin veya daha geniş bir spektrumda görebilirler, fakat ne gördüklerini anlayacak zekaya ya da farkındalığa sahip değillerdir. Dolayısıyla bilgisayarın görmesi esasen yapay zeka çalışmalarıyla yakından ilgilidir.

Bilgisayarla görü konusunda yapılan çalışmalar genellikle endüstrinin ihtiyaç duyduğu özel nesneleri (yüz, araba, yaya, uçak, gemi, vs.) tanıma konusunda özelleşmiş sistemler üzerinedir. Bunun nedeni her nesneyi tanıyacak genel bir nesne tanıyıcının oluşturulmasının ve uygulanmasının zorluğudur.

Bu çalışmada amaçlanan genel bir nesne tespit yönteminin oluşturulmasıdır. Bir nesneyi aramaya özelleşmiş sistemler o nesneyi bulmayı kolaylaştıracak yöntemler kullanırlar. Örneğin yayaların dik veya dike yakın kenarlardan oluşması, simetrik olmaları vb. kendilerine özgü niteliklerini kullanan ve yaya tespitinden başka bir yerde kullanılması mümkün olmayan bir tespit sistemi geliştirilebilir. Genel bir nesne tespit sistemi ise yayalara özgü bu nitelikleri açıkça kullanmadan yaya tespiti yapmayı hedefler.

Yaya tespiti, son zamanlarda üzerinde yapılan çalışmaların arttığı güncel araştırma konularından biridir. Birçok uygulama alanı bulunmaktadır; güvenlik videoları, robotik, görme engellilere yardım, sürüş desteği, içerik tabanlı indeksleme gibi. Sürüş destek

sistemleri, yaya tespitinin kullanıldığı alanlar arasında öne çıkmaktadır [4], [37]. Sürüş desteği amaçlı uygulamalarda akıllı araçların önlerine çıkabilecek yayaları önceden tespit edip sürücülerini uyarmaları ya da çarpmayı engelleyecek önlemleri almaları beklenmektedir. Yapılan araştırmalara göre sadece Amerikada bir yılda gerçekleşen 35000 ölümlü kazanın 5000 tanesinde yayalar bulunmaktadır[38].

Dollar vd. [11], bilinen en modern 16 yaya tespit sistemini ve 6 farklı veri kümesini kullanarak elde ettikleri sonuçları değerlendirirken, performansın iyileştirmeye çok açık olduğunu ve özellikle düşük çözünürlüklerde ya da kısmen birbirini örten yayalar için tespit oranlarının hayal kırıklığı yarattığını belirtmişlerdir. Dolayısıyla yaya tespiti gelişmeye ve araştırmaya açık olan bir konudur ve bu tezde önerilecek yöntemlerin buna katkıda bulunması hedeflenmektedir.

Yaya, tespiti en zor nesnelerden biridir, çünkü yayalar kesin bir şekle sahip değildir ve çok çeşitli duruş ve şekillerde bulunabilir. Özellikle kollar ve bacakların duruşu, farklı şekillerin ortaya çıkmasına neden olur. Görüş açışı, ışıklandırmanın yönü ve yoğunluğu yayanın şeklini önemli ölçüde etkiler. Yaş, cinsiyet, hava şartları ve kültürlere bağlı olarak giyilen giysiler yayanın görünüşünü çok değiştirebilir. Çanta, şemsiye gibi taşınan eşya ya da aksesuarlar yayaları kısmen örter. Gerçeğe uygun bir senaryoda kalabalık sahneler, hareket eden nesneler bulunur ve yayalar kısmen başka nesnelerin arkasında kalır ya da birbirlerine karışır. Bu tez sonucunda geliştirilecek nesne tespit yöntemiyle yayaların daha başarılı bir şekilde tespiti amaçlanmaktadır.

1.3 Hipotez

Bu çalışma boyunca geliştirilen yöntemlerle yaya tespit sistemlerinin başarıları arttırılmaya çalışılmıştır. Bu çalışmalar ilerleyen bölümlerde detaylı olarak anlatılmaktadır. İkinci bölümde yaya tespitine dair konuya giriş niteliğinde temel bazı bilgiler verilmektedir.

Üçüncü bölümde, aranan asıl nesneyi belirleyen her bir alt nesnenin ayrı ayrı tespiti, daha sonra bu alt nesnelerden yola çıkarak asıl nesnenin bütününü tespit etme amacıyla geliştirilen yaya tespit sistemi [39] anlatılmaktadır. Diğer bir deyişle, bu yöntem aşağıdan-yukarıya yaklaşımından yararlanmaktadır. Nesnelerin modellenmesi,

nesneye ait bölgelerin ortalama koyuluk ağırlıkları arasındaki ilişkiden yararlanan oran-şablonu [40] yaklaşımı kullanılarak yapılmıştır.

Dördüncü bölümde uyarlanabilir arttırma ("adaptive boosting") [41] yöntemine dayalı bir yaya tespit sistemi önerilmektedir. Bu yöntemde kullanılan öznitelikler Haar-benzeri şablonlar [16] kullanılarak üretildi. Geliştirilen bu yöntemin etkinliğini göstermek adına eğitimi sırasında farklı bir veri kümesi (NICTA [42]) ve testi için farklı bir veri kümesi (Penn Fudan [43]) kullanılmıştır. Beşinci bölümde uyarlanabilir arttırma yöntemine benzer bir yöntem geliştirilmiş [20] ve bu yöntem farklı öznitelik çıkarım yöntemleriyle desteklenerek zayıf sınıflandırıcılardan oluşan bir yaya tespit sistemi geliştirilmiştir.

Altıncı bölümde, uyarlanabilir arttırma ("adaptive boosting") yöntemi sınıflandırıcıların güvenilirlik derecelerini de kullanacak şekilde geliştirilmiş [19] ve bu şekilde geliştirilen arttırma yöntemi farklı sınıflandırıcılarla desteklenerek yaya tespit sisteminin performansı arttırılmaya çalışılmıştır. Bu bölümde ayrıca entropi bilgisinden yararlanan bir öznitelik çıkarım yöntemi ve bu öznitelikleri hızlı hesaplama yöntemi anlatılmaktadır. Bu entropi özniteliklerinin diğer öznitelik yöntemleriyle birlikte kullanılması halinde yaya tespit sisteminin başarısının oldukça arttığı gözlenmiştir.

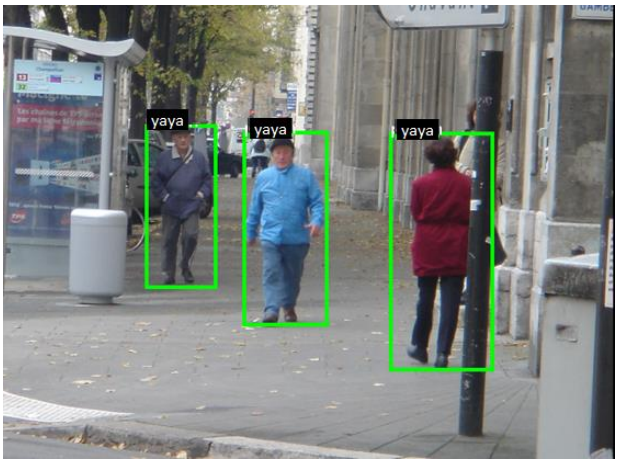
Böylece geliştirilen yeni öznitelik çıkarım yöntemleri ve sınıflama yöntemine yapılan katkılar sayesinde emsallerinden daha yüksek başarı oranlarına erişebilen genel bir nesne tespit sistemi oluşturulmuş, elde edilen sonuç ve öneriler son bölümde sunulmuştur.

BÖLÜM 2

SAYISAL RESİMLERDE YAYA TESPİTİ

Yaya tespiti, sayısal resimleri çeşitli görüntü işleme ve işlemsel zekâ algoritmaları kullanarak işleyerek resim içerisinde bulunan yaya nesnelerinin tespitini amaçlar. Resim içerisinde dik pozisyonda duran yani ayakta olan, yürüyen ya da koşan insanlar tespit edilmeye çalışılan yaya nesneleridir. Bu resimler sokakta, yolda, bahçede, alışveriş merkezinde, kampüste, vb. dış veya iç ortamlarda çekilmiş olabileceğinden arkaplan oldukça değişkenlik gösterir.

Tespit işlemi, resimde yaya içeren bölgenin dikdörtgensel çerçeveler içine alınması olarak tarif edilir. Bu çerçevelere sınır kutuları denilmektedir. Şekil 2.1'de de görüldüğü gibi yaya tespit sisteminin resimler içerisinde bulunan yayaları sınır kutularıyla işaretlemesi beklenmektedir.



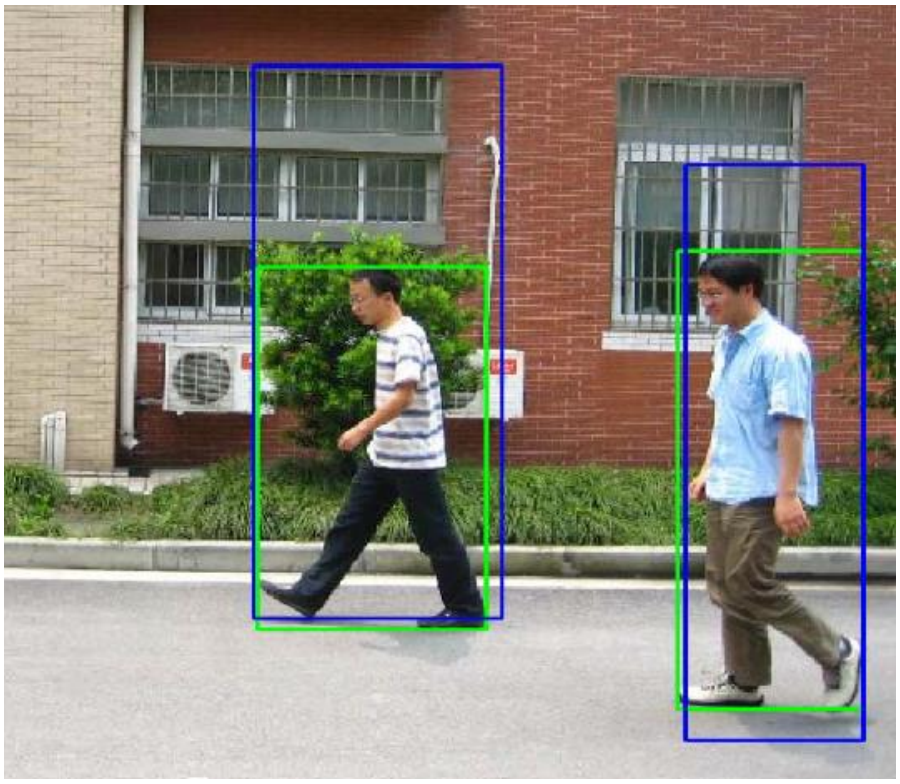
Şekil 2. 1 INRIA yaya veri kümesinden bir test resmi. Yeşil çerçeveler veri kümesi tarafından önceden işaretlenmiş yayaları temsil ediyor [25]

Yaya tespiti konusundaki çalışmalar, geniş veri kümeleriyle desteklenmektedir. Bir yaya veri kümesi, yaya içeren “pozitif” ve yaya içermeyen “negatif” resim gruplarından oluşur. Ayrıca bu veri kümelerinde eğitim için kullanılacak örnekler ile test için kullanılacak örnekler önceden ayrıştırılmıştır. Örnek resimler tespit edilecek bir yayanın boyutundadır (Şekil 2.2).

Şekil 2. 2 Nicta yaya veri kümesinden 32x80 piksel boyutunda pozitif (üst sıra) ve negatif (alt sıra) örnek resimler [42]

Tasarlanan yaya tespit sistemi tarafından tespit edilen sınır kutuları “tsk” ve veri kümesi tarafından atanan referans sınır kutuları “rsk”, PASCAL [44] ölçütüne (2.1) göre örtüşüyorsa yaya tespit edilmiş sayılır (Şekil 2.3):

$$a_0 = \frac{alan(tsk \cap rsk)}{alan(tsk \cup rsk)} \rangle 0.5 \qquad (2.1)$$



Şekil 2. 3 INRIA yaya veri kümesinden bir test resmi. Yeşil (açık gri) çerçeveler veri kümesi tarafından önceden işaretlenmiş yayaları temsil ediyor, mavi (koyu gri) çerçeveler ise bir yaya tespit sistemi tarafından bulunan sınır kutuları [25]

2.1 Yaya Yerini Belirleme Yaklaşımları

Yaya tespit sistemleri, bir resimde ya da resim dizisinde yaya barındıran sınır kutuları bulacak şekilde tasarlanır. Bir resimde, içinde yaya bulundurması muhtemel sınır kutularının sayısı oldukça yüksektir. İçerisinde yaya bulundurma ihtimali olan sınır kutularına aday sınır kutuları denir. Aday sınır kutularının bulunmasında kullanılan 2 yaklaşım vardır.

Kayan Pencere Yaklaşımı: Yaya tespit sistemlerinde en çok kullanılan yöntem kayan pencere yöntemidir. Bu yönteme göre aranan yaya boyutundaki bir pencere tüm resim üzerinde adım adım kaydırılarak her yeni pozisyonda bir yaya bulunup bulunulmadığına bakılır. Farklı boyutlardaki yayaların bulunabilmesi için pencere boyutu değiştirilebileceği gibi pencere boyutu sabit bırakılarak tüm resmin boyutu da değiştirilebilir.

Nesne Yerini Doğrudan Belirleme Yaklaşımı: Bu yöntemler aday sınır kutularının resim üzerindeki tüm pencerelerin tek tek kontrol edilmesine gerek kalmadan doğrudan

tespitini ve ardından tespit edilen aday sınır kutusunun yaya içerip içermediğinin kontrolü şeklinde çalışır.

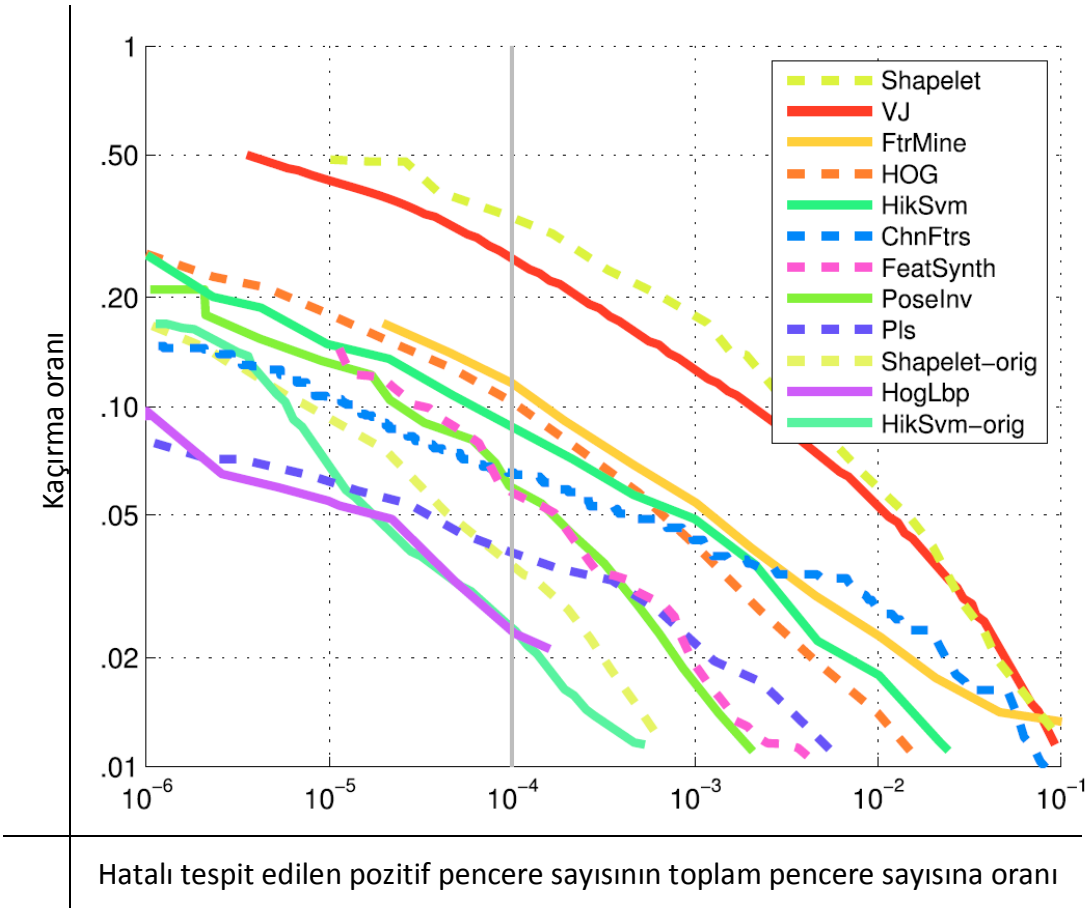
Üzerinde en çok çalışılan yöntem kayan pencere yöntemidir. [11]'de bu yöntemin düşük ve orta çözünürlükte en iyi sonuç verdiği ifade edilmektedir.

2.2 Veri Kümeleri

Yaya tespit sistemlerinin tasarımında en çok yararlanılan veri kümesi, [25] tarafından üretilen INRIA veri kümesidir. Aslında bu veri kümesi hem yeterince büyük değildir, hem de pozitif resimlerle aynı büyüklükte negatif resimler içermediğinden bu boyuttaki negatif resimlerin veri kümesindeki daha yüksek çözünürlüklü resimlerden türetilmesini gerektirir. Fakat bu konudaki en eski veri kümelerinden biri olması ve birçok çalışmada bu veri kümesiyle eğitilmiş sistemlerin performanslarının verilmesi nedeniyle halen kullanılmaya devam etmektedir. Bu veri kümesinin yanı sıra, şu an 10'dan fazla veri kümesi yaya tespit çalışmalarında kullanılmaktadır. Bu veri kümelerinden en popülerleri ve özellikleri Çizelge 2.1'de verilmiştir [11]. Günümüzdeki en başarılı yaya tespit sistemlerinin performansları Şekil 2.4 ve Şekil 2.5'deki grafiklerde gösterilmektedir. Bu yaya tespit sistemlerinin hepsi INRIA veri kümesiyle eğitilmiştir.

Çizelge 2. 1 En çok kullanılan yaya veri kümeleri. "–" sembolü ilgili özelliğin bulunmadığını, "+" ise bulunduğunu gösteriyor [11]

	Eğitim Seti			Test			Özellikler	
	# yaya	#neg. resim	#pos. resim	#yaya	#neg. resim	#pos. resim	renk	Video
INRIA [25a]	1208	1218	614	566	453	288	+	-
ETH [61a]	2388	-	499	12000	-	1804	+	+
NICTA [42a]	18700	5200	-	6900	50000	-	+	-
TUD-Brussels[45a]	1776	218	1092	1498	-	518	+	-
Daimler-DB [7a]	15600	6700	-	56500	-	218000	-	+
Caltech [46a]	192000	61000	67000	155000	56000	65000	+	+

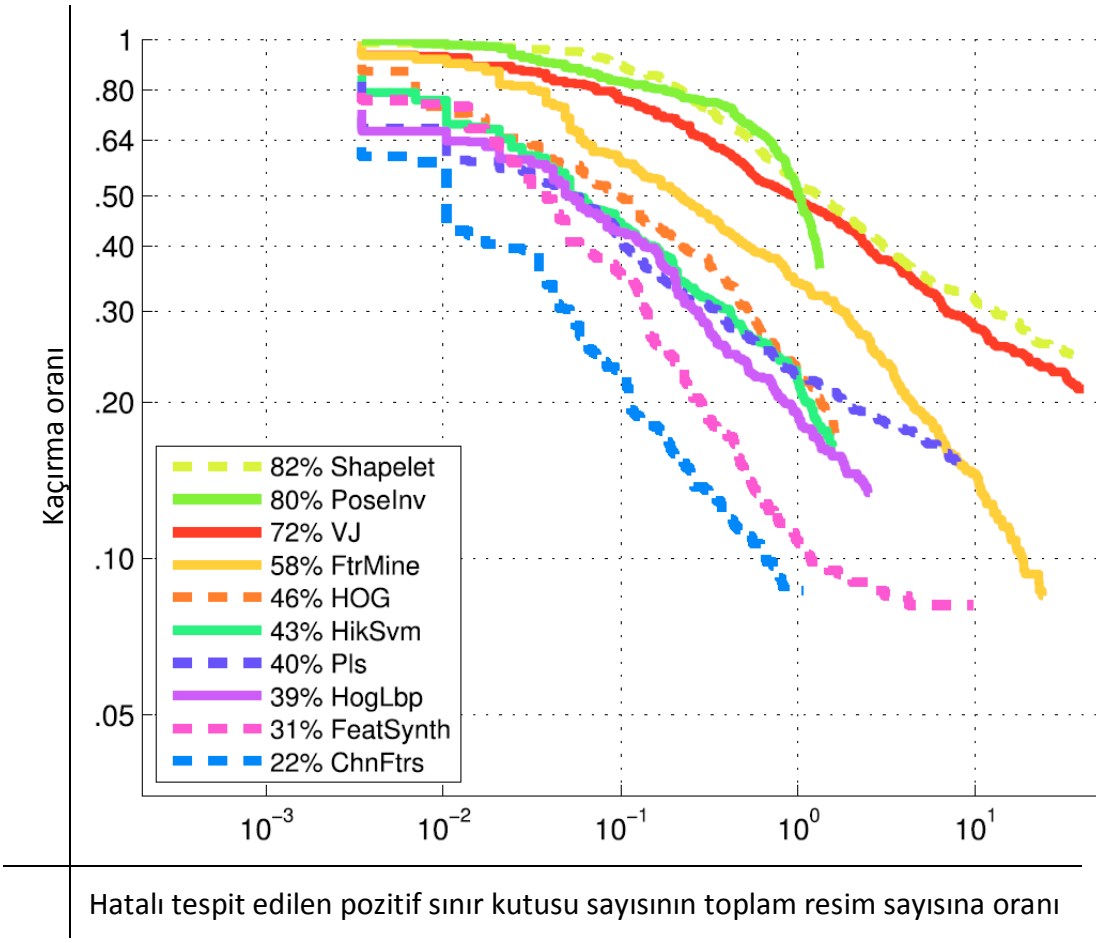


Şekil 2. 4 Bazı yaya tespit sistemlerinin pencere başarılarının karşılaştırılması [11]

2.3 Performans Ölçümü

Tespit sistemlerinde performans değerlendirmesi yapılırken aşağıdaki parametreler kullanılır:

- (HP) Hatalı Pozitif: Tespit sistemi tarafından pozitif olarak tespit edilen negatif örneklerdir. Bu durumun bir başka adı da hatalı alarm'dır.
- (HN) Hatalı Negatif: Tespit sistemi tarafından negatif olarak tespit edilen pozitif örneklerdir.
- (DP) Doğru Pozitif: Pozitif olan ve tespit sisteminin de pozitif olarak yaptığı tespitlerdir.
- (DN) Doğru Negatif: Negatif olan ve tespit sisteminin de negatif olarak yaptığı tespitlerdir



Şekil 2. 5 Bazı yaya tespit sistemlerinin resim performansları [11]

Dalal ve Triggs [25]'de kendi yaya tespit sistemlerinin performansını "Tespit Hatası Değişimi" (DET, Detection Error Tradeoff) denilen bir yöntemle göstermiş ve daha sonrasında yayınlanan yaya tespit çalışmalarında da genellikle aynı yöntem kullanılmıştır. Bu yöntemin diğer bir ölçüm metodu olan "Alıcı Çalışma Karakteristikleri"nden (ROC, Receiver Operating Characteristics) daha ayırt edici olduğu [47] tarafından konuşma işleme algoritmalarının karşılaştırılmasında gösterilmiştir.

Bu ölçüm metoduna göre grafikler Şekil 2.4 ve 2.5'deki gibi çizilir. Dikey eksen karşılaştırılması yapılmak istenen tespit sisteminin kaçırma oranını, yatay eksen ise hatalı pozitif oranını gösterir. Kaçırma oranı (KO) pozitif örneklerden kaçının tespit sistemi tarafından negatif olarak bulunduğunu gösterir ve denklem (2.2)'deki gibi hesaplanır. Hatalı pozitif oranı (HPO) ise negatif örneklerden kaçının tespit sistemi tarafından pozitif olan bulunduğunu gösterir ve denklem (2.3)'deki gibi hesaplanır. Bu iki oran da sıfıra yaklaştıkça başarı artar.

$$KO = \frac{HN}{DP + HN} \tag{2.2}$$

$$HPO = \frac{HP}{DN + HP} \tag{2.3}$$

Yaya tespit sistemlerinin karşılaştırılmasında kullanılan iki yöntem bulunmaktadır:

Pencere Başarılarının Karşılaştırılması: Bu karşılaştırma yönteminde yaya tespit sistemine sadece test etmesi gereken pencereler verilir ve bunları pozitif ya da negatif olarak sınıflandırması istenir. Performansın gösteriminde ise yatay eksende yanlış sınıflandırılan pencere sayısının toplamda test edilen pencere sayısına oranı verilir. Dikey eksende ise pozitif olmasına rağmen negatif olarak sınıflandırılan pencere sayısının veri kümesindeki toplam pozitif pencere sayısına oranı verilir (Şekil 2.4).

Yaya tespit sistemlerinin karşılaştırılması yapılırken kullanılan bazı grafiklerde yatay eksende pencere başına hatalı pozitif sayısı (FPPW: False Positives Per Window) yazılmaktadır. Bu aslında hatalı bir ifadedir, çünkü negatif pencere başına hatalı pozitif sayısı alınmalıdır. Eğer bu ifadede negatif pencere sayısı değil de, toplam pencere sayısı kullanılırsa pozitif pencere sayısı da buna katılmış olur. Oysa ki pozitif test pencerelerinin tespit sistemi tarafından yanlışlıkla pozitif olarak tespit edilmesi söz konusu değildir. Burada gösterilmek istenen kaç negatif pencerenin pozitif pencere olarak tespit edildiğidir. Dolayısıyla pencere sayısı kullanılırsa pozitif pencere sayısı da bu oranın paydasına dahil olur ve sağlıklı bir karşılaştırma yapılamaz.

Resim Başarılarının Karşılaştırılması: Bu yöntemde ise test pencereleri yerine pencere boyutundan daha yüksek çözünürlüklü test resimleri kullanılır. Bu resimler içerisinde veri kümesi tarafından işaretlenen referans sınır kutularının bulunması beklenir. Bu şekilde oluşturulan karşılaştırma grafiklerinde yatay eksende hatalı bulunan sınır kutu sayısının test edilen resim sayısına oranı verilir. Dikey eksende ise yaya tespit sistemi tarafından tespit edilemeyen referans sınır kutu sayısının toplam referans sınır kutu sayısına oranı verilir (Şekil 2.5).

Aynı yaya tespit sisteminin pencere ve resim başarıları ciddi farklar arzedebilir. Çünkü yaya tespit sistemlerinde bulunan ilgi bölgelerinin tespiti, resmi küçültme, büyütme vb. işlemlerle birlikte sistemin nasıl yapılandırıldığı resim başarısını büyük oranda etkiler.

Dolayısıyla, yaya tespit sistemleri karşılaştırılırken hangi karşılaştırma kriterinin kullanıldığı büyük önem taşır.

BÖLÜM 3

ORAN ŞABLONU YAKLAŞIMIYLA DOLAYLI YAYA TESPİTİ

Bu bölümde, aranan asıl nesneyi belirleyen alt nesnelerin ayrı ayrı tespiti ve bu alt nesnelerden yola çıkılarak asıl nesnenin bütününü tespit etmek amacıyla geliştirilen bir yaya tespit sistemi anlatılmaktadır. Diğer bir deyişle, bu bölümde anlatılan sistem aşağıdan-yukarıya yaklaşımından yararlanmaktadır.

Yaya tespitinde yayanın nasıl aranacağına dair genel olarak iki yaklaşım vardır:

Dolaylı Arama: Bu yaklaşımda, aranan nesneye ait alt-nesne ya da özniteliklerin bulunması amaçlanır. Bu bilgiler ışığında asıl nesnenin varlığı ve yeri tespit edilmeye çalışılır. Dolaylı yaklaşımda bulunan alt nesne ya da öznitelik bilgilerinin birleştirilerek asıl nesneye ilişkin bir hipotez oluşturulması gerekir ve bu genellikle kolay bir yöntem değildir.

Doğrudan Arama: Bu yaklaşımda ise nesne bir bütün olarak hedef görüntüde tespit edilmeye çalışılır. Doğrudan arama kullanan yaya tespit yöntemleri, dolaylı aramaya kıyasla genellikle daha hızlıdır fakat yanlış tespit oranları yüksektir.

Bu iki yaklaşımı birarada kullanmaya çalışan yaya tespit sistemleri de tasarlanmıştır [39], [43]. Fakat iki yaklaşımın olumsuz yönlerini almadan avantajlarını biraraya getirecek bir sistem kurulması incelikli bir tasarım gerektirir. Böyle bir sistem doğru tespit konusunda daha başarılı olsa da, sistemin tespit hızı büyük olasılıkla yukarıdaki iki yaklaşımdan da daha düşük olacaktır.

Bu bölümde anlatılan çalışmada dolaylı yaklaşım, diğer bir deyişle aşağıdan yukarıya yaklaşımı tercih edilmiştir. Yayaların beden ve uzuvlarının çok farklı pozisyonlarda bulunabilmesi yayanın doğrudan bir bütün olarak aranmasını güçleştirir. Kol ve bacak

gibi alt nesnelerin bağımsız olarak aranması ise farklı duruş şekillerinin tanınmasına izin verir ve yaya tespit sistemine bu konuda büyük esneklik kazandırır.

Dolaylı yaklaşım, sistemin var olan yayaları bulamama, yani eksik tespit etme olasılığını büyük oranda azaltabilir. Çünkü yayaya ait olabilecek bir alt nesnenin varlığı bile yayanın varlığına kanıt olarak kabul edilebilir. Fakat bu durumda yaya olmadığı halde yaya olarak tespit edilen nesnelerin, yani yanlış tespitlerin oranında bir artış olması beklenir. Dolayısıyla eksik tespit oranının azaltılması genelikle yanlış tespit oranının artmasına neden olur. Eksik tespitte bulunan bir yaya tespit sistemi daha ciddi sorunlar doğurabileceğinden yanlış tespite daha müsamahalı, ama eksik tespit yapmayan ya da olabildiğince az yapan bir sistem tasarlanmalıdır.

Nesnelerin nasıl modellendiği bir nesne tanıma sistemi için en kritik noktadır. Sistemin performansı büyük oranda kullanılan bu modelleme yöntemine bağlıdır. Bu bölümde anlatılan yaya tespit sisteminde nesnelerin modellenmesi, nesneye ait bölgelerin ortalama koyuluk ağırlıkları arasındaki ilişkiden yararlanan oran-şablonu yöntemi kullanılarak yapılmıştır.

3.1 Oran-Şablonu Yöntemi

Oran-şablonu yöntemi, ilk olarak Sinha [40], [48] tarafından önerilmiştir. Bu yöntem nesne üzerinde birbirlerine göre koyulukları değişmeyen bölgelerin olmasından yararlanır. Buna göre nesne üzerinde oluşturulan çeşitli bölgelerin koyuluk ortalamalarının oranları nesneyi tanımlayan karakteristik olarak kabul edilir.

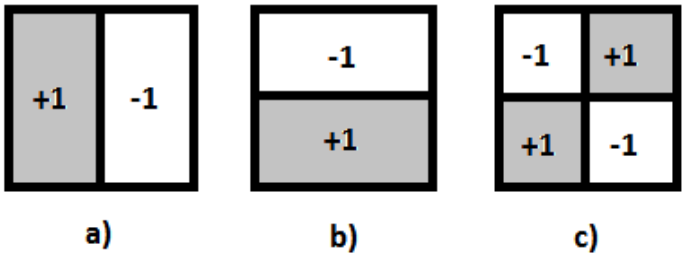
Oran şablonu yöntemi genellikle gri tonlamalı görüntülere uygulanmaktadır. Örneğin yüz tespiti sistemlerinde bu yöntemi gri tonlamalı görüntüler üzerine uygulamak başarılı sonuçlar vermektedir [16]. Bunun nedeni yüzde birbirine oranla ortalama koyuluk oranları değişmeyen bölgelerin bulunmasıdır. Örneğin, gözlerin olduğu bölge alnın olduğu bölgeye göre daha koyudur [10].

Yaya tespitinde ise, bir yaya nesnesi için bir bölgenin diğerine göre daha koyu olması gibi değişmeyen bir durumun bulunması daha az olasıdır. Çünkü yaya üzerindeki koyuluk oranları yayanın giysileri, duruşu, yayayı kısmen örten başka nesneler gibi nedenler dolayısıyla çok değişkenlik gösterir. Bununla birlikte bu modelleme metodu

Papageorgiou vd. [10], [22] tarafından yaya tespitinde denenmiştir. Bu çalışmalarda Papageorgiou ve arkadaşları bu metodu gri tonlu görüntüler üzerinde denemişlerdir. [22]'de üst seviyede bir modelleme yapmak yerine piksel değerleri ya da kenar bilgisi seviyesinde yapılacak modellemelerin başarı şansının oldukça düşük olduğu belirtilmektedir. Diğer bir deyişle aranan nesneyi hedef görüntüde piksel piksel yapılan bir karşılaştırmayla aramak uygulanabilirliği olan bir yol değildir. Bunun yerine aranan nesnenin çeşitli karakteristiklerinin çıkarılması ve hedef görüntüde bu karakteristiklerin aranması gerekir.

Viola ve Jones [16], oran şablonu metodunu uygulamak için 2 boyutlu Haar dalgacık şablonlarını andıran ama daha basit olan dikdörtgen öznitelikler adını verdikleri yaklaşımı kullanmışlardır. Dikdörtgen özniteliklerin hesaplanması oldukça basit ve hızlıdır ancak, örneğin 24x24 boyutlarındaki bir görüntü bloğu için tanımlanabilecek olası dikdörtgen özniteliklerin sayısı 180.000'den fazladır. Bu da tüm özniteliklerin kullanılması durumunda işlemsel karmaşayı azımsanamayacak bir şekilde arttırmaktadır.

Bu çalışmada [16]'da önerilen dikdörtgen özniteliklere benzeyen 3 adet kare öznitelik (Şekil 3.1) kullanılmıştır. Öznitelikler, doğrudan gri tonlu görüntü üzerine uygulanmak yerine bu görüntüden kenar çıkarma ile elde edilen kenar çizgileri görüntüsüne uygulanmıştır.



Şekil 3. 1 Öznitelik şablonları. a) Dikey öznitelik b) Yatay öznitelik c) Çapraz öznitelik

Dikey öznitelik şablonu, bloğu oluşturan eşit boyuttaki iki dikey dikdörtgendeki pikselleri toplar ve bu iki toplamın farkı bloğun dikey özniteliği olur. Yatay öznitelik şablonu ise bloğu oluşturan eşit boyuttaki yatay dikdörtgenlerdeki piksel toplamlarının farklarını bulur. Çapraz öznitelik hesaplanırken görüntü bloğu Şekil 3.1'de görüldüğü

üzere aynı boyuttaki 4 alt-kare bloğa ayrılır ve koyu bölgelerdeki piksellerin toplamından açık bölgelerdeki piksellerin toplamı çıkarılarak çapraz öznitelik elde edilir. Öznitelik şablonları şablonla aynı boyuttaki görüntü bloğuna uygulandığında her blok için 3 öznitelik elde edilir.

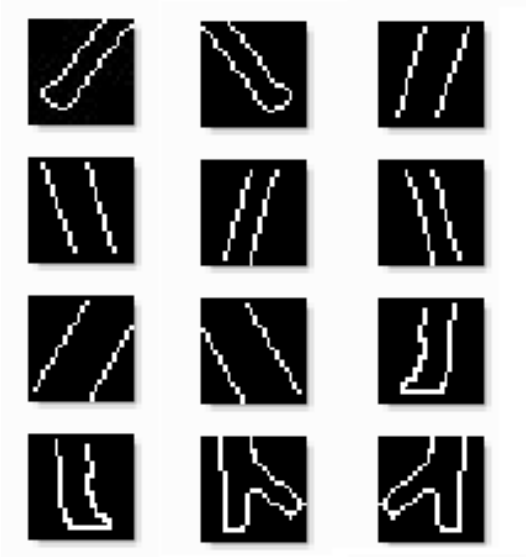
3.2 Alt Nesnelerin Tespiti

Bu çalışmada önerilen nesne tespit sistemi dolaylı arama, yani diğer bir deyişle aşağıdan-yukarıya yaklaşımını kullanmaktadır. Bu yaklaşıma göre hedef görüntü içerisinde bulunmak istenen asıl nesne yerine asıl nesneyi oluşturan alt nesneler aranır. Bu yöntemin özellikle yaya tespiti için birçok avantajı vardır. Bunlardan en önemlisi yayanın duruşuna karşı kazanılan esnekliktir. Yayanın duruşu, kol ve bacaklarının pozisyonları çok çeşitli şekillerde olabilir. Dolayısıyla yayayı bir bütün olarak ararken tüm bu olasılıkları da hesap etmek gerekir. Fakat yayayı bir bütün olarak aramak yerine yayayı oluşturan kol, bacak, yüz, kafa, el, gövde, ayak, vb. alt nesnelerin aranması bu güçlükleri azaltır. Bu yöntemin dezavantajı ise bulunan alt nesnelerden yararlanarak asıl nesneyi bulacak bir yöntemin tasarlanması gerekliliğidir. Yayayı bir bütün olarak arayan sistemlerde böyle bir düzenlemeye gerek yoktur [10], [40], [49].

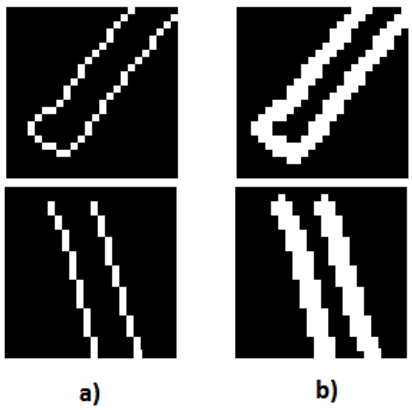
Dolaylı arama yönteminde hedef görüntüde bir yayaya ait olabilecek alt nesnelerin aranması amaçlanmaktadır. Bu nedenle, alt nesneleri aramada kullanılmak üzere bir eğitim kümesi hazırlanması gerekir. Eğitim kümesindeki her bir alt nesne örneği, ilgili alt nesnenin kenarlarından oluşan 24x24 piksellik bir görüntüdür. Bu örnekler gerçek görüntülerden alınarak veya elle çizilerek oluşturabilir (Şekil 3.2).

Eğitim seti oluşturulduktan sonra eğitim setindeki alt nesne örneklerinin öznitelikleri oran şablonu yöntemiyle Şekil 3.1'de gösterilen öznitelik şablonları kullanılarak hesaplanır. Bir alt nesne örneği hedef görüntüde aranırken, öncelikle hedef görüntünün kenar çizgileri bulunur ve birbirlerine belirli bir yakınlıkta olan dik ve dike yakın kenar çizgileri birleştirilir. Bu şekilde işlenen hedef görüntüden alınan blok ile aranan alt nesne bloğunun öznitelikleri arasındaki benzerlik karşılaştırılır. Ancak hedef görüntüden alınan bloğun arka planı hesaplanan öznitelikleri önemli oranda bozar. Bu etkiyi en aza indirmek için hedef bloğa aranan alt nesne örneği için oluşturulan bir maske uygulanır. Maskeleme görüntüleri alt nesne görüntülerinden kalınlaştırma

yapılarak elde edilir (Şekil 3.3). Böylece alt nesne resmi hedef görüntüde aranırken alt nesne resmini bulmayı engelleyen arka plandaki nesneler engellenir.

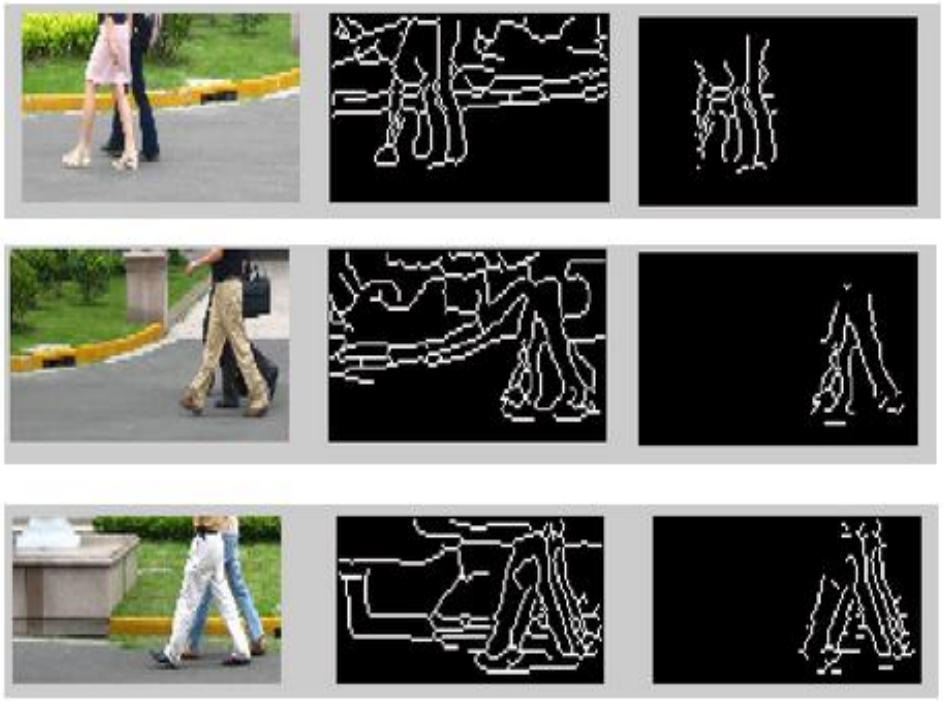


Şekil 3. 2 Bacak alt nesnesi için oluşturulan eğitim kümesindeki örnekler [39]



Şekil 3. 3 a) Bacaklar için kullanılan eğitim görüntüleri, b) Bu görüntülere ait maskeler [39]

Şekil 3. 4'te bacaklar için oluşturulan alt nesne örnekleri kullanılarak hedef görüntülerin alt kısımlarında tespit edilen bacaklar gösterilmektedir. Bunun için hedef görüntüde 24x24 piksellik bir pencere her adımda bir piksel ilerleyecek şekilde dolaştırılmış ve her adımda dikey, yatay ve çapraz öznitelikler bulunarak bu özniteliklerin eğitim kümesindeki özniteliklere ne derece benzediği ölçülmüştür. Belirli bir eşik değerinin geçilmesi durumunda mevcut adımdaki pencerenin bacak alt-nesnesini içerdiğine karar verilmiştir.

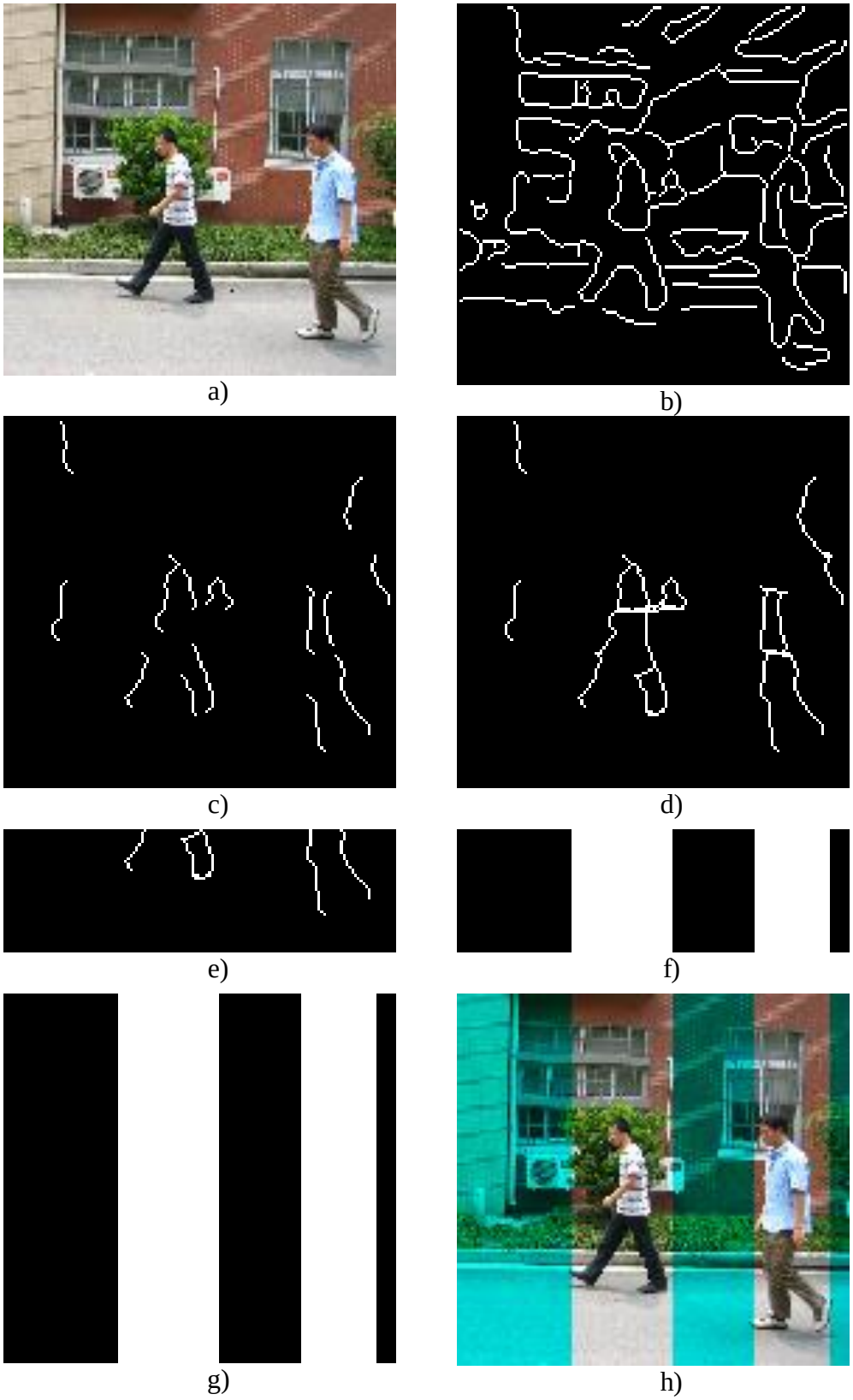


Şekil 3. 4 Bacaklar için oluşturulan eğitim kümesi kullanılarak tespit edilen bacaklar [39]

3.3 İlgi Bölgelerinin Tespiti

Yayayı oluşturan alt-nesnelerin tüm görüntüde aranması pratik bir yöntem değildir. Bunun için öncelikle hedef görüntüde yaya içerme olasılığı daha yüksek olan bölgelerin hızlı ve etkin bir şekilde tespit edilmesi gerekir. Böylece yaya tespit sisteminin sadece yaya olması olası bölgelere yoğunlaşması ve buraları detaylı bir şekilde taraması sağlanır.

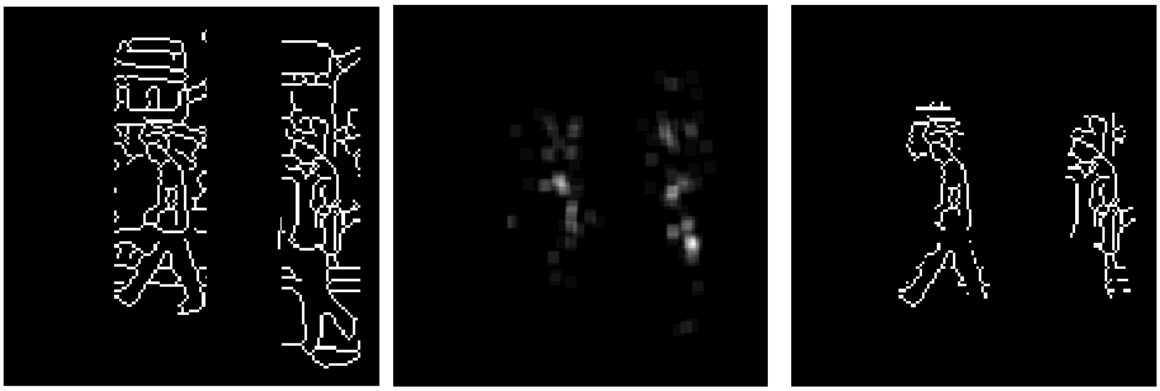
Yaya içermesi olası bölgelerin tespitinde kullanılabilecek en önemli bilgilerden biri bacaklardır. Bacaklar bir yayayı tespit için kullanılabilecek en ayırt edici ve belirleyici özelliklerdir. Bu çalışmada yayaların ayakta, yani dik konumda bulundukları varsayılmıştır. Dik duran insanların ve bu insanlara ait bacakların hedef görüntüde dik ya da dike yakın kenarlar oluşturması gerekir. Bunun yanısıra bacakların hedef resmin alt yarısında olması beklenir. Buna göre hedef resmin alt kısmında dik ya da diken yakın kenarlar içermeyen bölgeler elenerek yaya içermesi olası bölgeler (ilgi bölgeleri) hızlı bir şekilde tespit edilmiştir (Şekil 3.5).



Şekil 3. 5 İlgili bölgelerin tespiti. a) Özgün görüntü, b) Bulunan kenarlar, c) Yatay kenarların elenmesi, d) Çizgi birleştirme, e) 1/3'lük alt kısmın alınması, f) İlgi bölgelerini belirleyen maskenin alt kısmı, g) İlgi bölgelerini belirleyen maske, h) Özgün görüntü üzerinde ilgi bölgeleri [39]

3.4 Yaya Tespit Aşaması

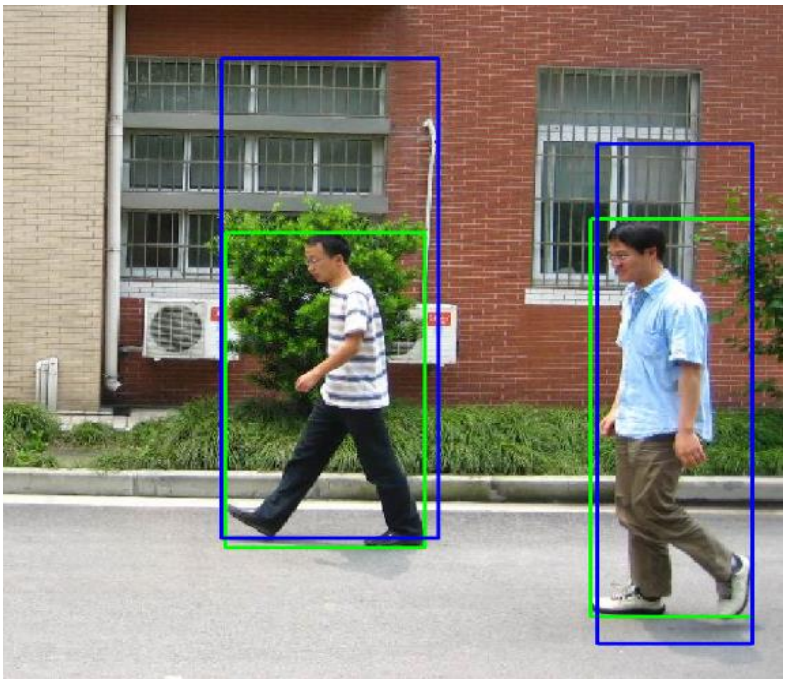
Yayanın bulunduğu varsayılan ilgi bölgeleri tespit edildikten sonra, bu bölgeler içinde yaya alt nesneleri aranmıştır. Hedef görüntüdeki alt nesneler bulunduktan sonra, asıl nesnenin bulunması gereklidir. Asıl nesnenin bulunabilmesi için asıl nesnenin merkezinin bulunan alt nesnelere göre olası yerini gösteren oylama vektörleri kullanılarak bir oy haritası oluşturulur. Oylama vektörleri alt nesne örnekleri oluşturulurken her bir örnek için eğitici tarafından sisteme girilmiştir. Hedef görüntüde bir alt nesne bulunduğuna karar verildiğinde oy haritası oluşturulurken o alt nesneye ait oylama vektörü kullanılır. En çok oy alan merkezler yaya olması en muhtemel bölgeler olarak kabul edilmiştir. Şekil 3. 6 b)'de böyle bir oy haritası görülmektedir. Bu merkezler Şekil 3.6'da beyaza yakın bölgeler olarak gösterilmiştir.



a) b) c)

Şekil 3. 6 Alt-nesne bilgilerinden asıl yayaların tespiti, a) İlgili bölgeler, b) Oylama Haritası, c) Tespit edilen yayalar [39]

Eğitim kümesindeki her bir alt nesne örneğinin hedef görüntüde aranması tamamlandıktan sonra oluşan oy haritasında bir eşik değerinden daha yüksek oy alan bölgeler tespit edilen yayaların merkezleri olarak kabul edilir. Bu merkezlere oy veren alt nesneler ilgili bölgelerdeki yayaların parçaları olarak kabul edilir. Asıl nesneye ait parçalar bulunduğunda bu parçaların sınırları yayanın bulunduğu bölgenin sınırları olarak işaretlenir (Şekil 3.7).



Şekil 3. 7 Bulunan yayalar. Yeşil (açık gri): referans yaya sınırları, Mavi (koyu gri): önerilen sistemin bulduğu sınırlar [39]

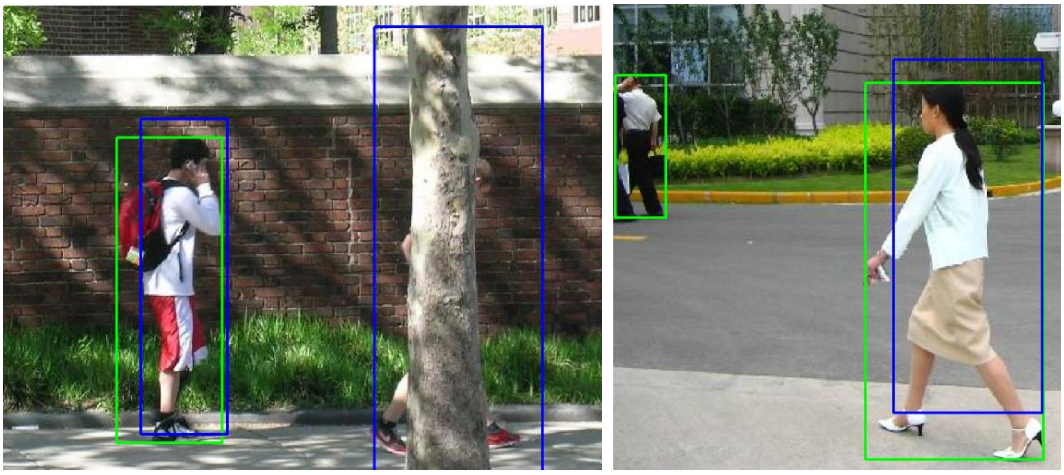
3.5 Test Sonuçları

Test veri kümesi olarak Pennsylvania Üniversitesi ve Fudan Üniversitesi tarafından oluşturulan Penn-Fudan Yaya Veri Kümesi'nden [43] alınmış 170 adet görüntü kullanılmıştır. Bu 170 görüntüde toplam 424 yaya mevcuttur. Veri kümesindeki görüntüler üniversitelerin yerleşkeleri ve civarlarındaki yerleşim yerlerinden elde edilmiştir. Her görüntünün içinde en az bir adet yaya mevcuttur.

Tetik ve Bolat [39] tarafından geliştirilen tespit sistemi, bir yayanın %60'ından daha fazlası veri kümesi tarafından işaretlenen bölge içerisinde kaldığında bu tespiti başarılı bir tespit olarak saymıştır. Aynı yaya 2 parça olarak tespit edildiğinde bu durum bir tespit olarak sayılmıştır. Yapılan testler sonucunda görüntülerde bulunan toplam 424 yayadan 327 adedi tespit edilebilmiştir. Buna karşılık yaya olmayan 34 nesne de yaya olarak değerlendirilmiştir. Sistem en fazla hatayı kalabalık görüntülerde ve özellikle ön planda diğerlerinden ayrı duran bir yayanın olduğu durumlarda yapmıştır. Ancak, sistem en öndeki yayayı her seferinde doğru olarak bulmuştur.

Bu bölümde anlatılan çalışmada [39] durağan görüntülerdeki yayaların oran-şablonu yaklaşımından yararlanarak dolaylı arama ile tespiti üzerinde durulmuştur. Yaya tespitinin ilk adımında görüntünün içerisinde yaya olması beklenen ilgi bölgeleri tespit edilmiş, daha sonra tespit edilen ilgi bölgeleri içerisinde yayalara ait alt nesnelerin yerleri bulunmuştur. Bulunan alt nesneler yardımıyla ilgi bölgesinin içinde yaya olup olmadığı ve varsa nerede olduğu hesaplanmıştır.

Sonuçlar incelendiğinde önerilen yaklaşımın ön plandaki yayaları tespit etmekte başarılı olduğu, ancak kalabalık görüntülerde geri planda kalan yayaları iyi ayırt edemediği görülmüştür. Bu durumun önemli nedenlerinden biri, ilgi bölgelerinin bulunmasında görüntünün alt üçte birlik bölgesinin kullanılmasıdır (Şekil 3.8). İlgi bölgelerinin görüntünün tamamını değerlendirilerek bulunmasının sistem başarımını arttıracağı öngörülmüştür.



a) b)

Şekil 3. 8 a). Ağaç yaya olarak tespit edilmiş, b). sol üstteki yaya ise ilgi bölgesi içerisinde kalmadığından tespit edilememiştir [39]

BÖLÜM 4

ÖZNİTELİK BULMA YAKLAŞIMIYLA DOĞRUDAN YAYA TESPİTİ

Bu bölümde anlatılan yaya tespit sisteminde, resimlerdeki yayalara özel nitelikler bulunmaya çalışılmış ve bu öznitelikler kullanılarak yayalar bir bütün halinde yani doğrudan resimlerden tespit edilmeye çalışılmıştır. Haar dalgacıklarına benzeyen ama çok daha hızlı hesaplanabilen bir öznitelik çıkarım yöntemi uygulanmış ve bu öznitelik çıkarım yöntemiyle hesaplanan çok sayıda özniteliği sınıflandırmak için uyarlanabilir arttırma ("Adaptive Boosting, AdaBoost") olarak adlandırılan bir yöntem izlenmiştir. Önerilen algoritmanın etkinliğini göstermek için eğitim için farklı bir veri kümesi (NICTA [42]) ve test için farklı bir veri kümesi (Penn Fudan [43]) kullanılmıştır. Elde edilen sonuçlar önerilen metodun etkinliğini doğrular niteliktedir.

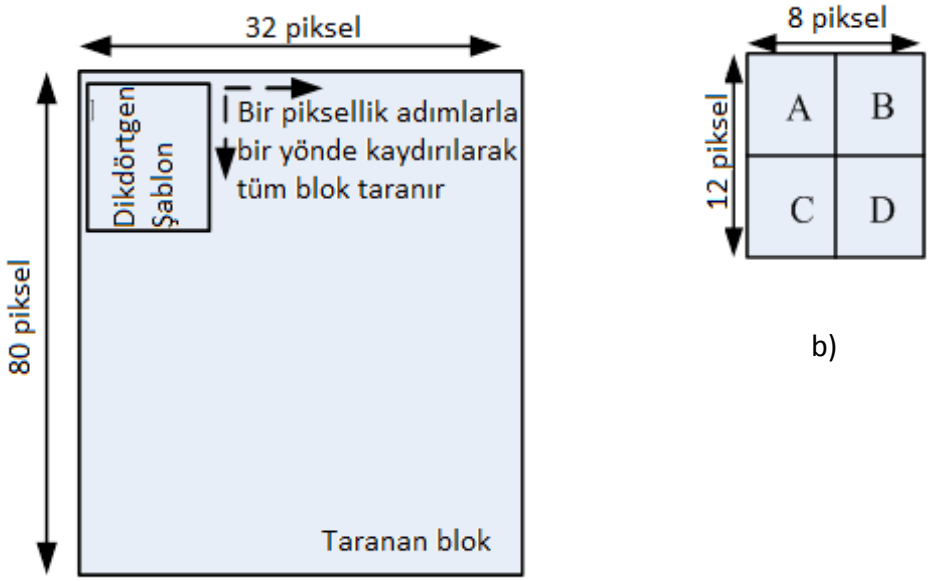
4.1 Özniteliklerin Hesaplanması

Papageorgiou ve arkadaşları [22], Haar benzeri dalgacık öznitelikleri kullanan bir yaya tespit sistemi önermiştir. Viola ve Jones [16], bu makaleden yola çıkarak Haar benzeri öznitelikler kullanan ve gerçek zamanlı çalışan bir yüz tespit sistemi geliştirmişlerdir.

Bu çalışmada [16]'daki özniteliklerden türetilen dikdörtgen öznitelikler kullanılmıştır. Öznitelikler, 32x80 boyutundaki bir bloğun 8x12 piksel boyutundaki şablon öznitelikler ile taranmasıyla hesaplanır. Şablon Şekil 4.1 b'deki gibi A,B,C ve D olarak adlandırılan dört alt dikdörtgensel alana bölünmüştür. Her bir alt alan için, o alanda kalan piksellerin toplamı o alanı ifade eden değeri verir. Taranan bloktaki her bir piksel için öznitelikler (4.1) - (4.8)'deki eşitliklerle hesaplanır.

Bu eşitlikler kullanılarak her bir blok için 10787 öznitelik hesaplanır. Alt alanlar arasındaki farklar yerine alt alanların birbirlerine oranlarının kullanılması hesaplanan

özniteliklerin ölçeğe göre değişmez olmasını sağlar, aynı zamanda aynı ölçekteki farklarla hesaplanan özniteliklere nazaran daha iyi bir öğrenme performansı vermektedir.



Şekil 4. 1 a) Dikdörtgen şablon kullanılarak öznitelik çıkarımı b) Dikdörtgen şablon

$$f(1) = (A + B) / (C + D + 1) \quad (4.1)$$

$$f(2) = (A + C) / (B + D + 1) \quad (4.2)$$

$$f(3) = (A + D) / (B + C + 1) \quad (4.3)$$

$$f(4) = A / (E + 1) \quad (4.4)$$

$$f(5) = B / (E + 1) \quad (4.5)$$

$$f(6) = C / (E + 1) \quad (4.6)$$

$$f(7) = D / (E + 1) \quad (4.7)$$

$$E = A + B + C + D \quad (4.8)$$

Sınıflandırmayı yüksek sayıda öznitelik kullanarak yapmak zorlu bir iştir. Özniteliklerin sayısının fazla olması hesaplamanın maliyetli ve sınıflandırmanın hataya meyilli olmasına neden olur. Öncelikle öznitelik sayısı arttıkça sınıflandırma için gerekli hesaplama zamanı da artar. İkinci olarak bu öznitelikler arasında sınıflandırmaya yardımcı olmayan ilgisiz öznitelikler olabilir ve bu öznitelikler sınıflandırıcının

sınıflandırma başarısını olumsuz yönde etkiler [39]. Bu iki tuzaktan da kaçınmak için, özniteliklere Adaboost algoritması [16] uygulanmıştır. Her bir öznitelik zayıf bir sınıflandırıcı elde etmek için kullanılır. Bir zayıf sınıflandırıcı %50'den farklı bir başarı oranıyla sınıflandırma yapabilen bir sınıflandırıcıdır. Zayıf sınıflandırıcıların karar verme sınırları pozitif ve negatif örneklerin ağırlıklı ortalaması hesaplanarak bulunur. Adaboost algoritması güçlü bir sınıflandırıcı elde etmek için bu zayıf sınıflandırıcıları birbirlerini en iyi tamamlayacak şekilde seçer. Çizelge 4.1'de bu çalışmada kullanılan Adaboost algoritmasının detayları verilmiştir.

Çizelge 4. 1 Adaboost algoritması [16]

- Eğitim örnekleri $(x_1, y_1),...,(x_n, y_n)$ olarak düzenlenir. Burada y_i, negatif örnekler için 0, pozitif örnekler için 1 olarak kabul edilir.
- m ve n sırasıyla pozitif ve negatif örnek sayısını göstermek üzere ağırlıklar $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ olacak şekilde her $y_i \in \{0,1\}$ için ilklenir.
- T iterasyon sayısı olmak üzere, her t=1,.,T için:
  1. Ağırlıklar normalize edilir.

  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

  2. Her bir j özniteliği için, sadece bu j özniteliğini kullanan bir h_j sınıflandırıcısı eğitilir. Hata w_t ağırlığına göre ölçülür.

  $$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

  3. En az ε_t hatasına sahip h_t sınıflandırıcısı seçilir.
  4. Ağırlıklar güncellenir:

  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

  Burada x_i doğru olarak sınıflandırıldıysa $e_i = 0$, aksi halde $e_i = 1$ olur. β_t ise $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ olarak hesaplanır.
- Sonunda oluşan sınıflandırıcı:

  $$h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) > \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & diger \end{cases}$$

  burada $\alpha_t = \log \frac{1}{1-\beta_t}$ olarak alınır.

4.2 Yayaların Tespiti

Adaboost sınıflandırıcısını eğitmek için Nicta [42] yaya veri kümesi kullanılmıştur. Bu veri kümesi 8x20, 16x20, 32x20, 32x80 ve 64x80 boyutlarında sırasıyla yaya içermeyen negatif (Şekil 4.2) ve içeren pozitif (Şekil 4.3) örneklerden oluşmaktadır. [42]'de büyük boyutların kullanılması önerilmiş olmakla birlikte 64x80 boyutundaki yaya içeren örneklerde arkaplan oldukça zengindir. Bu zengin arkaplan nedeniyle 64x80 boyutundaki veri örneklerinden yayaya ait karakteristiklerin çıkarılması göreceli olarak daha zordur. Bu nedenle eğitim için A veri kümesinde yer alan 32x80 boyutundaki örnekler kullanılmıştır. Bu veri kümesindeki 6000 pozitif ve 4000 negatif olmak üzere toplamda 10000 örnek eğitim için kullanılmıştır. Her bir örnek için bölüm 4.1'de anlatıldığı gibi öznitelikler hesaplanmıştır. Öznitelikler hesaplanırken kullanılan dikdörtgensel şablonun her zaman taranan blok içinde kalması sağlanmış, böylece bir blok için toplamda 10787 öznitelik hesaplanmıştır.

Bu öznitelikler kullanılarak her bir öznitelik için bir basit eşik değer sınıflandırıcısı olacak şekilde 10787 adet zayıf sınıflandırıcı oluşturulmuştur. K. zayıf sınıflandırıcı, verilen örneğe ait k. öznitelik eşik değerinden büyükse pozitif, değilse negatif kararı vermektedir. Zayıf sınıflandırıcıların eşik değerleri eğitimde kullanılan pozitif ve negatif örneklerin o öznitelik için hesaplanan değerlerinin ağırlıklı ortalaması alınarak bulunur.

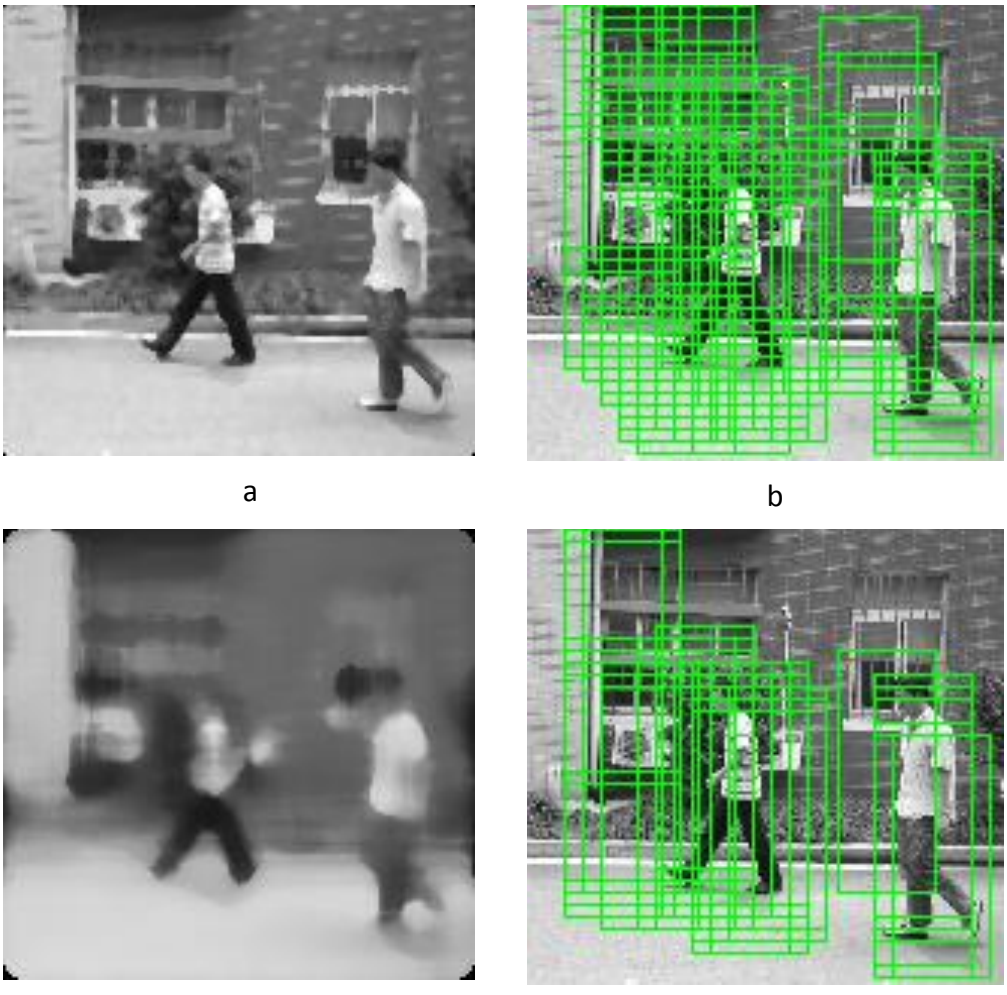
Şekil 4. 2 Negatif örnekler

Şekil 4. 3 Pozitif örnekler

Adaboost algoritması ilk olarak Freund ve Schapire [50] tarafından önerilmiştir. Algoritmanın amacı, eğitim örnekleri üzerinden verilen D dağılımına bağlı olarak düşük hata oranına sahip nihai bir hipotez oluşturmaktır. [16]'daki çalışmada, Adaboost algoritmasının [51]'de önerilen orijinal formunun daha pratik bir hali uygulanmıştır. Önerilen metod düzgün ("uniform") bir dağılımla çalışmaya başlar ve her adımda o adımdaki sınıflandırma performanslarına göre en iyi zayıf sınıflandırıcı seçilir. Ağırlıklar (w_j) normalize edilerek olasılık dağılım fonksiyonu elde edilir. Her adımda elde edilen olasılık dağılım fonksiyonu kullanılarak en iyi zayıf sınıflandırıcı seçilir ve bu sırada ağırlıklar Adaboost algoritmasına öngördüğü şekilde güncellenir. Böylece yeni adıma farklı ağırlıklar ve farklı bir olasılık dağılım fonksiyonuyla başlanır.

Yapılan denemelerde Adaboost algoritması 6x4'ten 60x80'e kadar değişen boyutlardaki şablonlardan üretilen birçok dörtgensel öznitelik kullanılarak eğitilmiş ve en iyi performansın 8x12 boyutundaki şablonlarla elde edildiği gözlenmiştir.

Eğitim aşamasında gerçekleştirilen 2000 iterasyonun ardından Adaboost algoritması kullanılan 10787 öznitelikten 696 tanesini en ilgili olarak seçmiştir. Bu noktada, Adaboost algoritması eğitim setindeki yayaların %95.1'ini doğru tespit etmektedir. İlk bakışta bu performans oldukça tatmin edici olarak görülebilir. Fakat 800x600 piksel boyutunda bir resim için yaya barındırması muhtemel 404.000 adet blok bulunur. %95.1 doğrulukla, Adaboost algoritması 19796 (404.000 x %4.9) hatalı tespit yapabilir. Hatalı tespitlerin sayısını azaltmak için resimlere 10x10 boyutunda ortanca filtresi uygulanmıştır. Ortanca filtresi uygulandığında hatalı tespit oranlarının büyük oranda düştüğü gözlenmiştir (Şekil 4.4). Buna rağmen halen oldukça fazla sayıda hatalı tespit olduğu Şekil 4.4'ten görülebilir. Bu hatalı tespitleri de engelleyerek hatalı pozitif sayısını azaltmak için sırasıyla iki işlem daha uygulanmıştır.

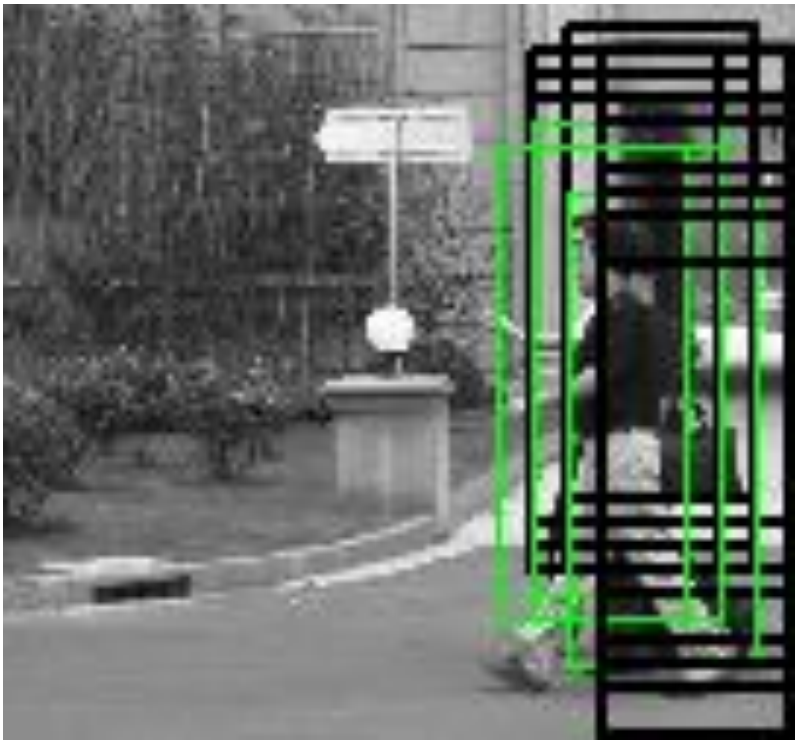


Şekil 4. 4 Ortanca filtresinin etkisi a. Orijinal Resim, b. Tespit Edilen Alanlar. c. Ortanca filtrelenmiş resim, d. Ortanca filtrelemenin ardından tespit edilen alanlar [21]

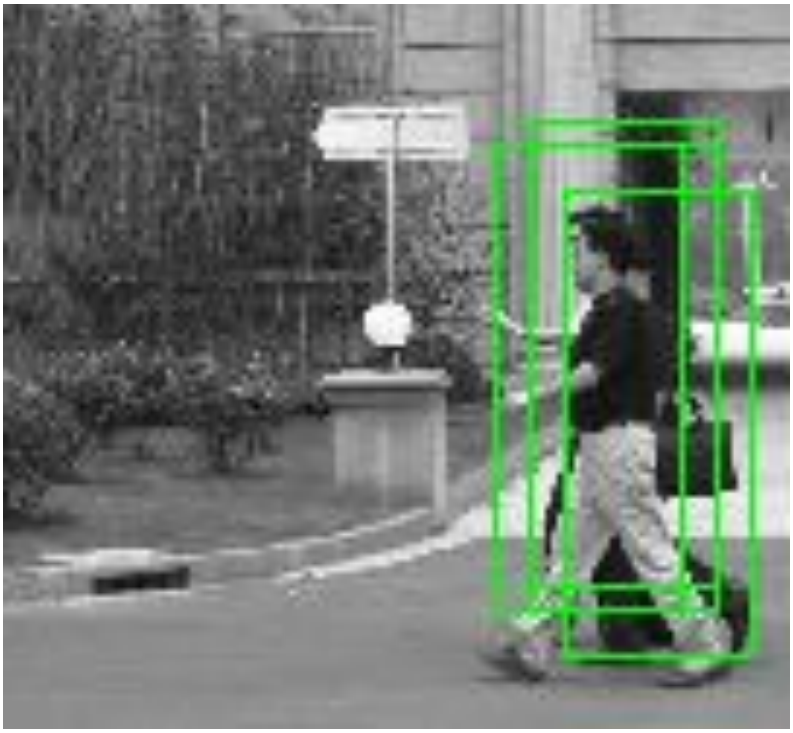
Birinci işlemde, tespit edilen alanlara dikey bir Sobel kenar bulma filtresi uygulanarak tespit edilen alanın bacak içerip içermediğini kontrol edilmiştir [39]. Bacaklar gövdenin dik ya da dike yakın uzantıları olduğundan, alt bölümü dikey kenarlar içeren aday bölgeler yaya adayı olarak seçilmiş, diğerleri elenmiştir (Şekil 4. 5).

Şekil 4. 5 Sobel filtresinin uygulanmasının ardından yaya aday bölgeleri. Kalın siyah dikdörtgenle gösterilen aday bölge Sobel filtresi tarafından elenmiş durumda [21]

İkinci işlemde ise, insan gövdesinin simetrik olmasından yararlanılmıştır. Buna göre tespit edilen aday bölgede eğer bir insan vücudu varsa simetrik kenarlar olmalıdır. Bu nedenle aday bölge dikey olarak iki eşit dikdörtgene ayrılmıştır. Sobel filtresinin sonucu ikili sistemde olduğundan, sağ ve soldaki dikdörtgenlerdeki kenar piksellerinin toplamı kenarların simetrikliği hakkında bir fikir verebilir. Aday bölgede simetrik kenarlar bulunması durumunda bölgenin ortadan sağ ve sol olarak ayrılmasıyla oluşan iki alt bölgesindeki kenar pikselleri toplamı oranlarının yaklaşık olarak 0.5 ile 1.5 arasında değiştiği gözlenmiş ve bu bilgi ışığında simetriklik oranı bu değerler arasında olan bölgeler yaya içeriyor (Şekil 4.6) olarak kabul edilmiştir.



a

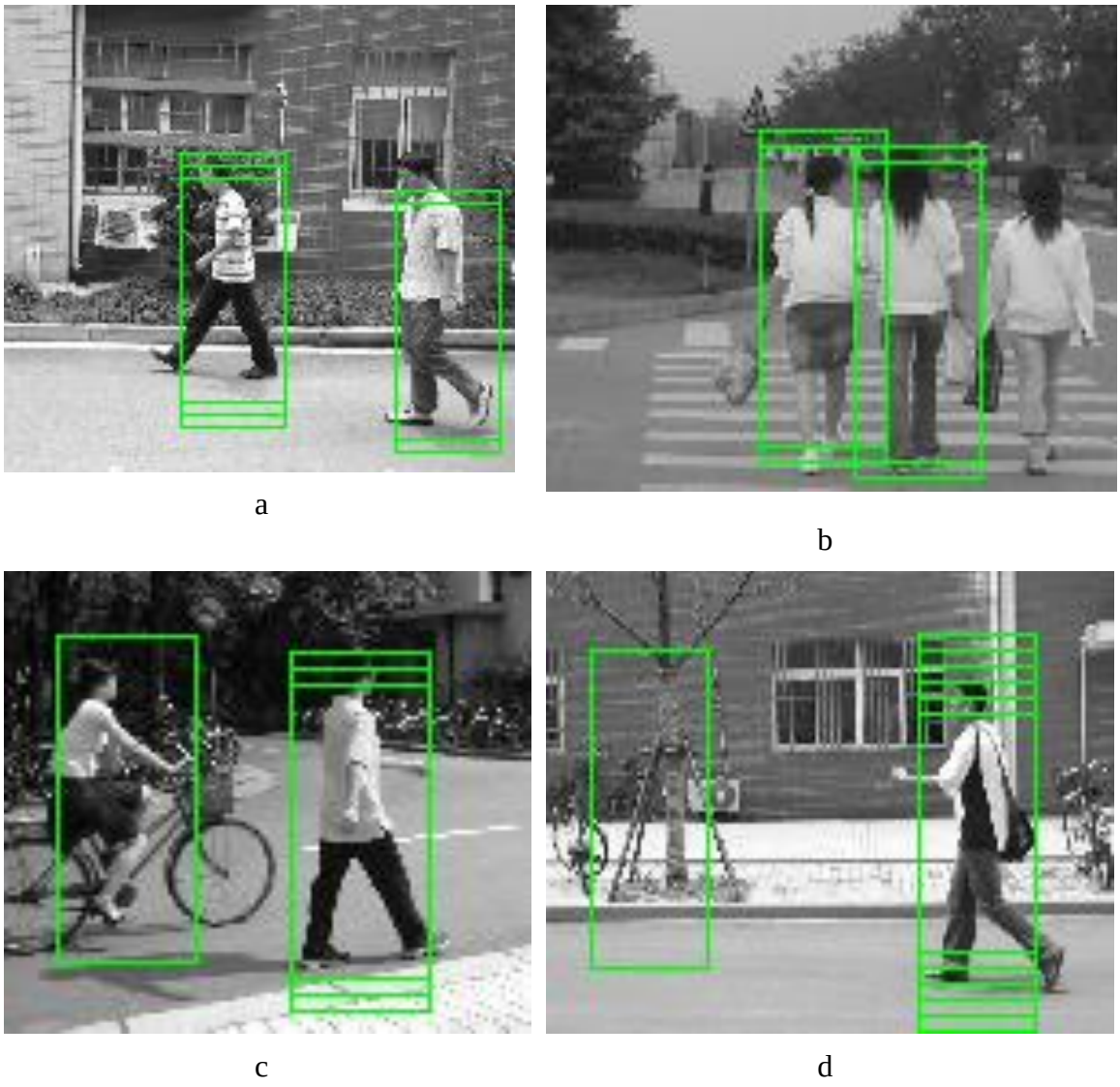


b

Şekil 4. 6 Simetri filtresinin etkisi. a. Kalın dikdörtgenlerle işaretlenen aday bölgeler simetri filtresi tarafından eleniyor. b. Simetri filtresinin uygulanmasının ardından kalan bölgeler [21]

4.3 Test Sonuçları

Bu iyileştirmelerin ardından oluşturulan nihai sınıflandırıcı Penn Fudan veri kümesiyle test edilmiştir [43]. Penn Fudan veri kümesinde, 423 adet işaretlenmiş yaya bulunduran 170 adet resim bulunmaktadır. Resimler sokaktan ve kampüs çevresinden çekilen sahnelerden oluşmaktadır. Her resim içerisinde en az bir adet yaya bulunur. Bu veri setindeki yayaların boyutları [180,390] pikseli geçmemekte ve işaretlenen yayaların hepsi ayakta durur şekilde bulunmaktadır. Testler sonucunda bulunan yayalar Şekil 4.7'de gösterilmiştir.



a b c d

Şekil 4. 7 Örnek çıktılar. a. Şekil 6.nin sonuç çıktısı b. yanlış negatif örneği: en sağdaki kız elindeki çanta simetrisini bozduğundan dolayı tespit edilememiş. c. yanlış pozitif örneği: bisiklet üzerindeki kız veri kümesinde işaretlenmemiş olmasına rağmen yaya olarak bulunmuş. d. Yanlış pozitif örneği: soldaki ağaç yanlışlıkla yaya olarak seçilmiş

Yapılan testler sonucunda veri kümesinde işaretlenen yayalardan 357 tanesi (%84.4) doğru şekilde tespit edilmiş, bunun yanısıra 23 adet bölge yaya içermemesine rağmen yaya olarak işaretlenmiştir. 66 adet yaya ise veri kümesinde işaretlenmiş olmasına rağmen tespit edilememiştir.

BÖLÜM 5

YAYA TESPİTİ İÇİN GENEL BİR ÇERÇEVE OLUŞTURULMASI

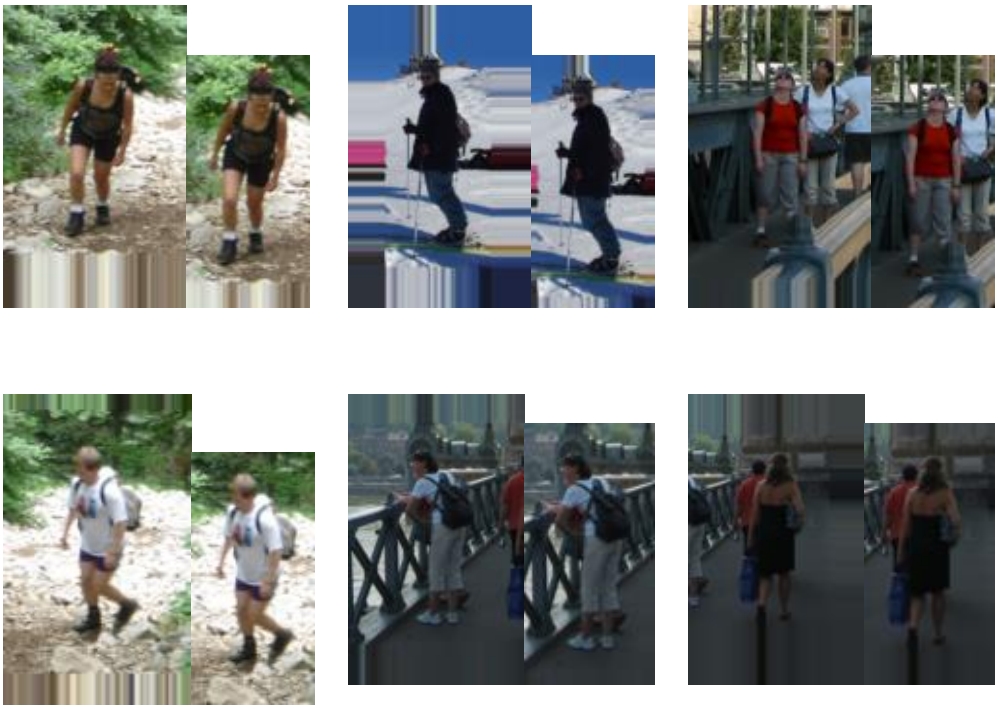
Bu bölümde çeşitli öznitelik bulma yöntemleri ve sınıflandırıcıların birlikte kullanılabilmesine imkân veren genel bir yaya tespit çerçevesi anlatılmaktadır. Bu çerçeve kullanılarak farklı öznitelikler, farklı öznitelik sınıflandırıcılar ve farklı ana sınıflandırıcılar birlikte denenebilir. Bu şekilde oluşturulan çeşitli yaya tespit sistemlerinin performansları birbirleriyle karşılaştırılabilir. Bu çerçeve, öznitelik kullanımına dayalı ve doğrudan yaklaşımı kullanan nesne tespit sistemleri oluşturmaktadır.

5.1 Veri Kümesinin Hazırlanması

Önerilen yaya tespit çerçevesi, literatürdeki veri kümelerinden herhangi biriyle eğitilebilir ve test edilebilir. Fakat birçok yaya tespit sistemi, INRIA veri kümesini [25] tercih ettiği için, önerilen çerçevenin de bu veri kümesiyle nasıl kullanabileceği bu kısımda anlatılmaktadır.

Önerilen çerçeveyle oluşturulan yaya tespit sistemlerinin eğitimi ve testi için sırasıyla INRIA veri kümesindeki “train_64x128_H96” ve “test_64x128_H96” klasörleri kullanılmıştır. INRIA veri kümesi, 64x128 piksel boyutunda ve yaklaşık 96 piksel yüksekliğinde yaya içeren pozitif resimler ve hiç yaya içermeyen negatif resimlerden oluşmaktadır. Pozitif resimlerin boyutu 64x128 piksel olduğundan resimlere sağ ve soldan 16’şar piksel ve yukarı ve aşağıdan 31’er piksellik ek yapılarak boyutları 96x160 piksel olarak ayarlanmıştır (Şekil 5.1). Bunun nedeni, bu resimlerin pencere dışına taşmayı gerektiren öznitelik bulma yöntemleriyle kullanılması durumunda taşmanın ek

üzerinde kalmasını sağlamaktır. Diğer bir deyişle, kullanılan öznitelik yöntemi pencere dışına taşmayı gerektirmiyorsa, ekler kırpılarak resimler orijinal boyutlarına getirilebilir.



Şekil 5. 1 INRIA veri kümesindeki 96x160 piksel boyutundaki pozitif eğitim örnekleri ve özgün halleri.

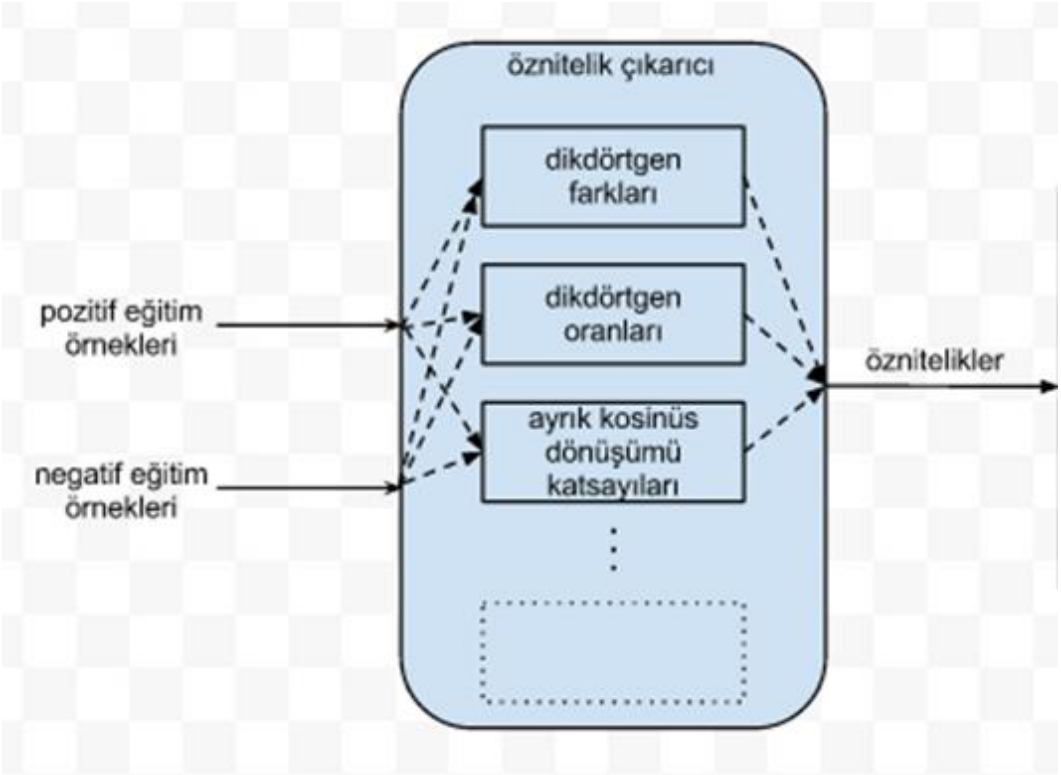
INRIA veri kümesindeki negatif resimler, hiçbir yayanın olmadığı dış ortam çekimlerinden oluşan 320x240 piksel boyutundaki resimlerdir. Negatif eğitim örnekleri, bu negatif resimlerin istenilen boyutta kesilmesiyle üretilir. Örneğin, kullanılan pozitif örneklerle aynı boyutta olan negatif örnekler üretebilmek için bu resimlerden 64x128 piksellik parçalar kesilerek negatif eğitim örnekleri oluşturulmuştur (Şekil 5.2).

Şekil 5. 2 INRIA veri kümesinden negatif bir resim örneği (solda) ve bu resimden üretilen 64x128 piksel boyutunda negatif eğitim örnekleri (sağda).

5.2 Özniteliklerin Bulunması

Eğitim örnekleri Bölüm 5.1'de anlatıldığı gibi 64x128 piksel boyutundaki resimlerden oluşmaktadır. Resmi oluşturan piksel değerleri doğrudan ya da dolaylı olarak kullanılarak resmin öznitelikleri hesaplanır. Örneğin, resmi oluşturan piksel değerlerinin ortalaması (ya da varyansı) bu resmi karakterize eden bir öznitelik olarak kullanılabilir.

Öznitelik bulma yöntemi, pozitif ve negatif eğitim örneklerine aynı şekilde uygulanarak bulunan özniteliklerdeki ayırt edici kriterler belirlenir. Şekil 5.3'de de gösterildiği gibi öznitelik çıkarıcı modül, öznitelik bulma yöntemlerinden birini veya birkaçını birarada kullanarak tüm eğitim örnekleri için öznitelikleri hesaplar. Bu öznitelikler, kullanılan öznitelik bulma yöntemi ve tekrar hesaplanabilmeleri için gerekli tüm bilgilerle birlikte saklanır.

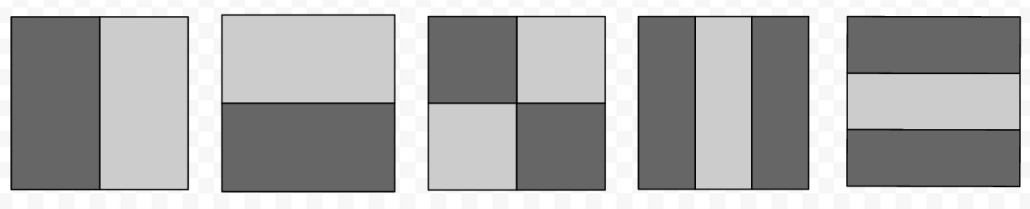


Şekil 5. 3 Tüm eğitim örnekleri için istenilen türde öznitelikleri çıkararak listeleyen modül.

5.2.1 Dikdörtgen Farkları

Bu öznitelik bulma yöntemi ilk defa Viola ve Jones [16] tarafından önerilmiş ve ardından en çok kullanılan öznitelik bulma yöntemlerinden biri haline gelmiştir. Dikdörtgen farkları yöntemi, esasen Papageorgiou ve Poggio [22] tarafından geliştirilen Haar dalgacık öznitelikleri bulma yöntemine benzemektedir. Fakat dikdörtgen farkları integral resimler [16] üzerinden hızlı bir şekilde hesaplanabilmektedir.

Bu öznitelik bulma yönteminde, dikdörtgen şeklindeki ve çeşitli boyutlardaki şablon denilen parçalar resim üzerinde dolaştırılarak her konumda yeni bir öznitelik değeri hesaplanır. Bu çalışmada, Şekil 5.4'de gösterilen dikdörtgen şablonlar kullanılmıştır. Kullanılan şablonların sayısı ve çeşitliliği arttırılabilir.



Şekil 5. 4 Dikdörtgen öznitelik şablonları.

Dikdörtgen farkları yöntemi, kullanılan öznitelik şablonundaki koyu bölgeye denk gelen piksellerin toplamından açık bölgedeki piksellerin toplamının çıkarılması sonucu elde edilen değerin bir öznitelik sayılması şeklinde kullanılır.

Bu çalışmada bir şablonun en küçük boyutu 24x24 piksel olarak tasarlanmıştır. Bir şablon öznitelikleri bulunacak resim içerisinde her adımda şablonun genişliğinin 1/3'ü kadar kaydırılır. Tüm resim bu şekilde tarandıktan sonra şablonun boyu 24'er piksel arttırılarak uzatılır ve tekrar tüm resim üzerinde dolaşması sağlanır. Bu işlem şablonun boyu resme sığdığı sürece tekrarlanır. Daha sonra aynı işlemler şablonun eni 24 piksel arttırılarak tekrarlanır. Böylece 64x128 piksellik her bir veri kümesi örneği için bu 5 adet şablon kullanılarak 690 adet öznitelik bulunur. Öznitelik sayısı, eğitim ve testlerin hızlı yapılabilmesi için az sayıda tutulmuştur. Benzer çalışmalarda, örneğin [52]'de 20x15 piksellik bir örnek resim için 24328 öznitelik hesaplanmıştır. [7]'de ise 134621 öznitelik hesaplanmaktadır.

5.2.2 Dikdörtgen Oranları

Dikdörtgen oranları yöntemi temelde dikdörtgen farkları yöntemiyle aynıdır. Dikdörtgen oranları yöntemi, Şekil 5.4'de görülen şablonlar kullanılırken koyu bölgelerin açık bölgelerden çıkarılması yerine oranlarının kaydedilmesini önerir [21].

Dikdörtgen oranlarının, dikdörtgen farklarına karşı en büyük avantajı farklı şablon boyutları için kullanıma uygun olmasıdır. Bir resimde yaya aranırken eğitimde kullanılan örneklerle aynı boyuttaki yani 64x128 piksel boyutunda bir pencere tüm resmi tarar ve yaya içeren bölgeleri tespit eder. Fakat bu şekilde sadece 64x128 piksel boyutundaki yayalar tespit edilebilir. Bundan daha küçük ya da daha büyük boyutlardaki yayaların tespiti pencere boyutu değiştirilerek yapılmak istenirse, pencere içerisinde kalan resmin özniteliklerini bulmaya yarayan şablonun boyutunun da aynı oranda değiştirilmesi gerekir. Böylece örneğin 32x64 boyutundaki resmin özniteliklerini ararken 24x24 yerine 12x12 piksellik bir şablonla başlanır ve her adımda 24 piksel yerine 12 adım ilerlenir. Diğer bir ifadeyle şablonların boyutu da yarı yarıya küçültülmüş olur. Bu şekilde 32x64 piksellik resim için de 64x128 piksellik resimde bulunan sayıda öznitelik hesaplanmış olur. Bu iki farklı boyuttaki resmin öznitelikleri dikdörtgen oranları yöntemi kullanılarsa birbirleriye uyumlu olur ve dolayısıyla şablon boyutu değiştirildiğinde de başarılı karşılaştırmalar yapılabilir.

5.2.3 Ayrık Kosinüs Dönüşümü Katsayıları

Ayrık kosinüs dönüşümü katsayıları yöntemi özellikle resim ve video sıkıştırmada çok kullanılan kosinüs dönüşümüyle bulunan katsayıların kullanımını öngörür. Bu yöntem de daha önce anlatılan öznitelik bulma yöntemlerinde olduğu gibi öznitelikleri bulunmak istenen resim üzerinde bir şablon dolaştırılmasını gerektirir. Fakat bu kez şablon bir karedir, alt dikdörtgenler içermez. Her adımda şablona denk düşen resim bölgesine eşitlik 5.1'deki ayrık kosinüs dönüşümü (AKD) uygulanarak dönüşüm katsayıları bulunur. Bu katsayılardan ilk 9 tanesi, yani kosinüs matrisinin sol üst köşesinde kalan 3x3'lük alan içerisindeki katsayılar öznitelik değerleri olarak kaydedilir.

$$X_{k,l} = \sum_{m=0}^{N_1-1} \sum_{n=0}^{N_2-1} x_{m,n} \cos\left(\frac{\pi k(m+0.5)}{N_1}\right) \cos\left(\frac{\pi l(n+0.5)}{N_2}\right) \quad (5.1)$$

Önceki yöntemlerde olduğu gibi örnek resim öncelikle 24x24 piksel boyutunda bir şablonla taranmaya başlanır. Resim içerisinde her seferinde şablon genişliğinin 1/3'ü kadar kayacak şekilde ilerler ve bir satır sonunda boyunun 1/3'ü kadar aşağıdaki satırdan devam eder. Tüm resim tarandıktan sonra şablonun eni ve boyu 24'er piksel arttırılarak aynı işlem tekrarlanır. Bu işlem şablon resim içerisine sığdığı sürece tekrarlanır. Bu şekilde 64x128 boyutunda bir resim için 630 öznitelik hesaplanır.

Yukarıda özetlenen öznitelik gruplarından hangisi veya hangilerinin daha başarılı olduğunu belirlemek üzere INRIA eğitim veri kümesinden 1208 pozitif ve 1208 negatif örnek alınmış ve ağırlıklı ortalama öznitelik sınıflandırıcılı Adaboost kullanılarak eğitim yapılmıştır. Test içinse yine INRIA test veri kümesinden 1126 pozitif ve 1126 negatif resim kullanılmıştır. Ana sınıflandırıcının maksimum yineleme sayısı 600 olarak seçilmiştir. Elde edilen sonuçlar Çizelge 5.1.'de özetlenmiştir.

Çizelge 5. 1 Öznitelik bulma yöntemlerinin karşılaştırılması

	Kaçırma oranı	Hatalı pozitif oranı
Dikdörtgen farkları	0.2052	0.0399
Dikdörtgen oranları	0.1261	0.0674
AKD katsayıları	0.1980	0.0559
D. oranları + D. farkları	0.1456	0.0550
D. oranları + AKD katsayıları	0.1172	0.0746
D. farkları + AKD katsayıları	0.1873	0.0630
D. farkları + AKD katsayıları + D. oranları	0.1412	0.0746

5.3 Özniteliklerin Sınıflandırılması

Bölüm 5.2'de anlatılan öznitelik bulma yöntemleri kullanılarak hesaplanan her bir öznitelik için bu bölümde anlatılan yöntemlerle bir sınıflandırıcı inşa edilir. Bu sınıflandırıcılar zayıf sınıflandırıcı ya da öznitelik sınıflandırıcısı olarak adlandırılır.

Veri kümesindeki tüm eğitim örneklerine ait öznitelikler hesaplanarak bir tabloda tutulur. Bu tablonun her bir satırı bir örnek için hesaplanan özniteliklerden oluşmaktadır. Örneğin, 200 pozitif ve 200 negatif örnek için dikdörtgen farkları yöntemine göre özniteliklerin hesaplanması halinde tek bir örnek için (5.2.1'de belirtildiği gibi) 690 öznitelik hesaplanır. Bu tablonun yanısıra bir özniteliğin nasıl hesaplandığını gösteren öznitelik rehber listesi tutulur. Bu rehberde hesaplanan özniteliğin kaçıncı öznitelik olduğu bilgisi indeks olacak şekilde Çizelge 5.2'deki şekilde bilgiler tutulur. Böylece indeks numarası verilen öznitelik için hangi boyut ve pozisyondaki şablon ve hangi öznitelik hesaplama metodu kullanılarak hesaplama yapıldığı bilgisine ulaşılabilir.

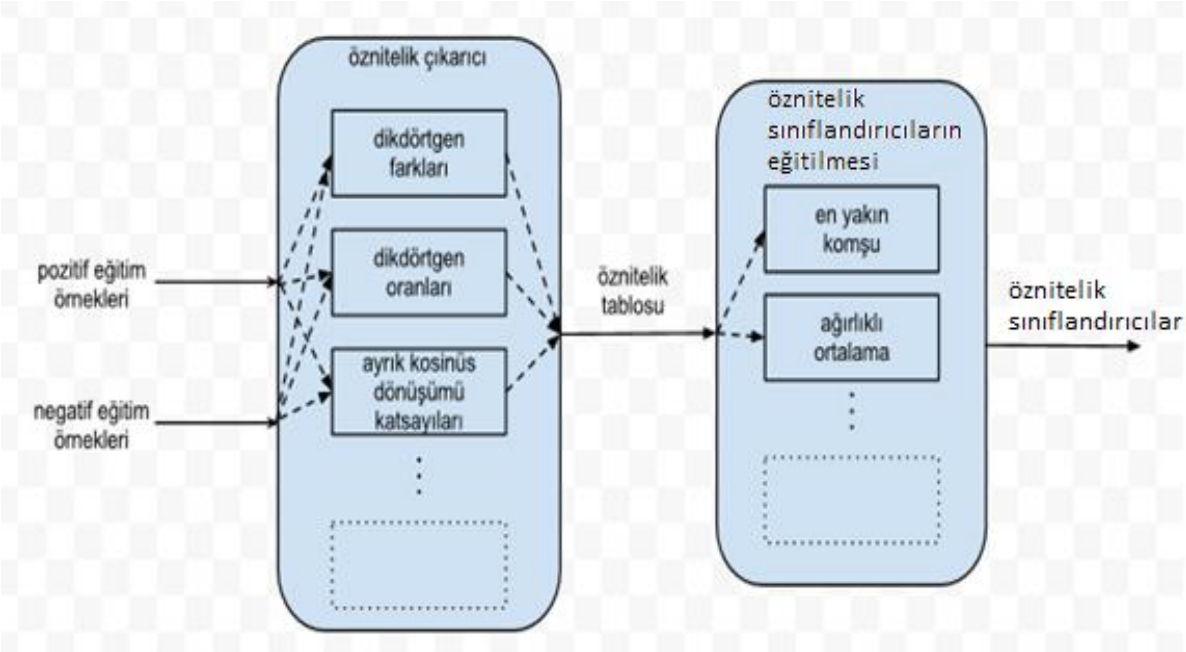
Çizelge 5. 2 Öznitelik rehber listesinde bir kayıt için tutulan bilgiler

indeksi	n
türü	dikdörtgensel fark
Boyutu	boy , en
konumu	x. satır, y. sütun

Öznitelik sınıflandırıcılarını oluşturmak için öznitelik tablosu ve öznitelik rehber listesi kullanılır (Şekil 5.5). Öznitelik tablosu, öznitelik sınıflandırıcıların eğitimi için kullanılmaktadır. Öznitelik rehber listesi ise eğitim aşamasında seçilmiş bir özniteliğin test aşamasındaki değerinin anında hesaplanabilmesi için kullanılmaktadır.

5.3.1 Ağırlıklı Ortalama Sınıflandırıcısı

Ağırlıklı ortalama sınıflandırıcısı basit bir eşik değer sınıflandırıcısıdır. Eğitim esnasında bu eşik değer ve eğitme verisindeki örneklerin etiketleri belirlenir. Bir öznitelik için sınıflandırıcının eşik değeri belirlenirken o öznitelik için pozitif örneklerden hesaplanan değerler ile negatif örnekler için hesaplanan değerlerin ayrı ayrı ortalaması alınır. Bu iki ortalamanın ortalaması eşik değer olarak kabul edilir.



Şekil 5. 5 Öznitelik Sınıflandırıcıların eğitilmesi.

5.3.2 En Yakın Komşu Sınıflandırıcısı

Ağırlıklı ortalama sınıflandırıcının en büyük dezavantajı sadece doğrusal olarak ayrıştırılabilen durumlar için başarılı sonuçlar vermesidir. Fakat eğitme verilerinin doğrusal olarak ayrıştırılabileceğini garantileyen herhangi bir ön koşul yoktur. Bu nedenle doğrusal olmayan bir sınıflandırıcının daha iyi sonuçlar vermesi beklenir.

Eğitme verisinin doğrusal ayrıştırılabilir olmayacağı da göz önüne alınarak Adaboost algoritmasındaki basit sınıflandırıcılar en yakın k komşu (kNN) sınıflandırıcısı ile değiştirilmiştir. Bu sınıflandırıcının eğitimi sırasında yapılan işlem bir öznitelik için tüm pozitif ve negatif eğitim örneklerinden elde edilen değerlerin bir listeye kaydından ibarettir. Test esnasında yeni bir değer geldiğinde bu değer listede bulunan kendisine en yakın k değerle karşılaştırılarak bu k değerin içinde en fazla örneğe sahip sınıfa dahil edilir. Ağırlıklı ortalama ve en yakın 2 komşu (2-EYK) sınıflandırıcılarının ana sınıflandırıcının performansına etkisi Çizelge 5.3'de gösterilmektedir. Kaçırma oranı ve hatalı pozitif oranı arasında yaklaşık olarak doğrusal bir ters orantı olduğu düşünülürse en yakın komşunun beklendiği gibi daha başarılı bir sınıflandırıcı olduğu sonucu çıkarılabilir.

Çizelge 5. 3 Öznitelik sınıflandırıcı türlerinin ana sınıflandırıcının performansına etkisi

	Kaçırma oranı	Hatalı pozitif oranı
Ağırlıklı ortalama	0.2095	0.0390
2-EYK	0.2149	0.0266

5.4 Hata Matrisinin Oluşturulması

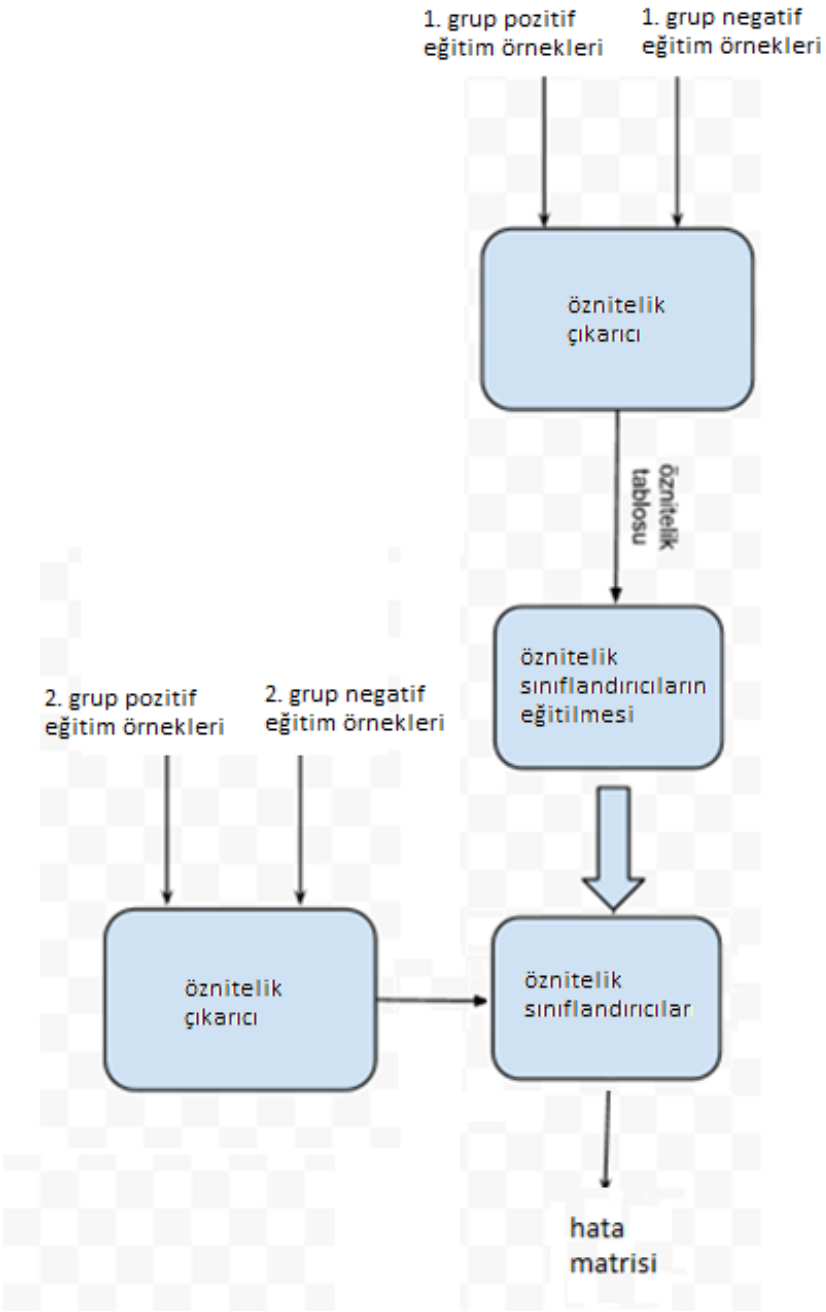
Hata matrisi, öznitelik sınıflandırıcıların test örneklerini doğru sınıflayıp sınıflayamadıklarını gösteren bir tablodur. Bu matrisin her bir sütunu bir öznitelik sınıflandırıcının test örneklerine verdiği yanıtları gösterir. Eğer i. sütunun j. elemanı 0 ise, i. sınıflandırıcı j. örneği doğru, 1 ise yanlış sınıflamıştır. Matrisin bir satırı ise bir örneğe karşı tüm öznitelik sınıflandırıcılarının verdiği yanıtları göstermektedir.

Hata matrisi, öznitelik sınıflandırıcılarının eğitilmesinde kullanılandan farklı bir test kümesi kullanılarak oluşturulmaktadır (Şekil 5.6). Çünkü öznitelik sınıflandırıcıların eğitiminde kullanılan örnek kümesinin kullanılması halinde matris eğitimin başarısını gösteren bir tablo olacaktır. Eğer öznitelik sınıflandırıcısı olarak kNN kullanılırsa, sınıflandırıcı doğası gereği tüm eğitim örneklerini ezberlediği ve doğru yanıt vereceği için anlamlı bir hata matrisi oluşmaz.

Hata matrisinin hesaplanabilmesi için öncelikle eğitim kümesinden farklı olarak seçilen test kümesinin öznitelikleri hesaplanır. Test örnekleri için elde edilen öznitelikler öznitelik sınıflandırıcılar ile sınıflandırılır ve hata matrisinde üzerinde çalışılan örneğin ilgili öznitelik sütununa doğru sınıflandırılması durumunda 0, yanlış sınıflandırılması durumunda 1 ya da -1 değeri atanır. İncelenen örnek pozitif olduğu halde sınıflandırıcı tarafından negatif olduğuna karar verirse bu bir kaçırma ya da hatalı negatif olur ve hata matrisinin ilgili gözüne 1 değeri yazılır; örnek negatif olduğu halde sınıflandırıcı pozitif olduğuna karar verirse bu bir hatalı alarm ya da hatalı pozitif olur ve -1 olarak işaretlenir (Şekil 5.7). Her bir öznitelik sınıflandırıcısının hata oranı kendine ait sütundaki sayıların mutlak değerlerinin toplamının örnek sayısına bölümüne eşittir.

Hata matrisinin elemanları x_{ji}, örnek sayısı n olmak üzere i. öznitelik sınıflandırıcısının performansı (5.2)'deki gibi hesaplanır:

$$performans(i) = 1 - \frac{\sum_{i=1}^{n} |x_{ji}|}{n} \tag{5.2}$$



Şekil 5. 6 Hata matrisinin oluşturulması

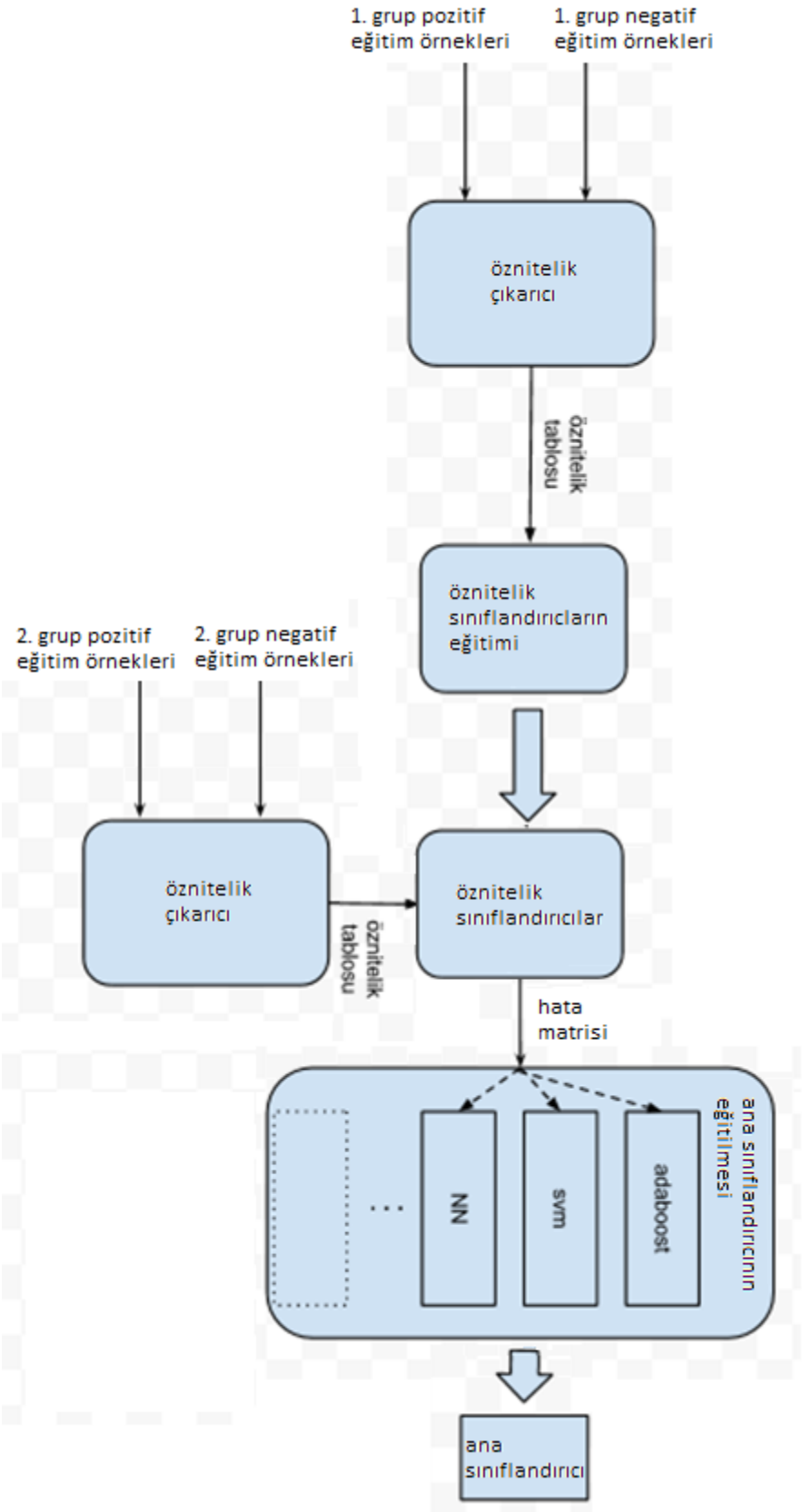
	1. sınıflandırıcı	2. sınıflandırıcı	3. sınıflandırıcı	4. sınıflandırıcı	5. sınıflandırıcı	6. sınıflandırıcı	7. sınıflandırıcı	8. sınıflandırıcı	9. sınıflandırıcı	10. sınıflandırıcı
1. örnek	1	0	0	1	1	0	0	1	1	1
2. örnek	0	-1	0	0	0	-1	0	0	0	0
3. örnek	0	0	-1	0	-1	0	-1	0	0	-1
4. örnek	0	-1	0	-1	0	-1	0	0	-1	0
5. örnek	1	0	0	1	0	0	1	1	1	0
6. örnek	-1	0	0	0	0	-1	0	-1	0	0
7. örnek	0	-1	0	0	-1	0	0	0	0	-1
8. örnek	0	0	1	0	0	0	1	1	0	0
9. örnek	1	1	0	1	0	1	0	1	1	0
10. örnek	0	1	0	1	1	0	0	0	1	1
performans	0.6	0.5	0.8	0.5	0.6	0.6	0.7	0.5	0.5	0.6

Şekil 5. 7 Hata matrisi ve öznitelik sınıflandırıcıların performansı

5.5 Ana Sınıflandırıcı

Ana sınıflandırıcı, öznitelik sınıflandırıcıların aynı örnek için verdikleri sonuçların değerlendirilerek, birbirlerini en iyi tamamlayan sınıflandırıcıların bir araya getirilmesi sonucu oluşan bir sınıflandırıcı topluluğudur. Diğer bir ifadeyle ana sınıflandırıcı öznitelik sınıflandırıcıların sınıflandırma sonuçlarını derleyerek daha doğru bir sınıflandırma yapmaya çalışır.

Yaya tespit çerçevesindeki son aşama ana sınıflandırıcının oluşturulmasıdır. Ana sınıflandırıcı, öznitelik sınıflandırıcılarının bir bileşkesi şeklinde oluşturulur. Öznitelik sınıflandırıcılardan, hata matrisi vasıtasıyla performansı ölçülüp seçilen öznitelik sınıflandırıcıları ana sınıflandırıcıya eklenir (Şekil 5.8). Performansı belirli bir değerden düşük olan öznitelik sınıflandırıcıların ana sınıflandırıcıda kullanılması engellenebilir. Örneğin, sadece performansı %60'dan yüksek olan öznitelik sınıflandırıcılarından bir ana sınıflandırıcı oluşturulması istenebilir.

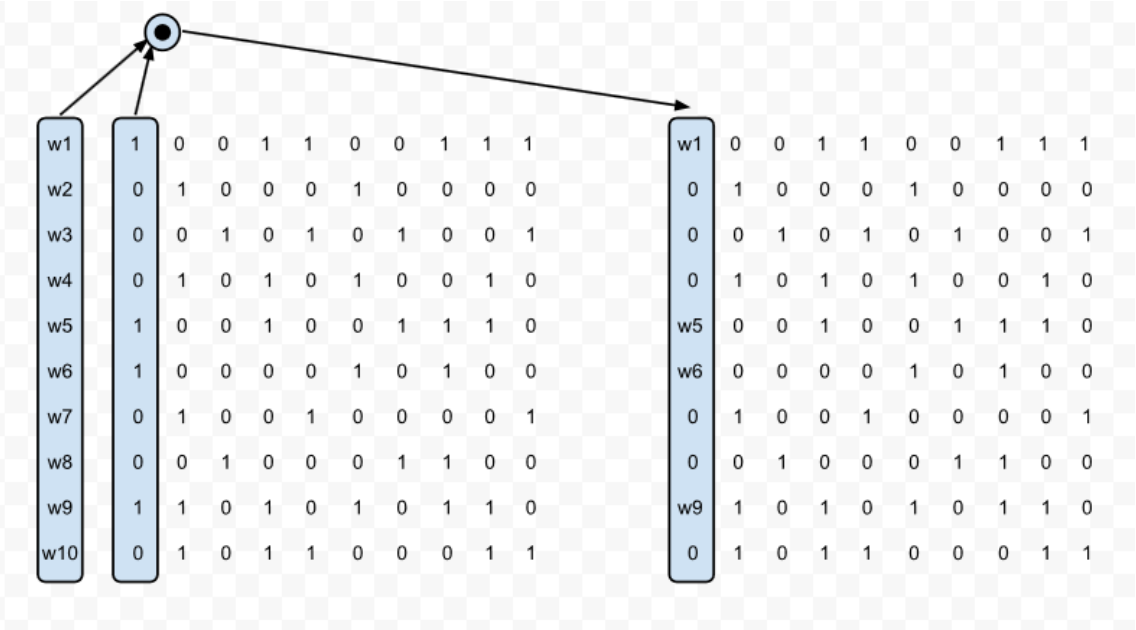


Şekil 5. 8 Ana sınıflandırıcının eğitilmesi

5.5.1 Adaboost Sınıflandırıcısı

Adaboost sınıflandırıcısı özellikle karar vermede kullanılacak parametre sayısı yüksek olduğunda tercih edilen sınıflandırma yöntemlerinden biridir. Bu yöntemle sınıflandırıcı inşa edilirken öznitelik sınıflandırıcılarının ağırlıkları da otomatik olarak belirlenir.

Adaboost sınıflandırıcısının birçok versiyonu bulunmaktadır fakat hepsinin temel çalışma prensibi, eğitme verisi kullanılarak oluşturulan öznitelik sınıflandırıcıların ağırlıklandırılarak birleştirilmesi prensibine dayanır. Buna göre eğitime başlanırken her örneğe eşit ağırlık verilir. Bu ağırlıklarla her bir sınıflandırıcının tüm örnekler için verdiği yanıtlar çarpılır. En az toplam hataya sahip öznitelik sınıflandırıcı seçilir. Seçilen öznitelik sınıflandırıcının hata yaptığı örneklerin ağırlıkları arttırılarak örnek ağırlıkları güncellenir ve bu yeni ağırlıklarla her bir öznitelik sınıflandırıcının yeni performansı bulunur ve en iyi performansı veren sınıflandırıcı seçilerek ana sınıflandırıcıya eklenir. Bu döngüye istenilen başarı oranına ulaşılıncaya kadar ya da hata oranı düştüğü sürece devam edilir. Bu işlem hata matrisi üzerinde şu şekilde yapılır. Her bir örnek için ağırlıkların tutulduğu, örnek sayısı n olmak üzere n x 1 boyutunda bir matris oluşturulur (Şekil 5.9).



Şekil 5. 9 Örnek ağırlıklarıyla sınıflandırıcı sonuçlarının nokta çarpımı alınarak hata miktarlarının belirlenmesi

Daha sonra oluşturulan örnek ağırlık matrisi ile hata matrisinin kolonları tek tek nokta çarpımı ile çarpılır. Bu çarpımlar sonucunda oluşan kolonlardan en az hata miktarına sahip olan kolon seçilir, bir diğer ifadeyle bu kolonu oluşturmak için ağırlık kolonunun çarpılmış olduğu kolon ana sınıflandırıcıya eklenecek öznitelik sınıflandırıcı olarak seçilmiş olur. Her bir kolon için hata miktarı hm(j) (5.3)'deki gibi hesaplanır. Böylece her döngüde yeni bir öznitelik sınıflandırıcı seçilir. Seçilen öznitelik sınıflandırıcı önceki döngülerde de seçilmiş olabilir fakat aynı sınıflandırıcının sürekli seçilmesi beklenen bir durum değildir. Her döngü sonunda seçilen sınıflandırıcıya sınıflandırıcının hatası ile bağlantılı (5.4) ile hesaplanan bir katsayı (b) atanır.

Örneklerin ağırlıkları, o döngüde seçilen en iyi sınıflandırıcının hata miktarıyla ilişkili bir şekilde güncellenir. Denklem (5.5)'te gösterildiği gibi bu güncelleme doğru sınıflandırılan örnek katsayılarının küçültülmesi şeklinde olur. Çünkü $b(t)^{1-x_{ij}}$ ifadesindeki x_{ij} değişkeni hata matrisindeki j'nci sınıflandırıcının i'nci örneğe verdiği yanıtı gösterir. Önceki bölümlerde de anlatıldığı üzere sadece hata durumunda x_{ij} değişkeni 1'e eşit olur. Bu nedenle üslü ifadenin üssü hata durumunda 0'a eşit olur. Sonuçta güncelleme katsayısı 1'e eşit olur ve güncelleme sonucu ağırlıkta değişiklik olmaz.

$$hm(j) = \sum_{i=1}^{n} x_{ij} \cdot w_i \tag{5.3}$$

$$b(t) = hm/(1-hm) \tag{5.4}$$

$$w_i(t+1) = w_i(t) \cdot b(t)^{1-x_{ij}} \tag{5.5}$$

Adaboost sınıflandırıcısı, örnekler için kullandığı ağırlıkları her döngü sonunda Çizelge 5.4'deki gibi günceller. Bu ağırlık güncelleme şekli sınıflandırıcının kullandığı öğrenme kuralı olarak kabul edilir. Adaboost'un standart öğrenme kuralı yerine başka öğrenme kuralları da geliştirilebilir. Çizelge 5.4'te, bu çalışma kapsamında mevcut öğrenme kuralı yerine geliştirilen ve denenen öğrenme kuralları gösterilmiştir.

Çizelge 5. 4 Adaboost Sınıflandırıcısı için gerçeklenen öğrenme kuralları. İlk iki kural bu çalışmada geliştirilmiştir, sonuncu kural ise orijinal Adaboost algoritmasında kullanılmaktadır.

	b(t) : t iterasyonunda seçilen sınıflandırıcının ana sınıflandırıcıdaki ağırlığı	Öğrenme kuralı	Sınıflandırma
1)	b(t)=hm en az hatayı yapan sınıflandırıcının hata miktarına eşittir.	k = 1 + b(t)*x_{ij} w(t+1)=w(t)*k x_{ij}, hata matrisinde en az hatayı veren sınıflandırıcı j için i örneğine verilen yanıt.	$\sum_{t=1}^{T}\frac{u(t)}{b(t)+1} > \frac{1}{2}\sum_{t=1}^{T}\frac{1}{b(t)+1}$ u(t), t. turda seçilen sınıflandırıcının test edilen örneğe verdiği yanıt.
2)	b(t)=1/hm	k = b(t)*x_{ij}, x_{ij} = 1 k = 1+hm(t), x_{ij} = 0 w(t+1)=w(t)*k	$\sum_{t=1}^{T}\frac{u(t)}{b(t)} > \frac{1}{2}\sum_{t=1}^{T}\frac{1}{b(t)}$
3)	b(t)=hm/(1-hm)	$w_i(t+1) = w_i(t)\cdot b(t)^{1-x_{ij}}$	$\sum_{t=1}^{T}u(t)\log\frac{1}{b(t)} > \frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{b(t)}$

Adaboost algoritmasının asıl öğrenme kuralı (Çizelge 5.4'de 3. eğitim kuralı) doğru sınıflandırılan örnek ağırlıklarının küçültülmesini sağlar. Böylece her adımda yanlış sınıflandırılan örneklerin ağırlıkları artmış olur. Önerilen 1 nolu eğitim kuralı ise yanlış sınıflandırılan örneklerin katsayılarının hata miktarlarıyla orantılı olarak artmasını sağlar. 2 nolu eğitim kuralı ise hem doğru hem de yanlış sınıflandırılan örneklerin ağırlıklarını hata miktarlarıyla bağıntılı olarak ama farklı oranlarda arttırılmasını sağlar. Bu öğrenme kurallarının sınıflandırıcının performansı üzerindeki etkisi Çizelge 5.5'de gösterilmiştir.

Eğitim işlemiyle ana sınıflandırıcıda kullanılacak öznitelik sınıflandırıcıları ve bunların ağırlıkları seçilmiş olur. Ana sınıflandırıcı, bir test resmini sınıflandırmak için yalnızca

ana sınıflandırıcıyı oluşturan öznitelik sınıflandırıcılara ihtiyaç duyacağı için test resimleri için tüm öznitelikler değil, sadece gerekli olanlar hesaplanır. Böylelikle test aşamasındaki işlemsel karmaşa azaltılmış olur. Bulunan öznitelik değerleri, ilgili öznitelik sınıflandırıcının ana sınıflandırıcıdaki katkısını gösteren katsayıyla çarpılır ve çarpımlar toplanarak bulunan toplamın bir eşik değerinden büyük olup olmadığına bakılır. Sınıflandırma sonucu bu eşik değer karşılaştırmasının sonuca göre pozitif (yaya var) ya da negatif (yaya yok) olarak elde edilir. Eşik değeri denklem (5.6)'daki gibi hesaplanabilir.

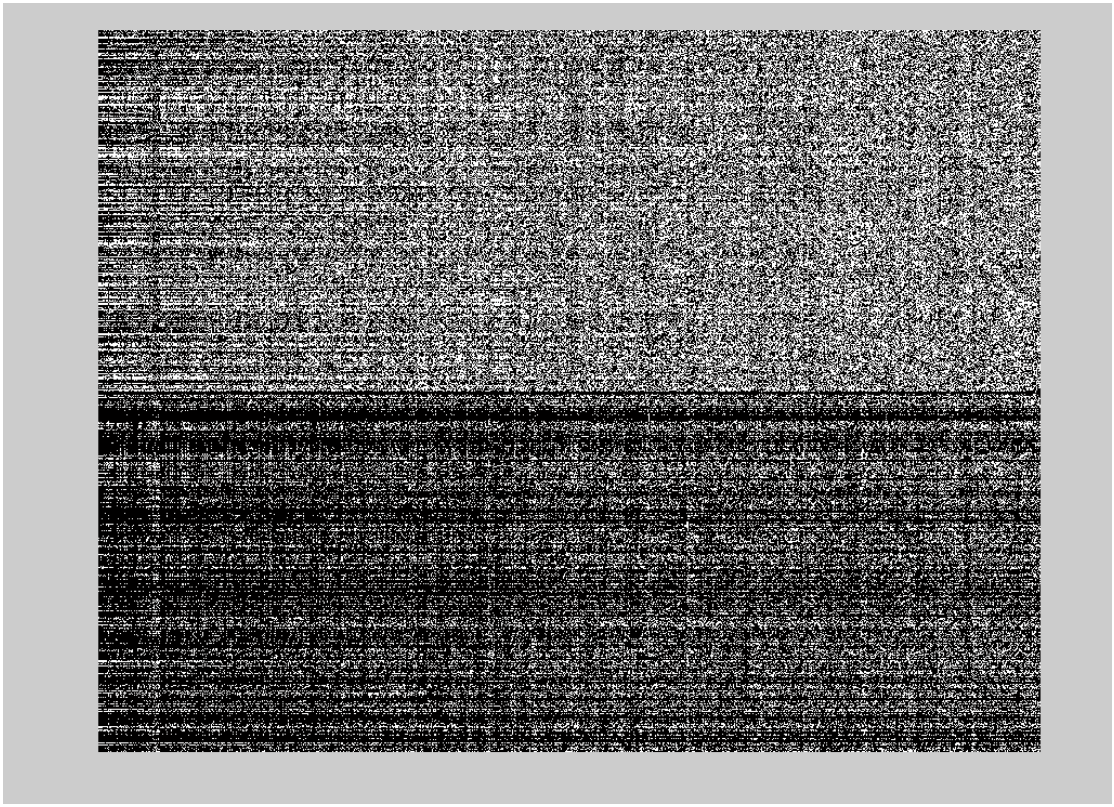
$$th = 0.5\sum_{t=1}^{T}\log\frac{1}{b(t)} \tag{5.6}$$

Çizelge 5. 5 Öğrenme kurallarının ana sınıflandırıcının performansına etkisi

	Kaçırma oranı	Hatalı pozitif oranı
1. eğitim kuralı	0.1909	0.04262
2. eğitim kuralı	0.2593	0.08525
3. eğitim kuralı (Adaboost)	0.2095	0.03907

5.5.2 Ana Sınıflandırıcı Oluşturma Probleminin Cebirsel Çözümü

Ana sınıflandırıcı önceki bölümlerde de bahsedildiği üzere öznitelik sınıflandırıcılarının bir bileşkesidir. Problem, her bir sınıflandırıcının ana sınıflandırıcı içerisindeki ağırlığını ana sınıflandırıcının performansını en yükseğe çıkaracak şekilde seçebilmektir. Şekil 5.10'daki resimde doğru sınıflandırılan örneklerin 0, hatalı sınıflandırılan örneklerin ise 1 ile işaretlendiği bir hata matrisi gösterilmiştir. Matrisi görselleştirebilmek amacı ile matrisin elemanlarının aldığı değerler 0 siyah, 1 ise beyaza denk gelecek şekilde gri tonlarla ifade edilmiştir. Ayrıca öznitelik sınıflandırıcılar en başarılı sınıflandırıcı matrisin en sol sütununa denk gelecek şekilde sıralanmıştır. Amaç bu sütunların ağırlıklı ortalamasını alarak daha başarılı tek bir sütun, yani ana sınıflandırıcıyı oluşturmaktır. Bu bileşke sütun, 0 ve 1 arasındaki değerleri yani gri tonları da içerebilir. Amaç, ana sınıflandırıcının 0'a yakın değere sahip eleman sayısını en fazla yapmaktır.

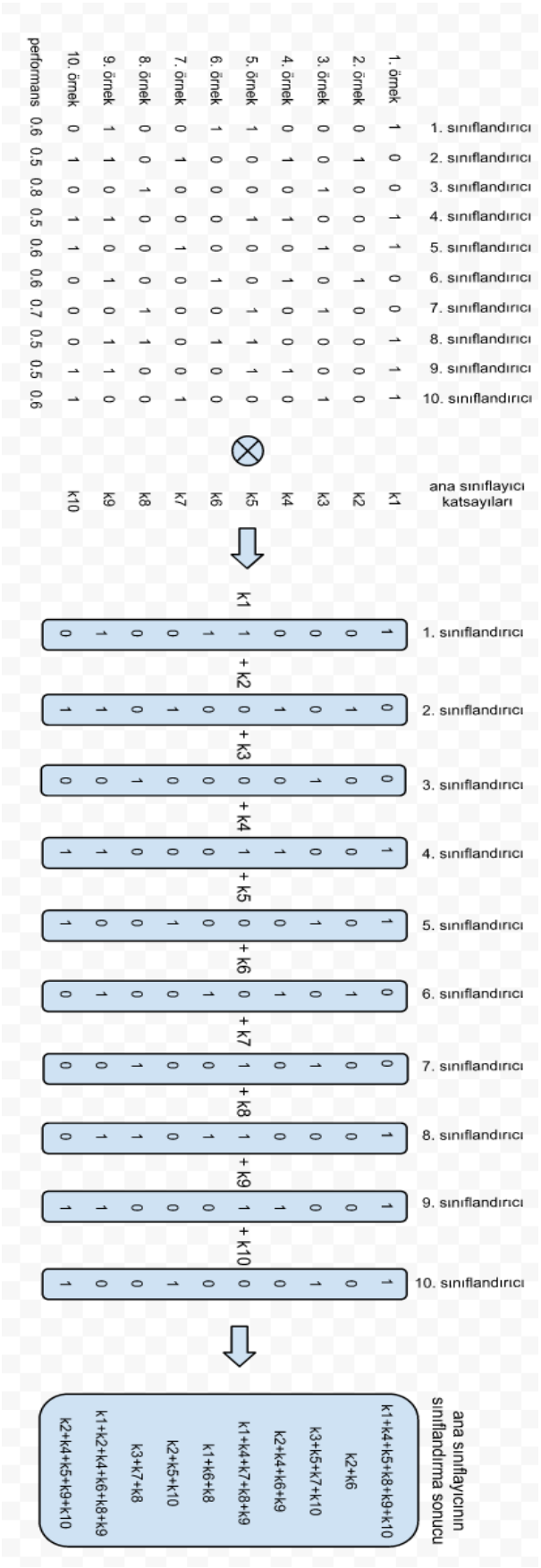


Şekil 5. 10 Örnek bir hata matrisinin resim halinde gösterimi

k sütun vektörü, öznitelik sınıflandırıcıların ana sınıflandırıcıdaki ağırlıklarından oluşsun. Hata matrisi, k vektörü ile çarpıldığında ana sınıflandırıcının sınıflandırma sonucu elde edilir. (Şekil 5.11). Burada istenen, çarpım sonucunda elde edilen vektörün tüm elemanlarının 0 olmasıdır. Bu problem, tüm k'lar sıfır yapılarak çözülebilir fakat bu durum, tüm öznitelik sınıflandırıcıların ana sınıflandırıcıdaki ağırlığının 0 olması, yani ana sınıflandırıcının olmaması anlamına gelmektedir. Dolayısıyla en az bir k sıfırdan farklı olacak şekilde bir çözüm geliştirilmelidir.

Hata matrisi D, ana sınıflandırıcı katsayı kolonu k ve ana sınıflandırıcı sınıflandırma sonucu da A ile gösterilecek olursa, problem denklem (5.7)'deki şekilde formüle edilebilir. D matrisi (örnek sayısı) x (sınıflandırıcı sayısı) boyutunda bir matris, k vektörü ise (sınıflandırıcı sayısı) x (1) boyutunda bir matristir. Dolayısıyla D matrisi ile k vektörünün çarpımı (örnek sayısı) x (1) boyutunda olacaktır.

$$Dk = A \tag{5.7}$$



Şekil 5. 11 Ana sınıflandırıcının oluşturulması

A vektörü, hatasız sınıflama durumunda sadece 0'lardan oluşan bir kolon vektördür. Bu durumda (5.7), (5.8)'deki gibi yeniden yazılabilir.

$$Dk = 0 \tag{5.8}$$

Bu problemin sıfırdan farklı bir çözümü olmayabilir fakat bu durumda bile en az kare toplamı yöntemiyle yaklaşık bir çözüm hesaplanabilir. Tüm verilerin hatasız sınıflandırılabilmesi için tüm örneklerin en azından bir sınıflandırıcı tarafından doğru sınıflandırılmış olması gerekmektedir. Diğer bir deyişle tüm öznitelik sınıflandırıcılar tarafından yanlış sınıflandırılan bir örnek bulunursa sıfırdan farklı bir k çözümü bulunamaz. Yani tam çözüm için hata matrisinde tümüyle birlerden oluşan bir satırın olmaması gerekir.

Denklem (5.7)'nin en az kare toplamı yöntemiyle çözülmesi sonucu bulunan k katsayılarından oluşan bir ana sınıflandırıcı tüm eğitim örneklerini yani hata matrisinde kullanılan tüm örnekleri başarılı bir şekilde sınıflamaktadır. Çünkü bu şekilde eğitim kümesi için mükemmel çözüm veren bir sınıflandırıcı oluşturulmaktadır. Fakat yapılan denemelerde bu durumda test kümesinde oldukça kararsız ve rastgeleden biraz daha iyi bir sınıflandırma yapıldığı görülmüştür. Bunun 2 nedeni bulunmaktadır; ilk neden, bulunan çözümün genel bir çözüm değil, eğitim kümesi için özelleşmiş bir çözüm olmasıdır. Başka bir deyişle, sistem problemi öğrenmek yerine eğitme verisini ezberlemiştir. İkinci neden ise, k vektöründeki bazı katsayıların eksi değerler almasıdır. Ancak bu durum, ana sınıflandırıcıya eksi yönde katkı yapan zayıf sınıflandırıcıların varlığı anlamına gelir ki, bu olmaması gereken bir durumdur. Bu nedenle, k_i'lerin 0 ile 1 arasında değerler alması gerekir.

5.5.3 Örneklerin Ağırlıklandırılarak Çözümün Bulunması

Ana sınıflandırıcıdaki k katsayılarının bulunabilmesi için Adaboost yönteminde olduğu gibi örneklerin ağırlıklandırılması yöntemi tercih edilebilir. Buna göre ilk başta en az hatayı veren sınıflandırıcı ana sınıflandırıcı olarak seçilir. Bu sınıflandırıcının hata yaptığı örneklerin ağırlıkları belirli bir oranda arttırılarak bu hatalı örnekleri başarıyla seçen bir sınıflandırıcı bulunmaya çalışır ve bu sınıflandırıcı ilk seçilen sınıflandırıcıya performansıyla orantılı bir şekilde eklenir. Böylelikle ana sınıflandırıcının bir önceki

adımda yapılan aynı hataları yapmaması beklenir, fakat sınıflandırıcının değişmesiyle birlikte yeni hatalar ortaya çıkabilir. Amaç her adımda ana sınıflandırıcıya yeni bir öznitelik sınıflandırıcı ekleyerek hata oranını düşürmektir.

Her bir öznitelik sınıflandırıcının ana sınıflandırıcıdaki ağırlığı k olarak ifade edilirse, hata matrisi k vektörü ile çarpıldığında ana sınıflandırıcının potansiyel sınıflandırma sonucu elde edilir. Elde edilen sonucun tüm elemanları 0'a olabildiğince yakın bir vektör olması amaçlanmaktadır. Bu katsayıların bulunabilmesi için öncelikle en az hatayı veren sınıflandırıcı ana sınıflandırıcı olarak seçilir. Bu sınıflandırıcının hata yaptığı örneklerin ağırlıkları belirli bir oranda arttırılarak hatalı sınıflandırılan örnekleri başarıyla sınıflandıran bir öznitelik sınıflandırıcı bulunmaya çalışılır. Bulunan yeni öznitelik sınıflandırıcı, ana sınıflandırıcıya hata miktarıyla ters orantılı olarak eklenir ve ana sınıflandırıcının bir önceki adıma göre daha yüksek bir başarıma ulaşması beklenir.

Amaç her adımda ana sınıflandırıcıya yeni bir zayıf sınıflandırıcı ekleyerek hata oranını düşürmektir. Bu yöntem bir çeşit arttırma ("boosting") algoritması [50] ya da diğer bir ifadeyle zayıf sınıflandırıcıların yanlış yaptıkları örnekleri kullanarak daha güçlü bir sınıflandırıcı elde etme algoritması sayılabilir. Ana sınıflandırıcıdaki hata sayısının, yani 0.5'den büyük değerlerin sayısının her iterasyon sonrasında düştüğü Şekil 5.12'de görülmektedir.

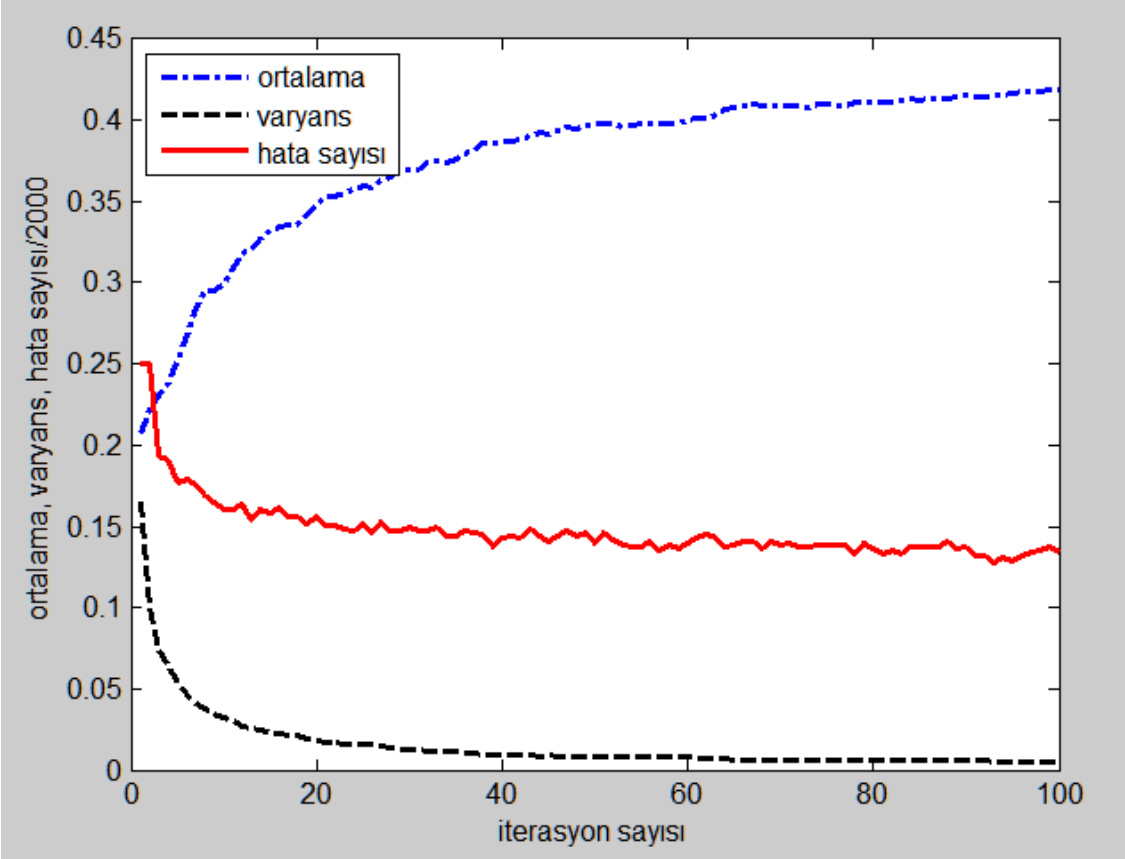
Ana sınıflandırıcının sınıflama sonucunu gösteren vektöre A vektörü denilirse, A'nın değeri 0.5'ten büyük elemanlarının sayısı zamanla azalırken vektörün ortalama değeri artar ve varyansı düşer. A vektörünün ortalama değeri 0.5'e ulaştığında eğitim sonlandırılır, çünkü bu ortalama olarak her örnek için kararsız bir duruma ulaşıldığını gösterir.

Algoritmanın t. iterasyonunda, j. zayıf sınıflandırıcının hata miktarı $\varepsilon(j)$ sembolü eşitlik (5.9)'daki gibi hesaplanır:

$$\varepsilon(j) = \sum_{i=1}^{n} d_{ij} w_i \qquad (5.9)$$

Denklem (5.9)'da, n örnek sayısı, d_{ij} D'nin i. satır, j. sütundaki elemanı, w_i ise i. giriş verisinin tüm sınıflandırıcılar tarafından ortalama ne kadar hatalı sınıflandırıldığını

gösteren bir ağırlık değeridir. Algoritmanın ilk adımında tüm ağırlıklar (w değerleri) 1/n olarak belirlenir. Denklem (5.9)'daki gibi hesaplanan hata miktarları kullanılarak en az hataya sahip zayıf sınıflandırıcı seçilir. Seçilen öznitelik sınıflandırıcı hata miktarıyla ters orantılı olarak ağırlıklandırılmış bir şekilde ana sınıflandırıcıya eklenir. Seçilen öznitelik sınıflandırıcının ana sınıflandırıcı içindeki ağırlığı k ile gösterilir ve (5.10) ile hesaplanır:



Şekil 5. 12 Ortalama, varyans ve hata sayısının iterasyona göre değişimi

$$k_t = \frac{1}{min(\varepsilon)} \tag{5.10}$$

Ana sınıflandırıcının çıkışı yeni eklenen öznitelik sınıflandırıcı da hesaba katılarak yeniden hesaplanır:

$$A_{t+1,i} = \frac{A_{t,i} \sum_{j=1}^{t-1} k_j + k_t d_t}{\sum_{j=1}^{t} k_j} \tag{5.11}$$

Çıkış vektörü A'da değeri T = 0.5'ten büyük elemanlar için w'lar (5.12) ile güncellenir:

$$w_{t+1,i} = w_{t,i} + \frac{1}{n} \tag{5.12}$$

Son olarak, w vektörü normalize edilir:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \tag{5.13}$$

Eğer daha önceden belirlenmiş olan bir iterasyon sayısına ulaşıldıysa ya da herhangi bir anda, E[.] beklenen değer operatörü olmak üzere, E[A] > T olursa algoritma durur, değilse (5.9)'dan itibaren yineleme sürer. Eğitme aşaması tamamlandığında, ana sınıflandırıcıya uygulanan herhangi bir x girişine ilişkin karar (5.14)'deki gibi hesaplanır:

$$y = \begin{cases} 1, & \sum_{l=1}^{L} k_l h_l(x) > \frac{1}{2} \sum_{l=1}^{L} k_l \\ 0, & diğer \end{cases} \tag{5.14}$$

Yapılan ilk denemede önerilen algoritma ile yaya tespit sistemlerinde sıkça kullanılan Adaboost algoritması karşılaştırılmıştır. Elde edilen sonuçlar Çizelge 5.6'da verilmiştir. Sadece dikdörtgen farkları kullanılarak ve 100 iterasyon için elde edilen sonuçlardan, önerilen yöntemin başarısının Adaboost'a yakın olduğu görülmektedir. Ancak, önerilen yöntem, hatalı pozitif veya hatalı negatif örneklere daha fazla önem verecek şekilde eğitilebilmektedir. Bu şekilde bu hata türlerinden birinin diğerinden daha kritik olduğu durumlarda, istenilen hata türünü azaltacak şekilde eğitilebilen bir arttırma ("boosting") algoritması (Çizelge 5.7) elde edilmiştir.

Çizelge 5. 6 Önerilen algoritma ve Adaboost'un başarımları

	Kaçırma oranı	Hatalı pozitif oranı
Adaboost	0.2095	0.03907
Önerilen yöntem (T=0.5)	0.2095	0.04706
Önerilen yöntem (T=0.45)	0.2015	0.03996

Çizelge 5. 7 Önerilen algoritmanın eğitim adımları

- Hata matrisi $x_{11},\ldots,x_{mn}$ olarak düzenlenir. Burada x_{ij}, hatalı sınıflandırmalar için 0, başarılı sınıflandırmalar için 1 olarak kabul edilir.
- m ve n sırasıyla örnek sayısı ve sınıflandırıcı sayısını gösterir. Ağırlıklar $w_{1,i}=1$ olacak şekilde ilklenir.
- Ağırlıklar güncellenirken kullanılacak adım miktarı $\Delta w = 1/m$ olarak ayarlanır.

- T iterasyon sayısı olmak üzere, her t=1,.,T için:

1. Ağırlıklar normalize edilir.

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$$

2. Her bir j sınıflandırıcısı için, tüm örneklere verilen sonuçlar x_{ij} ve w_t ağırlığına göre hata miktarı ölçülür.

$$\varepsilon_j = \sum_i w_{t,i} x_i$$

3. En az ε_t hatasına sahip d_t sınıflandırıcısı seçilir. d_t, hata matrisinde en az ε_t hatasına sahip kolonu gösterir.

4. Seçilen zayıf sınıflandırıcının ana sınıflandırıcıdaki ağırlığı hesaplanır.

$$k_t = \frac{1}{\min(\varepsilon_j)}$$

5. Ana sınıflandırıcının t. iterasyon sonunda her bir örneğe verdiği yanıt $A_{t,i}$ olmak üzere, son iterasyonda seçilen d_t zayıf sınıflandırıcısı, k_j ağırlığıyla toplama eklenir.

$$A_{t+1,i} = \left(A_{t,i} \sum_{j=1}^{t-1} k_j \right) + k_t d_t$$

6. Ağırlıklar güncellenir:

$$w_{t+1,i} = w_{t,i} + (A_{t+1,i} > 0.5)\Delta w$$

7. Son iterasyon sayısı T'ye ulaşıldığında ya da aşağıdaki koşul sağlandığında eğitim sonlandırılır.

$$\frac{A_{t+1,i}}{m} > 0.5$$

BÖLÜM 6

ENTROPİK ÖZNİTELİKLER, GÜVENİLİRLİK SKORU ve DOĞRUSAL OLMAYAN ÖZNİTELİK SINIFLANDIRICILAR

Bu bölümde 5. bölümde anlatılan yaya tanıma sistemleri üzerinde uygulanabilecek iyileştirmeler üzerinde durulmuştur. Öncelikle resmin entropisini hesaplamaya dayalı yeni bir öznitelik çıkarım yöntemi geliştirilmiş ve bulunan öznitelikler doğrusal olmayan öznitelik sınıflandırıcılarla sınıflandırılmış ve ana sınıflandırıcının, öznitelik sınıflandırıcıların sınıflandırmalarındaki güvenilirlik skorlarını kullanması sağlanmıştır.

6.1 Yeni Bir Öznitelik Çıkarım Yöntemi

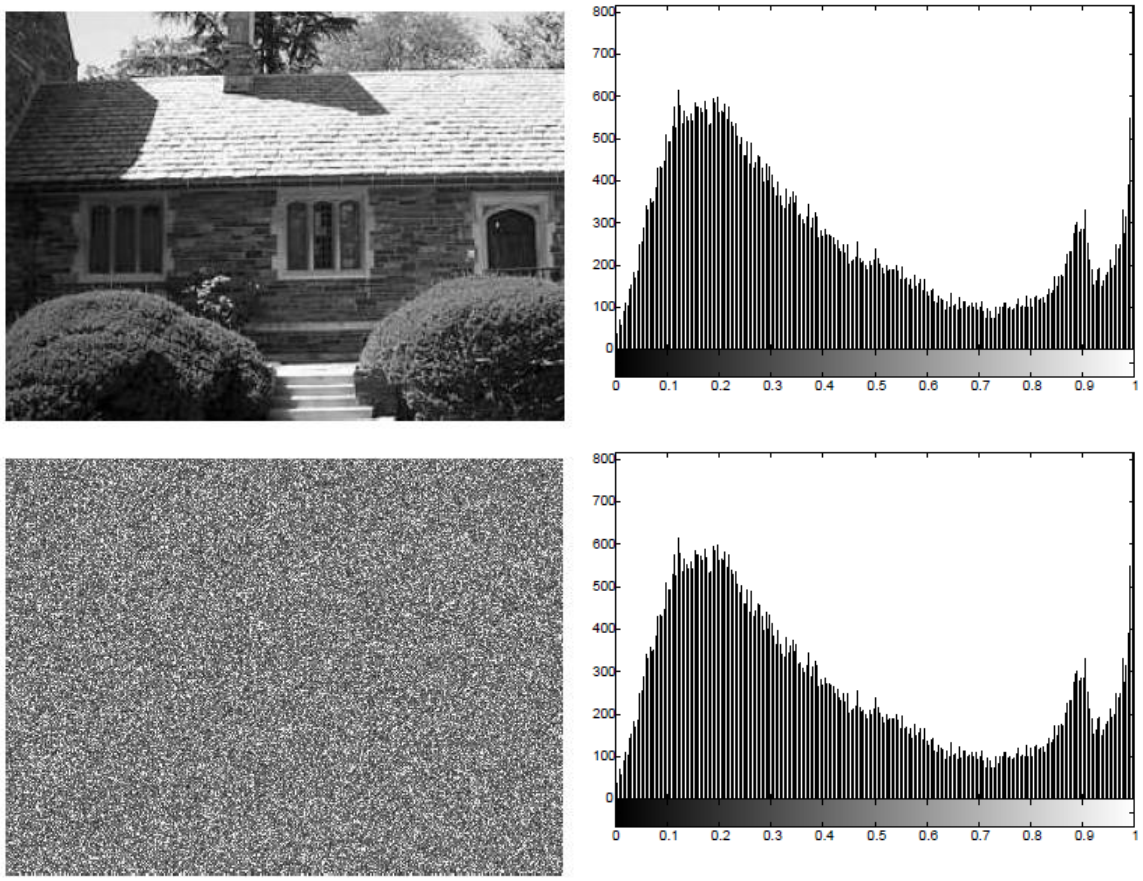
5. Bölümde yaya tespit çerçevesi farklı öznitelik çıkarım yöntemlerinin birarada ya da tek başlarına kullanılmasına imkan tanımaktadır. Kullanılan öznitelikler yaya tespit sisteminin başarısını doğrudan etkilemektedir. Öznitelikler, ne kadar ayırt edici ve birbirini tamamlayıcı olursa sistemin başarısı o ölçüde artmaktadır. Diğer bir deyişle kullanılan öznitelik sayısından daha önemli olan bu özniteliklerin ne kadar farklı bilgi taşıdığıdır, aynı bilgiyi taşıyan özniteliklerin çok sayıda kullanılması sistemin başarısını olumsuz yönde etkilemektedir [53].

Resimlerin içerdikleri bilgi miktarlarının Shannon'ın [54] önerdiği şekilde ölçülmesi bir takım zorluklar taşımaktadır. Bir resmin kesin olarak ne kadar bilgi içerdiğini ölçebilen bir yöntem henüz bulunabilmiş değildir. Bir resmin entropisi genellikle aşağıdaki yöntem kullanılarak bulunmaktadır.

$$E = -\sum_{i=0}^{255} p(l_i) log_2 p(l_i) \quad (6.1)$$

(6.1)'de verilen denklem, 256 tonlu gri bir resmin entropisini hesaplamaktadır. Bu formülde l_i gri ton seviyesini, $p(l_i)$ ise bu tonun resimde bulunma olasığını göstermektedir. Resim içerisinde belirli bir gri tondaki piksel sayısının resimdeki tüm piksel sayısına oranı, o gri tonun resimde bulunma olasılığını gösterir. Böylece gri tonlar için bir olasılık dağılımı bulunur ve bu dağılım kullanılarak resmin entropisi ölçülür.

Her bir gri tonun resimde kaç kez geçtiğini bulmak, o resmin histogramını çıkarmaya eşdeğerdir. O nedenle bu yönteme histogram entropisi de denilir. Ancak, entropisi aynı olan iki resmin birbirine eş olduğunu söylemek doğru değildir. Sabuncu [55], bu durumu Şekil 6.1'de göstermiştir.



Şekil 6. 1 Ev resminin histogramı korunarak pikselleri karıştırılarak ilk resimden tamamen farklı fakat aynı entropiye sahip ikinci resim elde edilmiştir [55]

Şekil 6.1'de öncelikle bir ev resminin entropisi bu yönteme göre hesaplanmış, daha sonra aynı resmin pikselleri karıştırılarak ikinci resim elde edilmiş ve entropisi hesaplanmıştır. Bu iki resmin entropileri aynı olduğu halde görünümleri birbirlerinden oldukça farklıdır. Diğer bir ifadeyle, iki resmin entropilerinin aynı çıkması halinde bu resimlerin aynı olduğu yargısına varılması doğru değildir, fakat entropilerinin farklı çıkması durumunda bu resimlerin farklı olduğu sonucu çıkarılabilir.

Entropi bilgisi kullanarak resim eşleştirme, kaydetme ya da resimler arasındaki farklılığı ölçme [55], [56], [57] gibi konularda çalışmalar yapılmış olmasına karşın bu bilginin nesne tespitinde kullanılması konusuna yukarıda ifade edilen durumdan ötürü yeterince ilgi gösterilmemiştir. Bununla birlikte bir resmin entropisini ölçebilmek için farklı yöntemler de önerilmiştir. Bu yöntemlerden biri Cassini Satürn görevinde kullanılan [58] komşu pikseller arasındaki farkın frekansını kullanarak Shannon entropisini hesaplama yöntemidir. Bunun dışında ayrıca resmin histogramı yerine tahmini bir olasılık dağılım fonksiyonu üzerinden entropi hesaplamayı öngören tak çalıştır (plug-in) [59] yöntemi ve örnek kümeleri üzerinden grafikler hesaplamaya dayanan entropik grafikler [60] yöntemleri de bulunmaktadır.

Önerilen öznitelik çıkarma yöntemi Şekil 5.4'deki şablonları kullanmaktadır. Bu şablonların alt bloklarındaki piksel toplamları yerine her alt bloğun entropisi hesaplanmaktadır. Dikdörtgen farkları yöntemindeki gibi koyu alt blokların entropileri açık blokların entropilerinden çıkarılarak her şablon için bir öznitelik değeri hesaplanır.

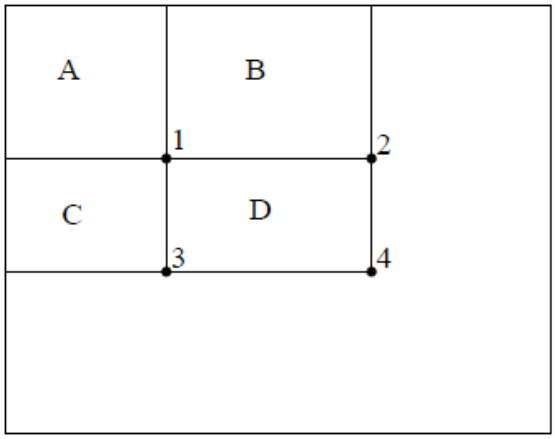
Bir alt bloğun entropisi (6.1)'de verilen denklem ile hesaplanır. Entropi hesaplanmadan önce o alt bloktaki gri tonların olasılıkları hesaplanmalıdır. Dolayısıyla resmi tararken her bir alt bloğun entropisini bulmak yoğun hesaplama gerektiren bir işlemdir. Gereken hesaplama miktarını azaltabilmek için çetele resimleri olarak adlandırdığımız yardımcı resimler kullanılmasını önermekteyiz.

Yardımcı resimlerin ait oldukları asıl resim kaç gri ton içeriyorsa o kadar çetele resmi kullanılması gerekir. Dolayısıyla 256 tonlu yani 8 bitlik bir gri resim için 256 adet çetele resmi kullanılmalıdır. Bir çetele resminin (x,y) koordinatında, asıl resmin bu koordinatının sol üst tarafında ilgili gri tondan toplamda kaç adet piksel bulunduğu bilgisi saklıdır (Şekil 6.2). Çetele resimleri (6.2)'deki gibi hesaplanır, bu denklemde ci_l

ifadesi l tonu için oluşturulan çetele resmini göstermektedir. Çetele resimlerinde, integral resimlerinden [16] farklı olarak piksel değerlerinin toplamı tutulmaz, bunun yerine bir gri-tondaki toplam piksel adedi tutulur.

$$ci_l(x,y) = \sum_{x`<x,y`<y} \begin{cases} 1 & i(x`,y`) = l \\ 0 & aksi\ halde \end{cases} \tag{6.2}$$

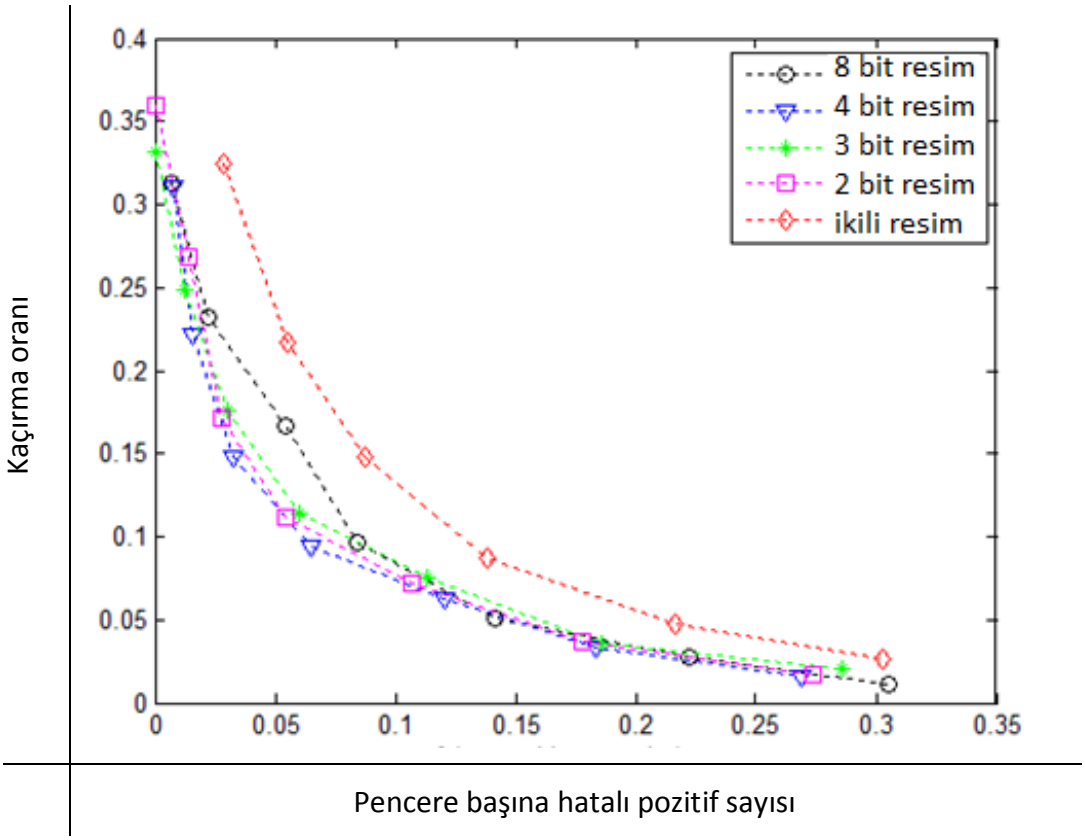
l gri tonundaki piksel adedinin bir alt-bloktaki toplam piksel adedine bölünmesi o alt blok için $p(l)$ değerini yani l gri tonunun olasılığını verir. Böylece, herhangi bir alt-bloğun entropisi çetele resimleri kullanılarak gri tonların olasılıklarının hesaplanmasıyla bulunur.



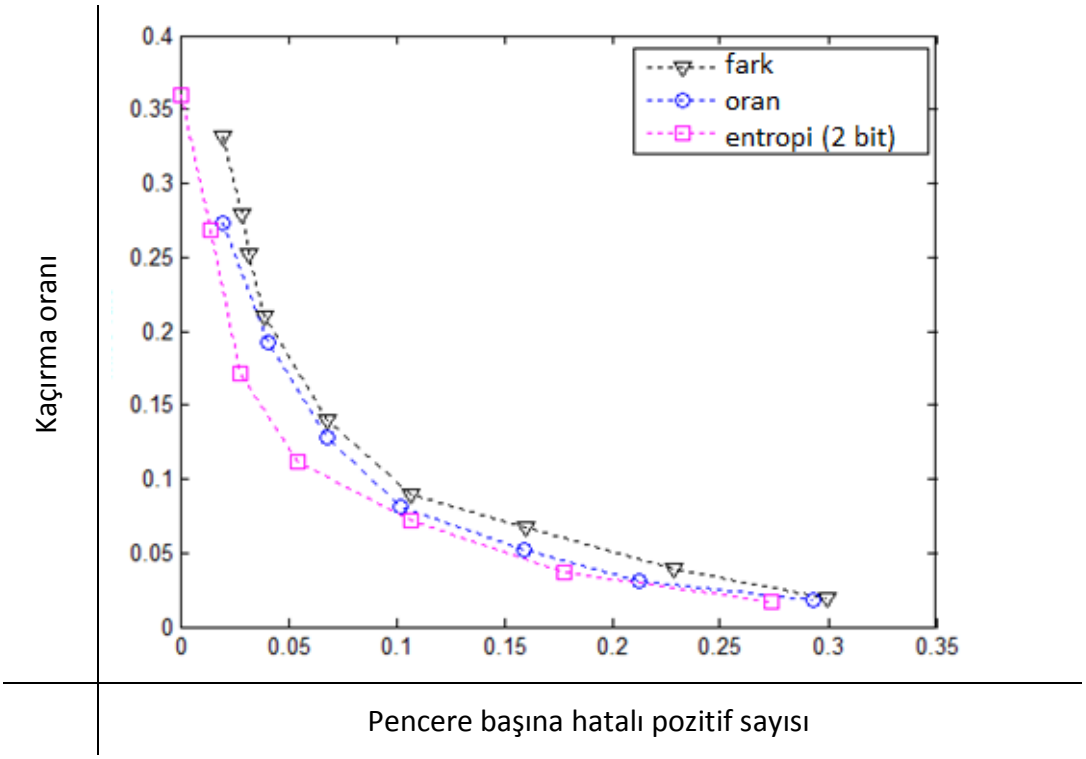
Şekil 6. 2 l gri tonu için çetele resmi, D bölgesinde kaç adet l tonunda piksel olduğu 4-2-3+1 şeklinde hesaplanabilir.

Çetele resimlerinin kullanımı, entropinin bulunması için gereken hesaplama miktarını oldukça düşürür, ancak çok miktarda bellek kullanımı gerektirir. Daha önce de bahsedildiği gibi 256 gri tonlu bir resim için 256 adet çetele resmini bellekte tutmak gerekir. Bu nedenle daha az derinlikte yani daha az bit kullanan gri resimlerin kullanılması gerekli çetele resmi adedini düşürecektir. Yapılan denemeler sonucunda entropilerin 2-bitlik gri tonlu resimler üzerinden hesaplanmasının yeterince iyi sonuçlar verdiği görülmüştür (Şekil 6.3). Bu şekilde, öznitelik hesaplanırken çok sayıda entropi bulmak için gerekli hesaplama sayısı ve bellek miktarı tespit başarımından ödün verilmeden azaltılmıştır.

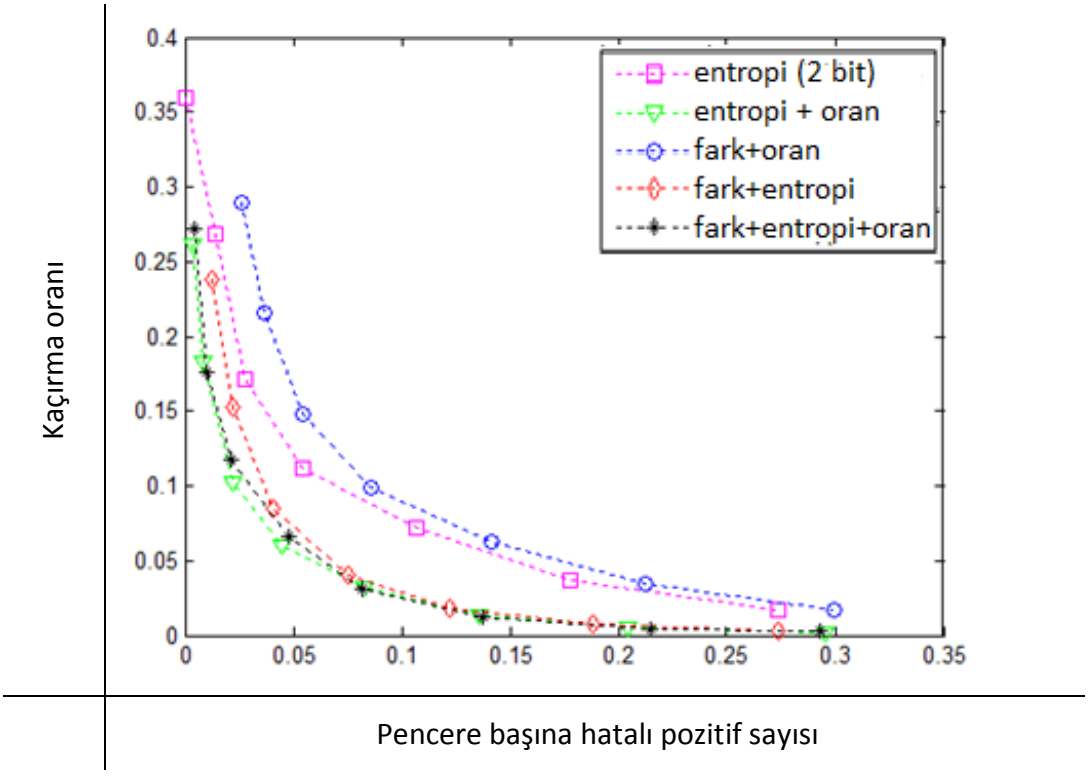
Şekil 6.4'daki grafikten entropik özniteliklerin diğerlerinden az bir farkla daha başarılı tespitler yaptığı görülmektedir. Fakat daha önce bahsedildiği gibi entropi, resimleri eşleştirmekten ziyade farklı resimleri bulmakta başarılıdır. Yani dikdörtgen oranları/farkları yöntemlerinden farklı bir karakteristikte sınıflandırma yapmaktadır. Bu nedenle entropik öznitelik bulma yöntemi diğer öznitelik bulma yöntemleriyle birlikte kullanıldığında oldukça yüksek başarı elde edilmektedir. Şekil 6.5'deki grafikte öznitelik bulma yöntemlerinin çiftler halinde birleştirilmesi ve üçünün de bir arada kullanılması sonucu elde edilen performanslar karşılaştırılmaktadır.



Şekil 6. 3 Farklı derinliklerde gri tonlu resimler üzerinden hesaplanan entropik öznitelikleri kullanan tespit sistemlerinin başarılarının karşılaştırılması.

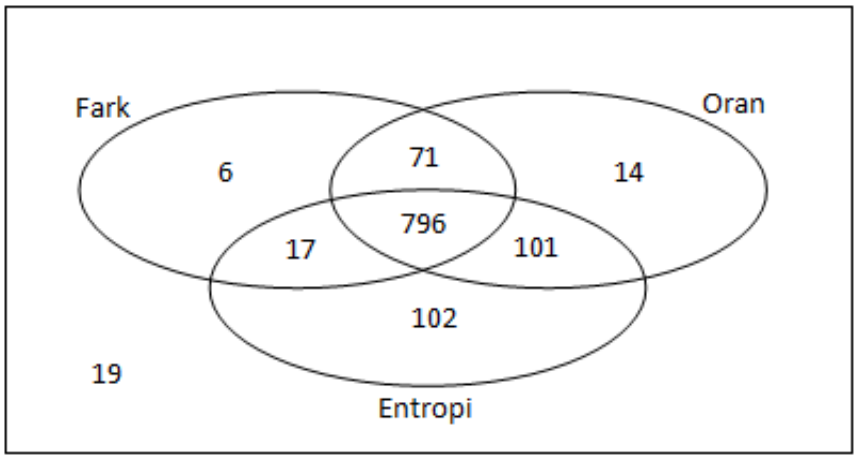


Şekil 6. 4 Entropik öznitelik bulma yönteminin diğer yöntemlerle karşılaştırılması



Şekil 6. 5 Öznitelik bulma yöntemlerinin birlikte kullanılması halinde elde edilen sonuçların karşılaştırılması

Şekil 6.5'de entropik öznitelik yönteminin dikdörtgen farkları ya da oranları yöntemiyle birlikte kullanıldığında performansın oldukça arttığı görülmektedir. Fakat dikdörtgen oranları ve dikdörtgen farkları yöntemleri birlikte kullanıldığında performans neredeyse hiç artmamaktadır. Bu dikdörtgen farkları ve dikdörtgen oranları yöntemlerinin genellikle aynı örneklerde başarılı/başarısız olmalarından kaynaklanmaktadır. Fakat entropik öznitelikler diğer özniteliklerin başarısız olduğu örnekleri daha başarılı şekilde temsil etmektedir. Bu nedenle de diğer öznitelik çıkarma yöntemlerini çok iyi tamamlamaktadır. Ayrıca 3 öznitelik yönteminin birlikte kullanılması halinde, performansın entropik özniteliklerin dahil olduğu ikili kombinasyonlara nazaran pek artmadığı gözlenmektedir. Bunun nedeni dikdörtgen oranları ve farkları yöntemlerinin birbirini kötü bir şekilde tamamlaması ve dolayısıyla birlikte kullanılmalarının pek anlamlı olmayışıdır. Şekil 6.6'da öznitelik bulma yöntemlerinin birbirleriyle ilişkisi gösterilmiştir.

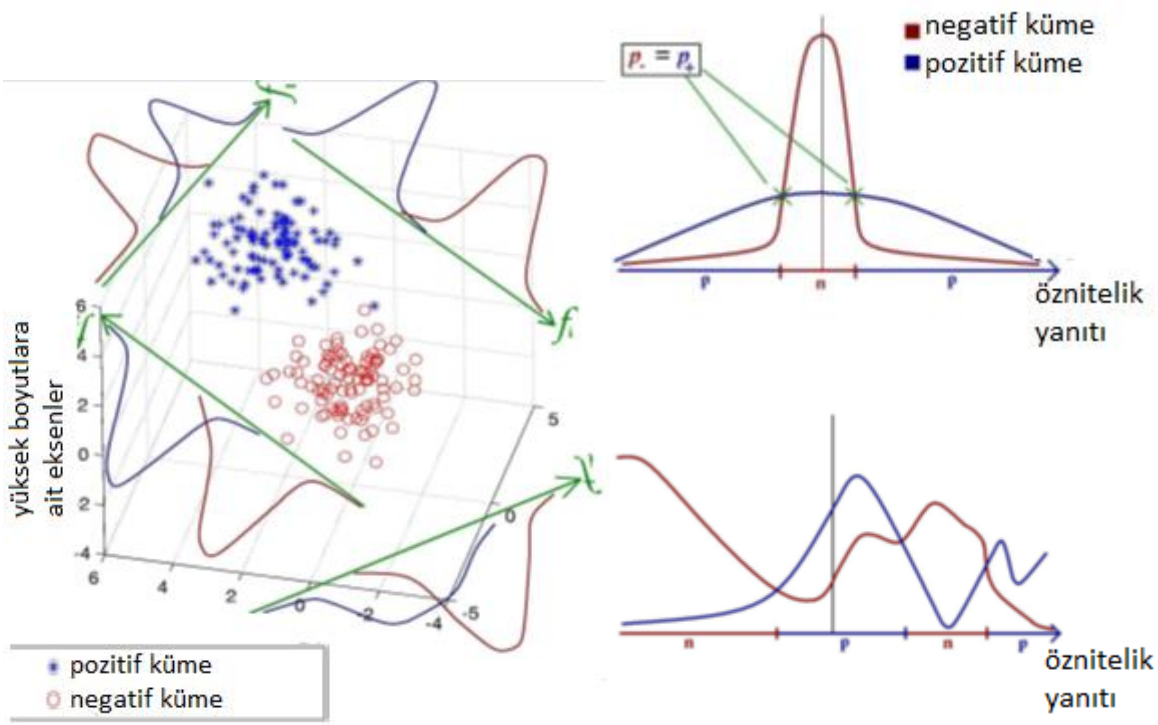


Şekil 6. 6 Öznitelik bulma yöntemlerinin 1126 pozitif test örneğini tespit etme ilişkileri.

Şekil 6.6'daki Venn diyagramı, 3 yaya tespit sisteminin 1126 pozitif test örneğine verdiği yanıtlarla oluşturulmuştur. Bu yaya tespit sistemlerinden her biri 690 adet aynı türde (fark, oran ya da entropi) özniteliği kullanan Adaboost sınıflandırıcının eğitilmesiyle oluşturulmuştur. Bu diyagrama göre 19 pozitif örneği 3 öznitelik bulma yöntemi de tespit edemezken, 796 yayayı her üç yöntem de tespit edebilmektedir. Venn diyagramından ikili birleşimlere bakıldığında en az tespitin fark ve oranın birleşimi halinde gerçekleştiği görülmektedir.

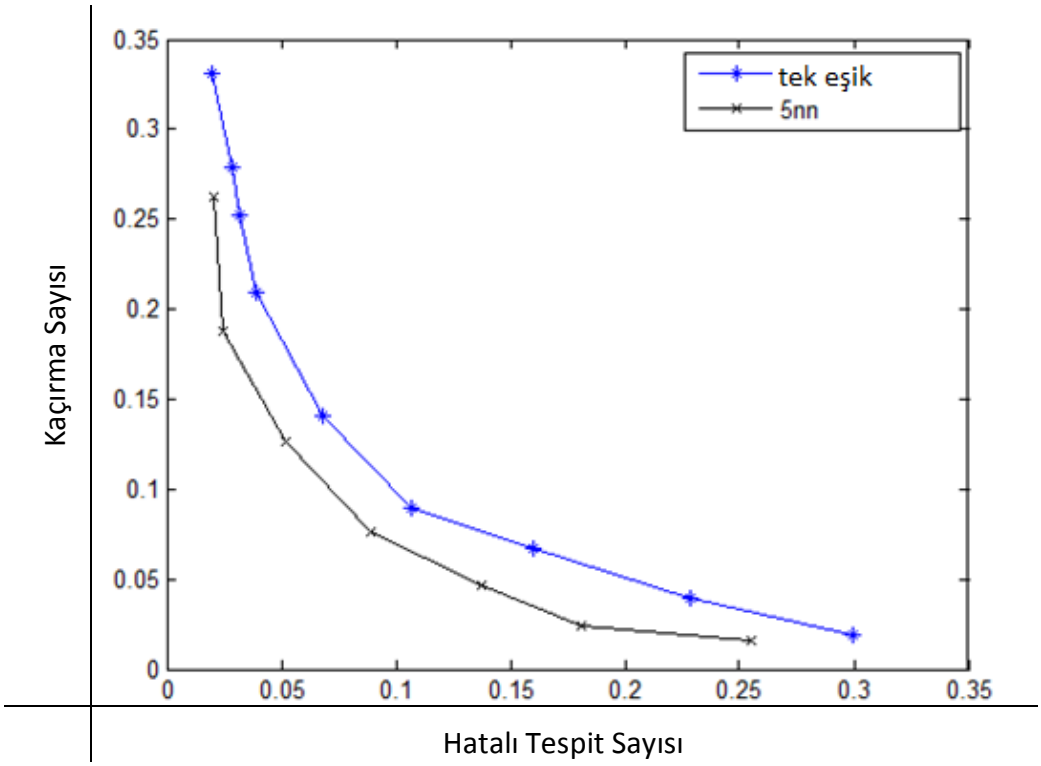
6.2 Doğrusal Olmayan Sınıflandırıcıların Etkisi

Bir öznitelik için eğitim örneklerinden alınan değerler genellikle doğrusal olarak ayrışmazlar. Bu nedenle tek bir eşik değeri kullanılarak elde edilen sınıflandırıcıların başarı oranları genellikle düşüktür. Rasolzadeh ve arkadaşları [62] yaptıkları çalışmalar sonucunda bu durumu açık bir şekilde ifade etmişlerdir (Şekil 6.7). Rasolzadeh ve arkadaşları bu problemin çözümü için birden fazla eşik değer kullanımını önermiş ve bu eşik değerlerin nasıl hesaplanabileceğini göstermişlerdir. Önceki bölümde anlatılan çalışmalarda her bir eğitim örneği için 690 adet öznitelik bulunmuş ve her bir özniteliğin 1208 eğitim örneği için aldığı değerler kullanılarak eşik değer sınıflandırıcıları oluşturulmuştu. Viola ve Jones'un [16]'da önerdiği gibi, bir özniteliğin bu 1208 örnek için aldığı değerleri en iyi ayıran eşik değer o öznitelik için sınıflama kriteri olarak seçilmişti. Özetle, her bir öznitelik için o özniteliği kullanan bir sınıflandırıcı oluşturulmuş, böylece 690 öznitelik için 690 adet sınıflandırıcı oluşturulmuştu.



Şekil 6. 7 Yaya olan ve olmayan örnekler, insan zihnindeki çok boyutlu bir uzayda soldaki resimdeki gibi iki grup oluşturur. Yeşil doğrular öznitelikleri temsil etmektedir. Sağda ise rastgele seçilmiş özniteliklere ait dağılımlar görülmektedir [62]

Bu bölümde yapılan çalışmalarda ise yukarıda bahsedilen sorunun aşılması için en yakın komşu sınıflandırıcısı kullanılmıştır. En yakın komşu sınıflandırıcısı (kNN:kth Neares Neighbour ve k=5 olmak üzere), eşik değer sınıflandırıcısıyla karşılaştırıldığında elde edilen sonuçlar Şekil 6.8'deki grafikte gösterilmiştir. En yakın komşu sınıflandırıcısı doğrusal olmayan sınıflandırmaları da yapabilmesinin yanısıra sınıflandırma sonucuyla birlikte bu sonucun güvenilirliğini de verebilmektedir. Örneğin, k=5 olarak seçildiğinde en yakın 5 komşudan 5'nin de aynı sınıfta olması durumunda sınıflandırma sonucunun güvenilirliği %100 ya da "1" olarak atanır.



Şekil 6. 8 Tek eşik değer sınıflandırıcısı ile 5nn sınıflandırıcılarının performanslarının karşılaştırılması.

6.3 Güvenilirlik Skoru Kullanan Adaboost

Tek bir sınıflandırıcının yaptığı sınıflamaya güvenmek yerine çok sayıda sınıflandırıcının verdiği sonuçları kullanarak daha iyi bir sınıflama yapmaya çalışmak 1990'lardan beri [63], [64] üzerinde çalışılan bir konudur. Bu konuda bilinen 2 yöntem bulunmaktadır; paketleme [65] ("bagging: bootstrap aggregating") ve yükseltme ("boosting") [41]. İlk olarak geliştirilen yöntem paketleme yöntemidir. Bu yönteme göre eğitim örnekleri aynı boyuttaki paketlere ayrılır. Her pakete eğitim kümesinden rastgele örnekler aynı

örneğin tekrar atanmasına izin verecek şekilde atanır. Her eğitim paketi için farklı bir sınıflandırıcı oluşturulur. Sınıflama, bu sınıflandırıcıların oy çokluğuna bakılarak yapılır. Yani bu şekilde geliştirilen sınıflandırıcıların çoğu, bir test örneğinin hangi sınıfta olduğunu söylüyorsa, örneğin o sınıfta olduğuna karar verilir.

Boosting yöntemi [41] paketleme yönteminden esinlenerek geliştirilmiştir. Fakat burada aşamalı bir yapı söz konusudur. İkinci ve sonraki paketler bir önceki pakette yanlış olarak sınıflandırılan örnekleri de içerecek şekilde oluşturulur. Bir diğer önemli farklılık ise, son kararın sınıflandırıcıların verdikleri sonuçların ağırlıklı ortalaması alınarak bulunmasıdır.

Günümüzde en çok kullanılan yöntem ise boosting algoritmasından türetilmiş olan Adaboost ("Adaptive Boosting") [41] algoritmasıdır. Adaboost algoritmasının ilk boosting algoritmalarından farkı, paketleme yerine örnekleri ağırlıklandırma fikrini getirmesidir. Daha önceki bölümlerde Adaboost algoritması detaylı bir şekilde incelenmişti. Bu bölümde Adaboost algoritmasının sınıflama sonucunu ve bu sonucun güvenilirliğini bir arada veren sınıflandırıcılarla kullanılması halinde daha başarılı sonuçlar elde edilip edilemeyeceği üzerinde durulacaktır.

Özgün Adaboost algoritmasında [16] kullanılan eşik değer sınıflandırıcısı (6.3)'deki gibi formüle edilmiştir. Burada x test edilecek örneği; $f_j(x)$, x örneğinin j. özniteliğini; θ_j, eşik değerini; p_j, eşitsizliğin yönünü ve $h_j(x)$, x örneğinin j. özniteliğini kullanan zayıf sınıflandırıcıyı göstermektedir:

$$h_j(x) = \begin{cases} 1 & \text{eğer } p_j f_j(x) < p_j \theta_j \\ 0 & \text{aksi halde} \end{cases} \qquad (6.3)$$

Sınıflama sonucunun yanı sıra güvenilirliği de veren bir eşik değer sınıflandırıcısı elde etmek için (6.4)'deki denklem kullanılmıştır. Bu denklemde, m_0 ve m_1 sırasıyla negatif ve pozitif örneklerin j. özniteliklerinin ortalamasını göstermektedir. Bu ortalama değerler zayıf sınıflandırıcıların eğitimi sırasında hesaplanmaktadır. Test edilen örneğin ortalama değerlere olan Öklid uzaklığı güvenilirlik skoru ve sınıflama etiketini birlikte belirtir. (6.4)'deki denklem, 0 ve 1 arasında değişen kesirli bir sonuç üretir. Bu sonuç test edilen örneğin pozitif bir örnek olma ihtimalini göstermektedir.

$$h_j(x) = \frac{|m_0 - f_j(x)|}{|m_0 - f_j(x)| + |m_1 - f_j(x)|} \quad (6.4)$$

Bir sınıflandırıcının bir örnek için yaptığı tahmini o örneğin gerçek etiketinden çıkararak tahmindeki hatanın miktarı bulunabilir. (6.4)'deki denklemde x_i, i. örneği; y_i, i. örneğin etiketini; $h_j(x_i)$, j. sınıflandırıcının i. örnek için yaptığı tahmini ve e_{ij} ise bu tahmindeki hatanın büyüklüğünü göstermektedir. Böylece, n adet örnek ve m adet zayıf sınıflandırıcı için, nxm boyutunda bir hata matrisi oluşur.

$$e_{ij} = |h_j(x_i) - y_i| \quad (6.5)$$

5. Bölümde özetlenen çalışmalarda oluşturulan hata matrisi "1" ve "0"'lardan oluşmaktaydı, ancak güvenilirlik skorunun kullanılmasıyla hata matrisi 1 ve 0 arasında değişen değerlere sahip matris haline gelmiştir (Şekil 6.9)

	1. sınıflandırıcı	2. sınıflandırıcı	3. sınıflandırıcı	4. sınıflandırıcı	5. sınıflandırıcı	6. sınıflandırıcı	7. sınıflandırıcı	8. sınıflandırıcı	9. sınıflandırıcı	10. sınıflandırıcı	örneklerin hata oranı
1. örnek	0.6	0.4	0	0.6	0.2	0.8	0.4	0.2	0.2	0.4	0.42
2. örnek	0.4	0	0.2	0.4	0.4	0	0.2	0.2	1	0.2	0.33
3. örnek	0.6	0	0.4	0.4	0.2	0.4	0.2	0.4	0.2	0.4	0.36
4. örnek	0.4	0.4	0.6	0.4	0	0.4	0	0	0.4	0.2	0.31
5. örnek	0.2	0	0	1	0.2	0	0.6	0.6	0.2	0.4	0.36
6. örnek	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.4	0	1	0.47
7. örnek	0.4	0.4	0.8	0.2	0.4	0	0.2	0.6	0.6	0.2	0.42
8. örnek	0.8	0.8	0.6	0.8	1	0.6	0.8	1	0.6	1	0.89
9. örnek	0.4	0	0.2	0	0.4	0.2	0.2	0.4	0.4	0.2	0.27
sınıflandırıcıların hata oranı	0.46	0.29	0.33	0.47	0.38	0.29	0.33	0.42	0.40	0.44	

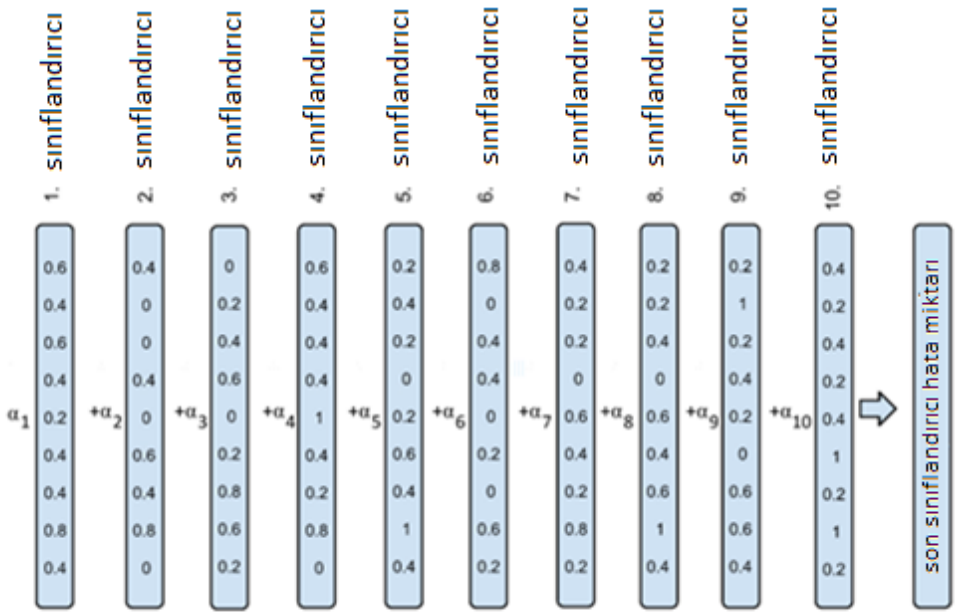
Şekil 6. 9 Sınıflandırıcıların yaptığı tahminlerdeki hata miktarını güvenilirlik skoruyla birlikte gösteren hata matrisi.

Şekil 6.9'da gösterilen hata matrisi örneğinden de görülebileceği üzere hata matrisindeki bir kolon bir sınıflandırıcıyı, bir satır ise bir örnek için tüm sınıflandırıcıların yaptıkları tahminlerdeki hataların büyüklüğünü gösterir. Amaç, bu kolonların doğrusal bir kombinasyonundan oluşan, her örnek için hata büyüklüğü 0.5'in altında olan, yani tüm satırları 0.5'den küçük yeni bir kolon elde etmektir. Burada tüm kolonları kullanmak gibi bir zorunluluk yoktur. Bu hedefi gerçeklemeye yeterli en az sayıda kolonu kullanmak daha az sayıda sınıflandırıcı kullanarak hedefi gerçekleştirmek anlamına geldiğinden tercih sebebidir.

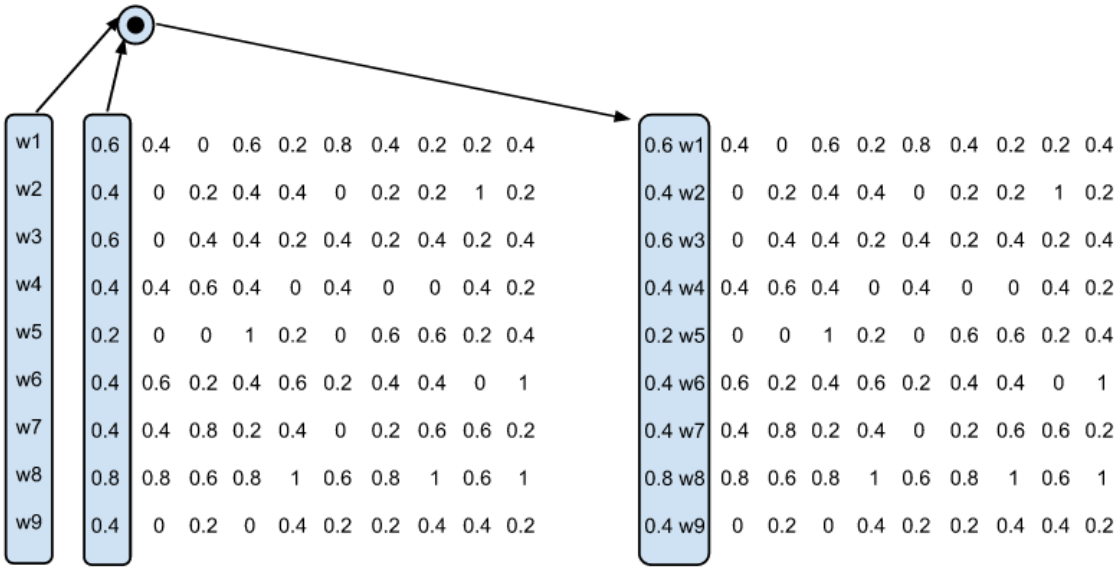
Her kolonun kombinasyondaki ağırlığını gösteren pozitif bir katsayı değeri α_t bulunur (Şekil 6.10). Hedef sınıflandırıcı, mevcut sınıflandırıcıların lineer bir kombinasyonu olduğundan, bu sınıflandırıcının hata miktarı da mevcut sınıflandırıcıların hata miktarlarının lineer kombinasyonudur. Dolayısıyla, bu hata miktarını en aza düşürecek şekilde zayıf sınıflandırıcılar ve katsayıları seçildiğinde hedef sınıflandırıcı da bulunmuş olur. Eğer bir örnek en azından bir sınıflandırıcı tarafından doğru sınıflandırılmadıysa bu örneğin oluşturulan hedef sınıflandırıcı tarafından doğru sınıflandırılması mümkün değildir. Şekil 6.9'daki 8 numaralı örnekte bu durum gösterilmiştir. Eğer mevcut sınıflandırıcıların hiçbiri tarafından sınıflanamayan bir örnek varsa, sisteme yeni bir sınıflandırıcı eklenmesi gereklidir.

Bu yaklaşımın temel problemi hangi kolonların seçileceği ve bu kolonların katsayılarının nasıl belirleneceğidir. En iyi çözümü bulabilmek için olası tüm kombinasyonları denemek, imkansız değilse bile çok zordur. Adaboost bu probleme pratik bir çözüm bulmaktadır, fakat bu çözümün en iyi olduğunu garanti etmez. İlk olarak en az hata oranına sahip kolon seçilir ve bu ilk seçilen kolon, aynı örneklerde hata yapmayan başka bir kolonla birleştirilir. Bu amaçla her örneğe bir ağırlık atanır. İlk başta tüm örneklerin ağırlıkları eşittir, en az hata oranına sahip ilk kolonun seçilmesi ardından bu kolonda hata yapılan örneklerin ağırlıkları arttırılır. Bu ağırlıklar örneklerin birbirlerine göre göreceli olarak önemlerini göstermektedir, böylece hata yapılan örneklerin önemleri yeni bir kolon seçilmeden önce arttırılırmaktadır. Her kolonun hata oranı bu yeni ağırlıklar kullanılarak yeniden hesaplanır (Şekil 6.11). Böylece muhtemelen önceki kolondan farklı yerlerde hata yapan yeni bir kolon seçilir. Bu şekilde kombinasyona her yeni kolonun eklenmesiyle bileşke kolonun hata oranın her iterasyonda düştüğü

gözlenir. Böylece Adaboost algoritması mevcut sınıflandırıcıların her birinden daha güçlü yeni bir sınıflandırıcı oluşturur (Çizelge 6.1).



Şekil 6. 10 Hedef sınıflandırıcı mevcut sınıflandırıcıların lineer bir kombinasyonu alınarak oluşturulur.



Şekil 6.11 Adaboost her örneğe bir ağırlık verir, böylece her iterasyonda ağırlık kolonu zayıf sınıflandırıcı kolonlarla tek tek nokta çarpımı yapılarak ağırlıklandırılmış hata miktarları bulunur.

Çizelge 6.1 Hata matrisini kullanan Adaboost algoritması

- İlk ağırlıklar verilir, $w_{1,i} = \frac{1}{m}, \frac{1}{l}$

  Burada i kaçıncı örnek olduğunu, m ve l sırasıyla pozitif ve negatif örneklerin sayısıdır.

- T adımda yapılacak bir eğitim için $t = 1, \dots, T$:

  1. Ağırlıklar normalize edilir

  $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{s=1}^{n} w_{t,s}}$

  2. Hata matrisi kullanılarak sınıflandırıcıların w_t ağırlığına göre hesaplanan hata miktarları bulunur,

  $\epsilon_j = \sum_i \ w_{t,i} \ e_{ij}$

  3. En düşük ϵ_t hatasına sahip h_t sınıflandırıcısı seçilir.

  4. Ağırlıklar güncellenir:

  $w_{t+1,i} = w_{t,i} \beta_t^{1-e_{ij}}$

  Burada j, seçilen sınıflandırıcının indeksidir ve $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ bu sınıflandırıcının ana sınıflandırıcıdaki ağırlığını gösterir.

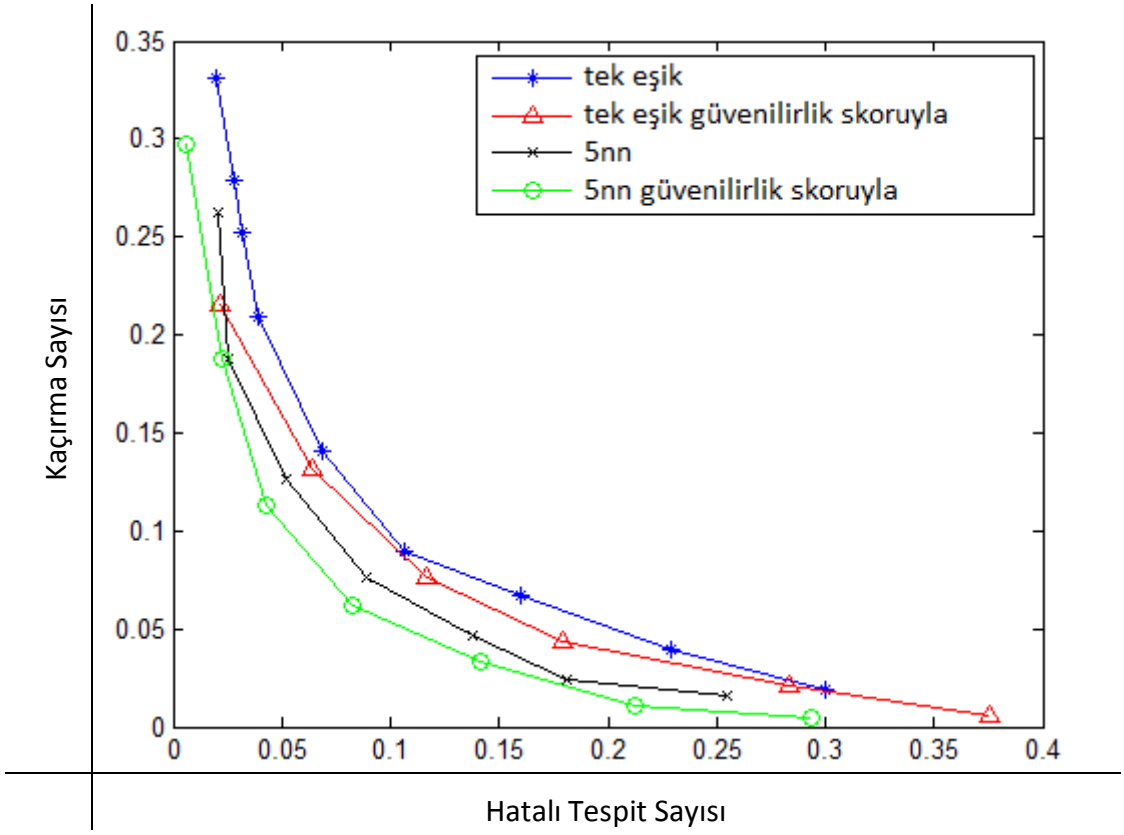
- T adım sonunda oluşturulan ana sınıflandırıcı hipotezi aşağıdaki gibidir:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{diğer türlü} \end{cases}$$

Viola ve Jones'un da 2001 yılındaki mihenk taşı niteliğindeki makalelerinde [16] kullandıkları Adaboost sadece tam sayı değerlerle çalışmaktadır. Bu nedenle örneklerin ağırlıkları sadece hata yapılan örnekler için ve bir iterasyon sırasında tüm hata yapılan örneklerde aynı oranda arttılır. Oysa ki, yukarıda anlatıldığı gibi kesirli bir hata miktarı kullanıldığında bir örnek doğru sınıflandırılsa dahi tahmindeki hata büyüklüğü 0'dan farklıysa, yani sınıflandırıcı bu örneği doğru sınıflandırdığından yeterince emin değilse örneğin ağırlığı arttırılır. Ayrıca her örneğin ağırlığı, o örneği tahminde yapılan hatanın büyüklüğü ya da sınıflandırıcının bu örneği sınıflarken ne kadar emin olduğuna bağlı

olarak farklı oranlarda arttırılır. Çizelge 6.1'de hata matrisini kullanan Adaboost algoritması verilmiştir.

Güvenilirlik oranlarının sistem başarımına etkisi Şekil 6.12'de görülmektedir. Bu çalışmada zayıf sınıflandırıcı olarak en yakın 5 komşu (5nn) kullanılmıştır. Güvenilirlik skoru ise incelenen örneğe en yakın 5 komşunun kaç adedinin örnekle aynı etikete sahip olduğu sayılarak hesaplanmıştır.



Şekil 6.12 Güvenilirlik skoru kullanılmasının tespit performansına etkisi

BÖLÜM 7

SONUÇ ve ÖNERİLER

Bu çalışmada tekil resimler üzerinde aranan nesneleri bulabilen genel bir nesne tespit sisteminin oluşturulması ve bu sistemin yaya tespiti üzerinde uygulanması amaçlanmıştır. Çalışmanın üçüncü bölümünde yayaların dolaylı olarak aranması üzerinde durulmuş, diğer bir ifadeyle yaya bütün olarak aranmaktansa yayayı oluşturan parçalar aranarak yayanın varlığı doğrulanmaya çalışılmıştır. Bu çalışmada şablon eşleştirme ve öznitelik yöntemlerinin karışımı yeni bir yöntem [39] önerilmiştir.

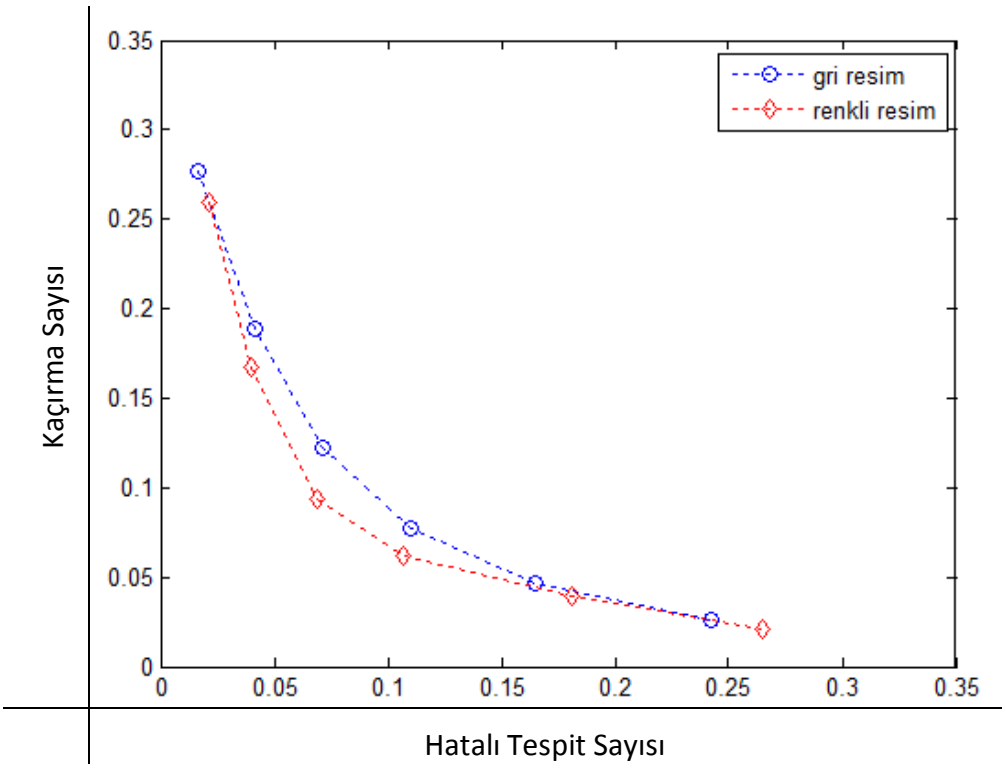
Dördüncü bölümde yayanın bir bütün olarak aranmasını öngören bir yaya tespit sistemi geliştirilmiştir. Bu yaya tespit sistemi [21] bulunan yayanın doğruluğunun sınanması için yayaya has nitelikleri arayan birden fazla sınıflandırıcıyı arka arkaya bağlayan bir çerçeve sunmaktadır. Ancak bu sınanma aşamasında kullanılan sınıflandırıcılar yayalara özel olarak geliştirilmiş sınıflandırıcılardır. Bu durum genel amaçlı bir nesne tespit sistemi oluşturma amacına uygun olmadığı anlaşıldığından çalışmanın sonraki bölümlerinde kullanılmamıştır.

Beşinci bölümde ise yayaların doğrudan bir bütün olarak aranmasını öngören ve öznitelik temelli çalışan bir yaya tespit sistemi geliştirilmiştir. Geliştirilen sistem farklı öznitelikleri bir arada kullanabilmekte, ayrıca farklı öznitelik sınıflandırıcıları ve ana sınıflandırıcıların birlikte kullanılmasına imkan vermektedir.

Altıncı bölümde bu genel nesne tespit sisteminde kullanılabilecek yeni bir öznitelik çıkarım yöntemi, doğrusal olmayan öznitelik sınıflandırıcılar ve güvenilirlik skoru yardımıyla daha güçlü bir ana sınıflandırıcı olan gelişmiş Adaboost [19] önerilmiştir.

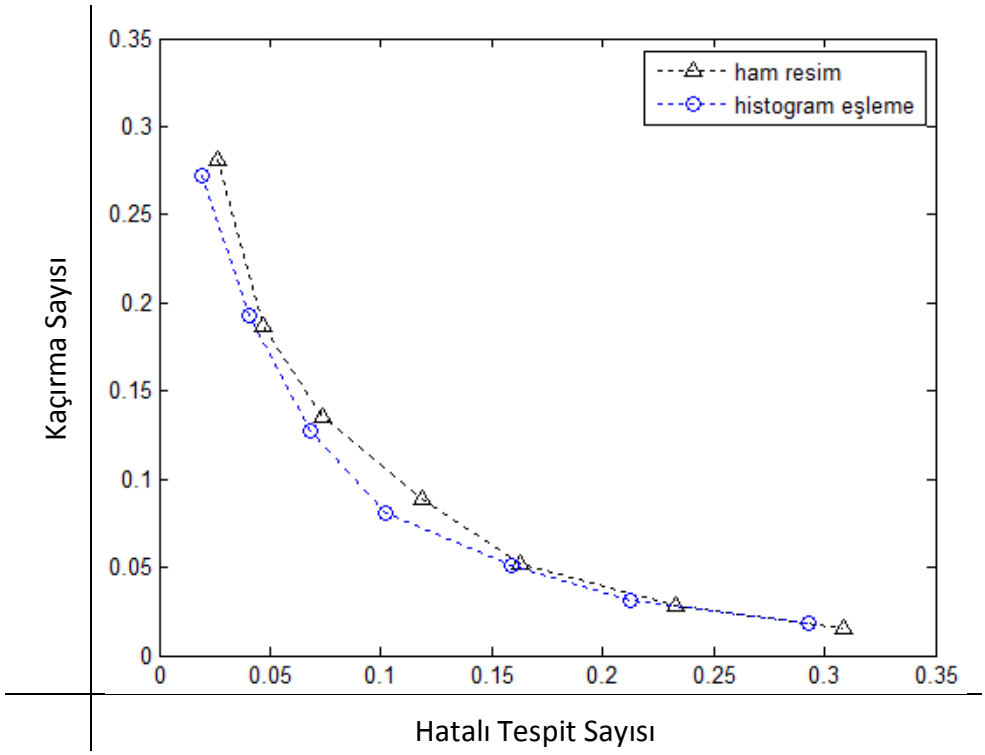
7.1 Önerilen Yaya Tespit Sistemi

Bu çalışmada geliştirilen yaya tespit sistemi tekil resimler üzerinde çalışmaktadır. Bu çalışmada gri tonlu resimler kullanılmıştır fakat geliştirilen tespit sisteminin renkli resimler üzerinde de kullanılmasına bir engel bulunmamaktadır. Renkli resimlerin kullanılması durumunda aynı örnek için tek bir gri tonlu resim işlenmesi yerine 3 adet temel bileşen resmi işlenmektedir. Renk bilgisinden yararlanıldığında yaya tespit sisteminin tespit başarısının küçük bir miktarda arttığı Şekil 7.1'de gösterilmektedir. Renkli resim kullanılması durumunda tespit başarısının dramatik bir şekilde artmaması renkli resim gri tonlu resme çevrildiğinde bilginin büyük bir kısmının korunduğu anlamına gelmektedir.



Şekil 7. 1 Renkli resim kullanmanın tespit performansına etkisi.

Resimler tespit sistemi tarafından kullanılmadan önce resimlerin ortalama parlaklık değerlerinin ayarlanması gerekmektedir. Bunun için kullanılan tüm resimlere histogram dengeleme işlemi uygulanmıştır. Histogram dengelemenin tespit performansına etkisi Şekil 7.2'den görülebilir.



Şekil 7. 2 Histogram dengeleme yöntemiyle parlaklık eşitlemenin tespit performansına etkisi.

7.1.1 Önerilen Öznitelik Bulma Yöntemi

Yaya tespit sisteminde dikdörtgen oranları ve entropik öznitelik bulma yöntemleri birlikte kullanılarak elde edilen öznitelikler kullanılmıştır. Dikdörtgen oranları yönteminin dikdörtgen farkları yönteminden az bir farkla daha başarılı sonuçlar verdiği Şekil 6.4'de görülmektedir. Bunun yanısıra dikdörtgen farkları yöntemi ile dikdörtgen oranları yönteminin birbirini iyi tamamlayamadığı, fakat entropik öznitelik yönteminin bu yöntemleri oldukça iyi tamamladığı Şekil 6.5'te gösterilmiştir. Dolayısıyla her 3 yöntemi de birlikte kullanmak, dikdörtgen oranları ve entropik öznitelik yöntemlerini birarada kullanmaktan çok farklı bir sonuç vermediği için sadece oran ve entropik özniteliklerin kullanılmasına karar verilmiştir.

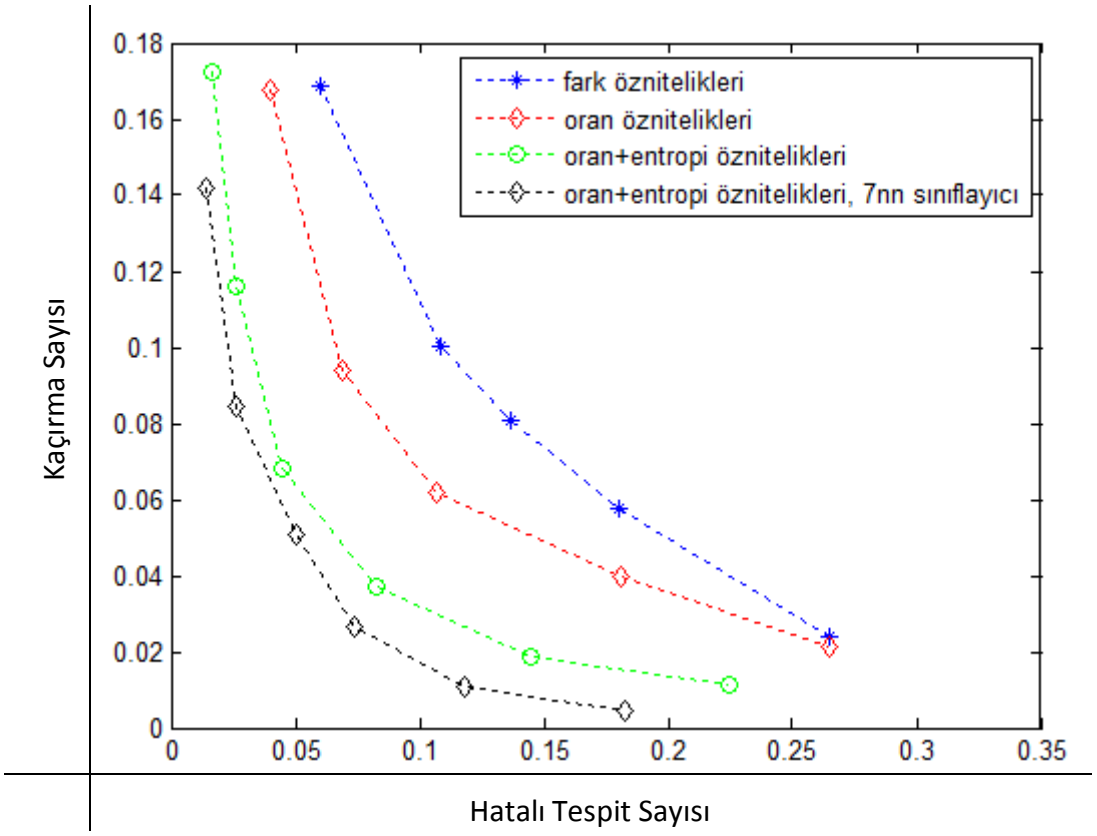
Bu çalışmada öznitelikler hesaplanırken kullanılan en küçük boyut 24x24 piksel olarak alınmıştır. Şablonun pencere içerisindeki ilerleme hızını belirleyen adım büyüklüğü şablon büyüklüğünün 1/3'ü olarak ayarlanmış ve tarama işlemi tamamlandıktan sonra şablon boyutu her seferinde sadece genişlik ya da yükseklik olarak 24 piksel arttırılarak tarama işlemi tekrarlanmıştır. Bu işlem şablon pencereye sığdığı sürece devam

ettirilmiştir. Böylece 64x128 piksellik bir pencere için Şekil 5.4'deki 5 adet şablon yardımıyla 690 adet öznitelik bulunmuştur. Bu sayının benzer çalışmalara kıyaslandığında oldukça düşük sayıda kaldığı gözlenir. Örneğin Viola ve Jones [16], 20x15 piksellik bir pencere için 24328 öznitelik hesaplamaktadır, keza Enzweiler [7] 134.621 adet öznitelik kullanmaktadır.

7.1.2 Önerilen Öznitelik Sınıflandırıcı

Bu çalışmada öznitelik sınıflandırıcı olarak en yakın komşu sınıflandırıcısı kullanılmıştır. Bu sınıflandırıcının tercih edilmesinin sebebi sınıflandırma sonucuyla birlikte güvenilirlik skorunu da verebilmesidir. Ayrıca doğrusal olmayan sınıflandırmaları yapabilmesi de önemlidir. Öznitelik sınıflandırıcı olarak karar ağaçlarının kullanılması da tercih edilebilirdi. Ancak yapılan denemeler sonucunda karar ağaçlarının en yakın komşu sınıflandırıcısı kadar performans sergileyemedikleri görülmüştür. Geliştirilen yaya tespit sistemi herhangi bir sınıflandırıcının kullanılmasına imkan tanımaktadır. Dolayısıyla destek vektör makinaları veya yapay sinir ağlarının da öznitelik sınıflandırıcısı olarak kullanılması mümkündür. Fakat bu sınıflandırıcıların eğitilmeleri, test edilmeleri ve saklanmaları en yakın komşu sınıflandırıcısı kadar kolay olmadığından tercih edilmemiştir. Örneğin yapay sinir ağlarının kullanıldığı durumda 87.000 adet öznitelik için 87.000 yapay sinir ağının eğitilmesi ve test edilmesi gerekmektedir. Bu ise işlemsel yük açısından pratik olmanın çok uzağındadır.

Yapılan bu geliştirmeler neticesinde oluşturulan yaya tespit sistemi ve her geliştirmenin sistemin performansına ne kadar etki ettiği Şekil 7.3'den görülebilir. Sonuç olarak, ilk önce fark öznitelikleri yerine oran öznitelikleri kullanılarak tespit başarısı arttırıldı. Daha sonra oran öznitelikleriyle birlikte entropik özniteliklerin de kullanılmasın sağlanarak tespit başarısında yüksek bir artış sağlandı. Sonraki adımda ise zayıf sınıflandırıcı olarak basit eşik değer sınıflandırıcısı yerine doğrusal olmayan bir sınıflandırıcı (7-NN) kullanılarak başarının arttırılması sağlanmıştır. Son aşamada ise ana sınıflandırıcının, zayıf sınıflandırıcıların ne kadar güvenli olduğunu da değerlendirmesi sağlanarak tespit başarısında başlangıçtaki duruma göre dramatik bir artış sağlanmıştır.



Şekil 7. 3 Yapılan geliştirmelerin tespit başarısına katkısı.

7.2 Sonuçlar

Bu tezde yapılan çalışmalar sonucunda yayaların tespitinde olduğu gibi başka nesnelerin tespitinde de kullanılabilecek genel bir nesne tespit sistemi oluşturulmuştur. Sınıflandırıcı olarak kullanılan Adaboost algoritması hata matrisi üzerinden çözülebilecek bir optimizasyon problemine indirgenmiş ve bu problemin cebirsel çözümünün her zaman mümkün olmadığı gösterilmiştir. Bu nedenle problemin çözümüne yönelik yinelemeli bir yöntem önerilmiştir. Bu yöntem sınıflandırıcıların, sınıflama sonuçlarının yanısıra bu sonuçları ne kadar güvenilirlikle verdiklerini gösteren bir güvenilirlik skorunu kullanmaları sağlanarak geliştirilmiştir. Hata matrisi üzerinden üzerinden gerçekleştirilen çözüm ihtiyaca göre hatalı pozitif ya da negatiflere daha duyarlı olan bir sınıflandırıcı üretebilmektedir.

Yaya tespit sisteminin başarımını arttırmak için resmin entropisini kullanan yeni bir öznitelik bulma yöntemi geliştirilmiş ve bu yöntemin literatürde mevcut olan diğer yöntemlerle birlikte kullanıldığında tespit doğruluğunu dramatik bir şekilde arttırdığı

gösterilmiştir. Ayrıca, işlemsel karmaşası fazla olan entropik özniteliklerin hızlı bir şekilde hesaplanabilmesi için yeni bir yöntem önerilmiştir.

Adaboost, öğrenme işlemi boyunca aynı anda ilgisiz öznitelikleri de elemekte, bir başka deyişle öznitelik seçimi yapmaktadır. Bu alanda kullanılan öznitelik bulma yöntemleri genellikle görüntüdeki piksel sayısından daha fazla öznitelik üretmekte, daha sonra ilgisiz öznitelikler çoğunlukla Adaboost, ya da benzer bir algoritma ile elenmektedir. Gelecekte yapılacak çalışmalarda Adaboost yerine özniteliklerin birbirleri ile etkileşimlerini de değerlendirebilecek bir öznitelik seçme yönteminin kullanılması düşünülmektedir. Ayrıca görüntüdeki nesneleri daha iyi temsil edecek yeni özniteliklerin tasarlanması da önemli bir problem olarak görülmektedir.

KAYNAKLAR

[1] Zhao, L. ve Thorpe, C.E., (2000). “Stereo and neural network-based pedestrian detection”, IEEE Trans. On Intelligent Transport Systems, 1(3):148-154.

[2] Broggi, A., Bertozzi, M., Fascioli, A. ve Sechi, M., (2000). “Shape-based pedestrian detection”, IEEE Intelligent Vehicles Symposium, 215-220.

[3] Liu, Y., Chen, X., Yao, H., Cui, X., Liu, C. ve Gao, W., (2009). “Contour-motion feature (CMF): a space-time approach for robust pedestrian detection”, Pattern Recognition Letters, 30(2):148-156.

[4] Wang, H., Lu, R., Wu, X., Zhang, L. ve Shen, J., (2009). “Pedestrian detection and tracking algorithm design in transportation video monitoring system”, International Conference on Information Technology and Computer Science, 53-56.

[5] Xu, F., Liu, X. ve Fujimura, K., (2005). “Pedestrian detection and tracking with night vision”, IEEE Transactions on Intelligent Transportation Systems, 6(1):63-71.

[6] Zheng, G., ve Chen, Y., (2012). “A review on vision-based pedestrian detection.” In Global High Tech Congress on Electronics (GHTCE), 1:49-54.

[7] Enzweiler, M. ve Gavrila, D. M., (2009). “Monocular pedestrian detection: survey and experiments”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(12):2179-2195.

[8] Cao, X. B., Xu, Y. W., Chen, D. ve Qiao, H., (2009). “Associated evolution of a support vector machine-based classifier for pedestrian detection”, Information Sciences, 179:1070-1077.

[9] Shashua, A., Gdalyahu, Y. ve Hayun, G., (2004). “Pedestrian detection for driving assistance systems: single-frame classification and system level performance”, IEEE Intellligent Vehicles Symposium, 1- 6.

[10] Oren, M., Papageorgiou, C., Sinha, P., Osuna, E. ve Peggio, T., (1997). “Pedestrian detection using wavelet templates”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 193-199.

[11] Dollar, P., Wojek, C., Schiele, B., Perona, P., (2012). "Pedestrian Detection: An Evaluation of the State of the Art," IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(4):743-761.

[12] Li, B., Yao, Q., ve Wang, K., (2012). “A review on vision-based pedestrian detection in intelligent transportation systems.” In Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on, 1: 393-398.

[13] Dai, C., Zheng, Y. ve Li, X., (2007). “Pedestrian detection and tracking in infrared imagery using shape and appearance”, Computer Vision and Image Understanding, 106(2-3):288-299.

[14] Premebida, C., Ludwig, O. ve Nunes, U., (2009). “LIDAR and vision-based pedestrian detection system”, Journal of Field Robotics, 26(9):696-711.

[15] Leibe, B., Seamann, E. ve Schiele, B., (2005). “Pedestrian detection in crowded scenes”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1:878-885.

[16] Viola, P. ve Jones, M., (2001). “Rapid object detection using a boosted cascade of simple features.”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1:511-518.

[17] Dollár, P., Belongie, S., & Perona, P., (2010). “The Fastest Pedestrian Detector in the West.” In BMVC, 2(3):7.

[18] Dollár, P., Tu, Z., Perona, P., & Belongie, S., (2009). “Integral Channel Features.” In BMVC 2(4):5.

[19] Tetik, Y. E. ve Bolat, B., (2013). “Pedestrian detection with an improved adaboost”, Innovations in Intelligent Systems and Applications (INISTA), 1:1-4.

[20] Tetik, Y. E. ve Bolat, B., (2013). “A Pedestrian detection system with weak classifiers”, IEEE SIU 21st Signal Processing and Communication Applications Symposium, 1:1:4.

[21] Tetik, Y.E. ve Bolat, B., (2011). "Pedestrian Detection From Still Images", Innovations in Intelligent Systems and Applications (INISTA), 1:540-544.

[22] Papageorgiou, C. P., Oren, M. ve Poggio, T., (1998). “A general framework for object detection.” Sixth International Conference on Computer Vision,1:555-562.

[23] Freeman, W. T. ve Roth, M., (1995). “Orientation histograms for hand gesture recognition.”, In International Workshop on Automatic Face and Gesture Recognition, 12:296-301.

[24] Lowe, D.G., (2004). “Distinctive image features from scale-invariant keypoints.”, International journal of computer vision, 60(2):91-110.

[25] Dalal, N. ve Triggs, B., (2005). “Histograms of oriented gradients for human detection.”, In Computer Vision and Pattern Recognition IEEE Computer Society Conference, 1:886-893.

[26] Min, K., Son, H., Choe, Y., ve Kim, Y. G. (2013, June). “Real-time pedestrian detection based on A hierarchical two-stage Support Vector Machine.” In Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on 1:114-119.

[27] Mogelmose, A., Prioletti, A., Trivedi, M. M., Broggi, A., ve Moeslund, T. B., (2012). “Two-stage part-based pedestrian detection.” In Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on 1:73-77.

[28] Li, W., Zheng, D., Zhao, T., ve Yang, M., (2012). “An effective approach to pedestrian detection in thermal imagery.” In Natural Computation (ICNC), 2012 Eighth International Conference on, 1:325-329.

[29] Xu, T., Liu, H., Qian, Y., ve Wang, Z. (2012, July). “A fast and robust pedestrian detection framework based on static and dynamic information.” In Multimedia and Expo (ICME), 2012 IEEE International Conference on,1:242-247.

[30] Hahnle, M., Saxen, F., Hisung, M., Brunsmann, U., ve Doll, K. (2013). “FPGA-Based Real-Time Pedestrian Detection on High-Resolution Images.” In Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, 1:629-635.

[31] Wei, D., Zhao, Y., Cheng, R., ve Li, G. (2013). “An enhanced Histogram of Oriented Gradient for pedestrian detection.” In Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference on 1:459-463.

[32] Sabzmeydani, P. ve Mori, G., (2007). “Detecting pedestrians by learning shapelet features”, IEEE Conference on Computer Vision and Pattern Recognition, 1:1-8.

[33] Wu, B. ve Nevatia, R., (2005). “Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors.”, Tenth IEEE International Conference on Computer Vision, 1:90-97.

[34] Yao, W. ve Deng, Z., (2012). “A robust pedestrian detection approach based on shapelet feature and Haar detector ensembles.” Tsinghua Science and Technology, 17(1):40-50.

[35] Yongzhi, W., Jianping, X., Xiling, L. ve Jun, Z. (2010). “Pedestrian Detection Using Coarse-to-Fine Method with Haar-Like and Shapelet Features.”, In Multimedia Technology (ICMT), 2010 International Conference on, 1:1-4.

[36] Hong, T., Lixia, W. ve Xiaoqing, D., (2011). “Pedestrian detection based on merged cascade classifier.” In Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of, 1:237-241

[37] Byun, K.Y., Kim, B.S., Kim, H.K., Shin, J.E. ve Ko, S.J., (2012) "An effective pedestrian detection method for driver assistance system", IEEE International Conference on Consumer Electronics (ICCE), 1:229-230.

[38] Shankar, U., (2003), “Pedestrian Roadway Fatalities”, Technical Report of Dept. of Transportation.

[39] Tetik, Y.E. ve Bolat, B., (2011). “Detection of Pedestrians From Still Images”, IEEE SIU 19th Signal Processing and Communication Applications Symposium, 1:670:673.

[40] Sinha, P., (1994). ''Object Recognition via Image Invariants: A Case Study'', Investigation Ophthmology and Visual Science, (35):1735-1740.

[41] Freund, Y., (1995). "Boosting a weak learning algorithm by majority.", Information and computation 121(2): 256-285.

[42] Overett, G., Petersson, L., Brewer, N., Andersson, L. ve Pettersson, N., (2008). “A new pedestrian dataset for supervised learning.” IEEE Intelligent Vehicles Symposium, 1:373-378.

[43] Wang, L., Shi, J., Song, G. ve Shen, I., (2007). “Object Detection Combining Recognition and Segmentation”, Proceedings of 8th Asian Conference on Computer Vision (ACCV), 1:189-199.

[44] Everingham, M., Van Gool, L., Williams, C. K., Winn, J. ve Zisserman, A., (2010). “The pascal visual object classes (voc) challenge.” International journal of computer vision, 88(2):303-338.

[45] Wojek, C., Walk, S. ve Schiele, B., (2009). “Multi-cue onboard pedestrian detection.” In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 1:794-801.

[46] Dollár, P., Wojek, C., Schiele, B. ve Perona, P. (2009). “Pedestrian detection: A benchmark.” In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 1:304-311.

[47] Martin, A., Doddington, G., Kamm, T., Ordowski, M. ve Przybocki, M. (1997). “The DET curve in assessment of detection task performance.” National Inst. of Standarts and Technology Gaithersburg Md.

[48] Sinha, P., (1994). ''Qualitative image-based representations for object recognition'', MIT AI Lab Memo No.1505.

[49] Papageorgiou, C. P., ve Poggio, T., (1999). “Trainable Pedestrian Detection.” International Conference on Image Processing,4:35-39.

[50] Freund, Y. ve Schapire, R.E., (1997). “A decision-theoretic generalization of on-line learning and an application to boosting”, Journal of Computer and System Sciences, 55:19-139.

[51] Gaoa, C., Sang, N. ve Tanga, Q., (2010). “On selection and combination of weak learners in AdaBoost”, Pattern Recognition Letters, 31(9):991-1001.

[52] Viola, P., Jones, M. J., & Snow, D., (2005). “Detecting pedestrians using patterns of motion and appearance.” International Journal of Computer Vision, 63(2):153-161.

[53] John, G. H., Kohavi, R. ve Pfleger, K. (1994). “Irrelevant Features and the Subset Selection Problem.”, In ICML, 94:121-129.

[54] Shannon, C., (1948). “A mathematical theory of communication,” The Bell System Technical Journal, 27:379-423,623-656.

[55] Sabuncu, M. R., (2006). Entropy-based Image Registration, Doktora Tezi, Princeton University, Princeton.

[56] Tsai, P. S., ve Wu, M. H., (2005). "Entropy-based 2D image dissimilarity measure.", Multimedia Signal Processing, 2005 IEEE 7th Workshop on., 1:1-4.

[57] Guo, X., ve Wang, W., (2006). "Image matching algorithm based on subdivision wavelet and local projection entropy.", Intelligent Control and Automation, The Sixth World Congress on. 2:10380-10383.

[58] O’Brien, D., Cassini Lossy Compression Software Tests, http://pds-rings.seti.org/cassini/iss/COISS_0011_DOCUMENT/report/iss/5.0/5.5/5.5.1_lossy_com.pdf, 30 Kasım 2013.

[59] Beirlant, J., Dudewicz, E. J., Györfi, L. ve Van der Meulen, E. C., (1997). “Nonparametric entropy estimation: An overview.”, International Journal of Mathematical and Statistical Sciences, 6:17-40.

[60] Hero III, A. O., Ma, B., Michel, O. J. ve Gorman, J., (2002). “Applications of entropic spanning graphs.” Signal Processing Magazine, IEEE, 19(5):85-95.

[61] Ess, A., Leibe, B. ve Van Gool, L., (2007). “Depth and appearance for mobile scene analysis.” In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, 1:1-8.

[62] Rasolzadeh, B., Petersson, L. ve Pettersson, N., (2006). "Response binning: Improved weak classifiers for boosting." Intelligent Vehicles Symposium, 2006 IEEE. IEEE, 2006.

[63] Schapire, Robert E., (1989). "The strength of weak learnability.", 30th Annual Symposium on Foundations of Computer Science 1:28-33.

[64] Efron, B. ve Tibshirani, R. (1993). “An introduction to the bootstrap.”, Vol. 57. Chapman & Hall/CRC, 1993.

[65] Breiman, L., (1996). "Bagging predictors.", Machine learning, 24(2):123-140.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Yusuf Engin TETİK

Doğum Tarihi ve Yeri : 01.03.1983 Adıyaman

Yabancı Dili : İngilizce

E-posta : yusufengin@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Y. Lisans	Elektronik-Haberleşme	İzmir Yüksek Teknoloji Enst.	2008
Lisans	Bilgisayar	Ege Üniversitesi	2005
Lise	Fen	Açı Koleji	2001

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2009-2014	TTNET	Video Cihazları Yöneticisi
2008-2009	CABOT Communications	Yazılım Geliştirme Müh.
2006-2008	VESTEL, Vestek Ar-Ge	Tasarım Mühendisi
2005-2006	VESTEL, Vestelkom Ar-Ge	Yazılım Geliştirici

YAYINLARI

Makale

1. Erdur, R.C., Dikenelli, O., Önal, A., Gümüs, Ö., Kardas, G., Bayrak, Ö., Tetik, Y.E., 2005. “A Pervasive Environment for Location-Aware and Semantic Matching Based Information Gathering”, ISCIS.

Bildiri

1. Tetik, Y. E. ve Bolat, B., (2013). “Pedestrian detection with an improved adaboost”, Innovations in Intelligent Systems and Applications (INISTA), 1:1-4

2. Tetik, Y. E. ve Bolat, B., (2013). “A Pedestrian detection system with weak classifiers”, IEEE SIU 21st Signal Processing and Communication Applications Symposium, 1:1:4.

3. Tetik, Y. E. ve Bolat, B., (2011). "Pedestrian Detection From Still Images", Innovations in Intelligent Systems and Applications (INISTA), 1:540-544.

4. Tetik, Y. E. ve Bolat, B., (2011). “Detection of Pedestrians From Still Images”, IEEE SIU 19th Signal Processing and Communication Applications Symposium, 1:670:673.

5. Tetik, Y. E. ve Bolat, B. (2011). “Gürültülü ortamlarda konuşma tespiti için yeni bir öznitelik çıkarım yöntemi”, Fırat Üni., Elektrik Elektronik Bilgisayar Sempozyomu (FEEB) 1:86-89.

Patent

1. Tetik, Y.E., Ateskan, Y.S., Coskun, O., Sahin, A., (2012). “Streaming Media Player and Method “, US Patent Trademark Office Patent. No: US8195829

2. Tetik, Y.E., Sahin, A., (2011). “A Method for Image Compression“ European Patent Office Patent No:EP2360928

3. Tetik, Y.E., Sahin, A., (2011). “Block Motion Searching Device and Method“ European Patent Office EP2373032

4. Coskun, O., Sahin, A., Tetik, Y.E., Ateskan, Y.S., (2008). “Method and Apparatus for Transcoding a Video Signal “, US Patent Office Pub. No: US 2008/0304570 A1

Proje

1. IPTV (Tivibu) Projesi

   STB’nin MW, VOD sunucuları ve diğer servislerle entegrasyonu

   Kullanıcı Arayüzü yazılımı

   Android ve iOS uygulamaları

2. Video Kodlayıcı Projeleri

   H261 Video kodlayıcı ve çözücü

   MJPEG/H264 RTP,RTSP sunucu ve oynatıcı

   H264 (MPEG-4 Part 10) Video çözücü iyileştirme

3. Görüntü İyileştirme Projesi, DCT Katsayıları Yardımıyla Görüntü Keskinleştirme

4. Gömülü Sistemler için Media Oynatıcılar

   ST & VIA chipset üzerinde gömülü media oynatıcısı

   Arm11 üzerinde taşınabilir media oynatıcı

5. GPS, GPRS ve anlamsal eşleme teknolojilerini birarada kullanan konum tabanlı bilgi toplama sistemi. (Ege Üniversitesi’nin 10.000 TL sponsorluğu bulunmaktadır.)