T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İÇERİK TABANLI RESİM ARAMA MOTORU

MEHMET ZAHİD YÜZÜGÜLDÜ

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
HABERLEŞME PROGRAMI

DANIŞMAN
PROF. DR. ABDULLAH BAL

İSTANBUL, 2015

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İÇERİK TABANLI RESİM ARAMA MOTORU

Mehmet Zahid YÜZÜGÜLDÜ tarafından hazırlanan tez çalışması 27.11.2015 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Abdullah BAL

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Prof. Dr. AbdullahBAL

Yıldız Teknik Üniversitesi _____

Doç. Dr. Fethullah KARABİBER

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Yusuf ŞAHİN

Marmara Üniversitesi _____

ÖNSÖZ

Öncelikle, bilgisayarla görü alanı ile tanışmamı sağlayan tecrübesi ve bilgileri ile her zaman yanımda olan, bana çalışmamı tamamlamam için her konuda azami destek veren değerli danışman hocam Prof. Dr. Abdullah BAL'a sonsuz şükranlarımı sunuyorum. Umarım ileriki yıllarda tekrar kendisi ile çalışma fırsatı bulabilirim.

Aralarına katıldığım ilk günden itibaren bana gösterdikleri sıcak ilgi ve sunulan çalışma imkânları nedeni ile Yıldız Teknik Üniversitesi bünyesinde Elektronik ve Haberleşme Mühendisliği Bölümü ailesine teşekkür ederim.

Bölüme başladığım günden bu yana bana gösterdikleri sıcak ilgi ve alaka ile farklı bir bölümde yüksek lisans yapma zorluklarını kolaylıkla atlamamı sağlayan Elektronik ve Haberleşme Bölümü'nde görevli yakın arkadaşlarıma ve Tübitak'ta beraber çalıştığım proje arkadaşlarıma teşekkürlerimi sunar hayatlarında başarılar dilerim.

Son olarak, hiç bir teşekkürün yeterli olmayacağı, çalışmalarımı yapabilmem için bana hiçbir zaman desteğini esirgemeyen, hayatımın her noktasında üzerimde ki emeklerini hissettiğim değerli aileme teşekkür ederim.

Kasım, 2015

Mehmet Zahid YÜZÜGÜLDÜ

İÇİNDEKİLER

SİMGE LİSTESİ

F	Öznitelik matrsi
N	Öznitelik sayısı
θ	Anahtar noktanın döngüsel açısı
ρ	Uzaklık metriği
I(p)	P özniteliğinin yoğunluğu
μ	Ortalama
σ	Standart sapma
m	Gradyent
T	Eşik değeri

KISALTMA LİSTESİ

BovW	Bag of Visual Words
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Independent Elementary Features
BRDH	Bulanık Renk Doku Histogramı
DoG	Difference of Gaussians
EMD	Earth Mover's Distance
FREAK	Fast Retina Keypoint
FCTH	Fuzzy Color Texture Histogram
GFS	Google File System
HDFS	Hadoop Distrubuted File System
HRAN	Hızlı Retina Anahtar Noktası
IBM	International Business Machines Corporation
IDF	Inverse Document Frequency
İVTYS	İlişkisel Veri Tabanı Yönetim Sistemleri
LoG	Laplacian of Gaussian
LTrPs	Local Tetra Patterns
LSH	Locality Sensetive Hashing
NoSQL	Not Only SQL
ORB	Oriented FAST and Rotated BRIEF
ÖBÖD	Ölçekten Bağımsız Öznitelik Dönüşümü
QBIC	Query By Image Content
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
TF	Term Frequency

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

ÖZET

İÇERİK TABANLI RESİM ARAMA MOTORU

Mehmet Zahid YÜZÜGÜLDÜ

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Prof. Dr. Abdullah BAL

Veri miktarının sürekli büyüdüğü günümüzde doğru ve güvenilir bilgiye ulaşmak oldukça büyük bir önem arz etmektedir. Bilgi en büyük güçtür ve bilgiye en kısa sürede ulaşanlar tüm yarışlarda bir adım öne geçeceklerdir. Hemen her gün insanlar internet üzerinde, bilgisayarlarında, e-posta kutularında, fotoğraf arşivlerinde bir şeyler aramaktadır. Arama ve bilgiye ulaşma günlük hayatın önemli bir rutini haline gelmiştir. Merak edilen hemen her şey arama motorlarından öğrenilebilinmektedir. 2014 yılında Google'da günde ortalama 5,8 milyar kelime aranmıştır [1]. Google'ın insan hafızasına zarar verip tembelleşmeye ittiğini bilim insanları ifade etmiştir [2].

Farklı türden bilgilerinişlenip belli bir değer haline getirilmesinde veri miktarının büyüklüğü nedeni ile zorluklarla karşılaşılmaktadır. Bu zorluklara verinin saklanması dahil edilince yapılmak istenen analiz çalışmaları haftalar hatta aylar alabilmektedir. Büyük hacimli verileri hızlı bir şekilde okumak ve işleyebilmek için Büyük Veri teknolojileri geliştirilmiştir.

Bu tezde içerik tabanlı resim arama motoru yapabilmek için Büyük Veri teknolojilerini kullanarak örnekteşkil edecek bir mimari tasarlanmıştır. Veri seti internette bulunan haber sitelerinden sağlanmıştır. 20 haber sitesinin manşet haberleri günde 2 defa olmak üzere bir hafta boyunca taranmıştır. Taranılan haberlerde bulunan, haber ile ilişkili resimler Apache HBASE veri tabanına kaydedilmiştir. Bu şekilde yaklaşık 200bin resimlik bir veri seti oluşturulmuştur. Daha sonra kaydedilen resimlerin öz nitelikleri, MapReduce çatısı altında görüntü işleme algoritmaları yardımı ile çıkartılıp Apache Solr içerisinde indekslenmiştir.

Resim sayısının her an artıyor olması, arama işlemi sonucunun dönüş zamanını da artırmaktadır. Bu işlemi sistemin başarısını düşürmeden yapabilmek için kademeliarama

mimarisi tasarlanmıştır. Kademeli arama mimarisi, veri setinde sorgulanan resim ile alakası olmayan resimler elendikten sonra daha başarılı görüntü işleme algoritması ile sıralanarak yapılmaktadır. Bu noktada ilk eleme işlemi kullandığımız diğer algoritmalara göre çok daha hızlı olan Bulanık Renk Doku Histogramı (BRDH - FCTH) algoritması ile yapılmıştır. Bu aşamada 200bin resim 500 resme indirgenerek sorgulanan resim ile net olarak ilişkisi olmayan resimler elenmiştir.

Eleme işleme sonrasında 500 adet resim içerisinde sorgulanan resme benzeyen resimler Hızlı Retina Anahtar Noktası (HRAN – FREAK) ve Ölçekten Bağımsız Öznitelik Dönüşümü (ÖBÖD – SIFT) algoritmaları ile sıralandırma yapılmıştır. Sıralama sonrasında doku olarak benzerlik yakalanamayan fakat renk dağılımında belirgin benzerlik bulunan resimler de Renk Haritalama algoritması yardımı ile benzerlik sıralamasında yerini almıştır. Sıralanan 500 resmin ilk 60 resmi kullanıcıya sunulmuştur.

Sunulan resimlerden üst sıralarda bulunanların, sorgulanan resme daha çok benzemesi beklenmektedir. Başarı ölçümüleri geri getirme oranı (recall) ve kesinlik (precision) değerleri ile yapılmıştır. Geri getirme oranı, beklenilen resimlerden kaç tanesinin ilk sıralarda yer aldığına bakılarak yapılmıştır ve %92 olarak hesaplanmıştır. Kesinlik değerinin hesaplanabilmesi için, benzerlik oranına eşik değeri uygulanmıştır. Farklı eşik değerlerindeki kesinlik ve geri getirme oranı değerleri sonuç bölümünde tablo halinde verilmiştir. Eşik değeri artırılmasıyla yapılan analizler, kesinlik değeri artarken geri getirme oranının azaldığını göstermiştir.

Anahtar Kelimeler: BüyükVeri, görüntü işleme, arama motoru, resim öz nitelik çıkarma, SIFT, FREAK, FCTH, MapReduce, Apache Hadoop, Apache Solr, Apache HBASE

ABSTRACT

CONTENT BASED IMAGE SEARCH ENGINE

Mehmet Zahid YÜZÜGÜLDÜ

Department of Electronics and Communications Engineering

MSc. Thesis

Adviser: Prof. Dr. Abdullah BAL

Nowadays, since the amount of the information is constantly increasing, reaching accurate and clear data is gaining a great importance. Information has an enormous strength and people who attain it first move one step further at all levels. People are searching everything on the internet, and in their computers, e-mails, as well as photograph archives almost every day. Hence, searching and reaching the information became an everyday routine. People are able to learn almost anything which they are curious about. For instance, 5.8 billion words were searched per day using Google [1] in 2014. However scientists argue that not only Google damages the minds but also causes laziness [2].

Since there is enormous data and information out there, it is very hard to process different types of information and create a value. In addition, a requested analysis may take long time periods such as weeks or even months, when storing the information is added to these difficulties. In order to read and process quickly big amount of data, Big Data technologies are developed.

In this thesis, to manage content-based image search engine, we built a potential architectural design using Big Data technologies. Data set is prepared from news websites on the internet. Headline news of 20 news websites are scanned twice a day for a week. Images are stored in Apache HBASE database. By doing so, a dataset containing 200 thousand images is prepared. The features of images are extracted by

using MapReduce framework with the help of image processing algorithms and then extracted features are indexed by using Apache Solr.

Increasing the number of pictures will also increase the response time. To solve this problem we designed cascading search architecture by using different image processing algorithms. In cascading search architecture, images which are irrelevant to queried image inside the data set are filtered and then ranked with the help of more prosperous image processing algorithms. At this point we choose Fuzzy Color Texture (FCTH) algorithm which has less response time then other image processing algorithms compared (FREAK, SIFT, SURF, BRIEF, BRISK, ORB). By doing so the number of the pictures is decreased from 200 thousand to 500, approximately in 0.2 seconds.

After filtering stage, 500 images are ranked according to similarity of the queried image with Fast Retina Keypoints (FREAK) and Scale Invariant Feature Transform (SIFT) algorithms respectively. Finally, the images having different types of texture but having similarity are ordered again among all the peers with the help of Color Mapping algorithms.

The similary between query and result images decreases according to the descending order of result images. The performance of systems is measered by recall and precision values. Recall measured as 92 % without treshold usage. To measure precision, 6 different tresholds are used. We used treshold values ascending order and relaize that recall value decreases if precision is increasing.

Keywords: Big Data, Content Based Image Retrieval, Search Engine, Image feature extraction, SIFT, FREAK, FCTH, MapReduce, Apache Hadoop, Apache Solr, Apache HBase

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Resim arama ve büyük veri tabanlarında benzerlik bulma konuları 1970 başından bu yana aktif bir çalışma alanıdırve literatürde birçok açıdan yapılan çalışmalar bulunmaktadır. Resim veri tabanlarının amacı, resimleri saklamak ve ihtiyaç olduğu zaman sorgulanan resimle ilgili resimleri elde etmeye olanak sağlamasıdır[3]. Bu alanda yapılan çalışmalar şu maddelerde toparlanabilir.

Yazı temelli resim arama

İçerik temelli resim arama

1.1.1 Yazı Temelli Resim Arama Sistemleri

Bu tip sistemler resim içeriğinden ziyade resmin ne ile alakalı olduğunu önemsemektedir. Resim içerisinden bilgiler çıkartılırken, nesneler sınıflandırılarak yada resim içerisinde bulunduğu web sayfasının konusu ile ilişkilendirilerek hazırlanan sistemlerdir. Bu tür sistemlere Google, Yahoo gibi arama motorlarının resim arama özellikleri gösterilebilir.

1.1.2 İçerik Tabanlı Resim Arama Sistemleri

1992 yılında T. Kato resimlerin renk ve şekillerine göre arama yapılabileceğini açıklamış ve bunun için çalışmalar yapmıştır [2]. Profesyonel bir uygulama aynı yıllarda Query By Image Content (QBIC, Resim İçeriği ile Sorgulama) adı ile IBM tarafından gerçekleştirilmiştir.[3]

Resim özellikleri, renk, doku, şekil bilgilerine göre belli algoritmalarca çıkartılır ve resim benzerlikleri özellikler arasındaki uzaklıklara göre hesaplanmaktadır. Günümüzde Google, Yandex, Yahoo, Tineye gibi internetteki resimleri sürekli tarayan ve anlık aramaya olanak sağlayan içerik tabanlı resim arama sistemleri mevcuttur.

Öz nitelikleri çıkartılmış resimlerin Büyük Veri teknolojileri kullanılarak karşılaştıran sistemler internet kullanımı ile yaygınlaşmıştır. DongSheng Yin 2013 yılındaki çalışmasında SURF[6] ve ÖBÖD[7] algoritmalarını tercih etmiştir[8]. MapReduce programlama çatısı kullanarak Local Tetra Patterns (LTrPs) konularını Raju 2015 yılındaki çalışmasında kullanmıştır[9]. Youssef 2012 yılında yaptığı çalışmada yine resim işleme algoritmalarını Hadoop üzerinde kullanmıştır[10]. Hem Hadoop hem Lucene teknolojisini kullanan Chunhao 2012 yılında mimari üzerinde çalışmalar yapmıştır[11].

1.1.3 Resimlerin Öz Niteliklerin Çıkartılması

Benzer resimlerin bulunmasında karşılaşılan birçok problem vardır. Bu problemlerin başında ölçek değişikliği, açısal değişiklikler, dönüşümler ve gürültü olarak sıralayabiliriz.

Benzer resimlerin bulunmasında ilk olarak renk histogramları kullanılmıştır[12]. Histogramda her bir bin belli bir renk aralığının yoğunluğunu göstermektedir. Renk histogramı resmin döndürülmesi gibi dönüşümlerle değişmemektedir ve fakat kolaylıkla manipüle edilebilmektedir. Resmin kesilmesi ile değişen renk histogramlarının benzerliğini bulmak için bölgesel renk histogramı çıkarımı konusunda çalışmalar yapılmıştır.[13]

Yapılan bu çalışmalar göstermiştir ki resim özelliklerinin çıkartılmasında renk kendi başına yeterli olmamaktadır. Bunun için renk ile birlikte doku ve şekil bilgilerin hesaba katılması gerekmektedir. Bu bağlamda 2011 yılında Hazra tarafından Wavelet ve gri level doku eş-oluşumlu matrisler ile çalışmalar yapılmıştır[14].

Kenar ve köşe bilgileri resimlerin dokularını nitelemektedir. Bu nedenle anahtar nokta bulma algoritmaları kanar ve köşe bilgilerinin konumu ve yoğunlu ile ilgilenmiştir. Köşelerin belirlenmesi ilk olarak 1988 yılında Harris ve arkadaşları tarafından gerçekleştirilmiştir[15]. Kenar bilgisi farklı ölçeklerde değişebildiği için bu algoritma Mikolajczyk ve Schmid tarafından ölçeklemeye duyarlı hale getirilmiştir. [16]

Bu tür çalışmaların en ünlüsü David Lowe tarafından 2004 yılında sunulan Ölçekten Bağımsız Öznitelik Dönüşümü (ÖBÖD - SIFT) algoritmasıdır. [7]. ÖBÖD algoritması sağlamlığı ve açısal değişikliklere göre başarısı diğer algoritmaları geride bırakmıştır[17]. ÖBÖD tanımlayıcısının boyutunu 128 den 36 düşürerek işlemi hızlandırmaya çalışan Temel Bileşenler Analizi kullanılarak oluşturulan PCA-ÖBÖD [18] algoritması ÖBÖD algoritmasının yavaşlık problemini çözmeye çalışmıştır.

SURF[6] de yine ÖBÖD algoritmasından ilham almış ve Hessian Matriks ve Haar Wavelet sonuçlarını kullanarakanahtar noktaların bulunmasında ve bu noktaların tanımlayıcısının çıkarılmasında hız kazandırmıştır. ÖBÖD ve SURF algoritmaları işlem hacmi ve vektör büyüklü nedeni benzerlik işleminin bulunmasında yeterli başarıyı sağlasa da hız olarak tatmin etmemektedir. Bu nedenle benzerlik hesabı daha hızlı olan ve anahtar noktaların tespitinde yine ölçek ve dönüşümlere karşı dayanıklı olan ikili tanımlayıcılar konusunda çalışmalar yapılmıştır. BRISK[19], HRAN[20], BRIEF[21], ORB[22] bu ikili tanımlayıcı algoritmaların arasında en yaygın olarak kullanılanlardır. Butür ikili tanımlayıcıların temel özellikleri; anahtar nokta tespiti yapılıktan sonra, anahtar noktanın bulunduğu alanda belli bir örüntü kullanarak o alanın tanımlayıcısını çıkarmasıdır. Hesaplanması ve karşılaştırılması daha hızlı olan bu algoritmaların kullanımı son yıllarda artış göstermektedir. Resimlerin renk histogramını ve Haar Wavelet dönüşümü yardımı ile çıkartılan doku bilgisini birleştiren BRDH[23] algoritması, öz nitelik vektörün küçüklüğü ve daha kompakt olması nedeni ile dikkat çekmektedir.

1.2 Tezin Amacı

Bu tez kapsamında Büyük Veri teknolojileri kullanılarak içerik tabanlı resim arama motoru geliştirilmesi amaçlanmıştır. Arama motorlarında başarım oranının önemi kadar sonuca ulaşma zamanıda önemlidir. Başarıya hızlı ulaşmak için alakasız resimlerin hızlı bir şekilde elenmesi gerekmektedir. Başarımı yüksek fakat bir resim arama motorunda kullanılamayacak kadar yavaş olan ÖBÖD algoritmasının, alakasız resimlerin BRDH tekniği ile elenerek sayısı azaltılmış resim kümesinde kullanılması hedeflenmiştir.

1.3 Hipotez

Sürekli büyümeye açık resim kümelerinde arama yaparken başarı oranı kadar sonuçları elde etme zamanı da önemlidir. Bu tezde BRDH, HRAN, ÖBÖD ve Renk Haritalama algoritmalarınıkademeli arama mimarisinde kullanarak makul zaman içerisinde başarılı sonuç alınacağı hipotezi üzerinde çalışmalar yapılmıştır.

BÖLÜM 2

BÜYÜK VERİ

Bugün dünyada her gün büyük miktarda veri üretilmektedir. Geçmişten bugüne internet kullanımın değişimi veri üretimindeki en büyük etkendir. 2000'li yıllardan itibaren internetin yaşamımızın her alanında yer alması, sosyal medyanın yaygınlaşması, mobilite, algılayıcı teknolojileri bu kadar büyük miktarda verinin üretilmesini sağlayan temel etkenlerden bir kısmıdır.

Bugün hemen herkes sosyal medya ile iç içe bir yaşam sürdürmekte ve bununla birlikte iletişim biçimlerimiz ciddi bir dönüşüme uğramış durumdadır. Hatta iş hayatımız bile bu durumdan fazlasıyla etkilenmiştir. Örneğin, insanlar artık mobil telefonları ve bulut teknolojileri vasıtasıyla evden sürekli işlerini takip edebiliyor. Yine benzer şekilde insanlar iş yerlerinde arkadaşları ile sürekli iletişim halinde, çeşitli organizasyonlar yapabiliyor. Bir diğer ifadeyle; işimiz evimize taşınmış, evimiz ve arkadaşlıklarımız da işimize taşınmış durumda.

2011 yılı rakamlarına göre dünyada 2 milyar internet ve 4,6 milyar mobil telefon kullanıcısı bulunuyor. Bu kadar kullanıcının internette sürekli resim, video, doküman, e-posta vb. veri paylaştığını düşündüğümüzde karşımıza korkunç rakamlar çıkacaktır. Google'ın bir günde 20 Petabayt veriyi işlediğini, Facebook'a her gün 15 Terabayt veri girildiğini, Twitter'da bir günde 7 Terabayt veri üretildiğini düşündüğümüzde ve bunun her yıl artarak devam ettiğini düşündüğümüzde bunu ancak büyük ifadesi ile açıklayabiliriz. Günümüzde kurumlar artık petabayt ve ekzabayt ölçeğinden konuşmaktadırlar. Çoklu kaynaklardan üretilen veri miktarı devasa boyutlara ulaşmıştır ve uzmanlara göre her gün dünya üzerinde 2,5 ekzabayt (2,5 milyon terabayt) veri oluşturulmaktadır.

Bununla birlikte üretilen verinin büyüklüğü bu verinin depolanması, yönetilmesi, işlenmesi, bilgiye dönüştürülmesi ve anlamlandırılması noktasında birçok problem karşımıza çıkarmaktadır. Bu büyüklükteki veriler üzerinde geleneksel yöntemler yetersiz kalmakta, yeni yöntem ve teknolojilere ihtiyaç duyulmaktadır.

2.1 Büyük Veri Kavramı

"Büyük Veri" kavramını tanımlarken ilk olarak aklımıza çok fazla miktarda veri gelmektedir. Ancak bu kavram sadece büyüklük ile tanımlayabileceğimizden çok daha fazlasını ifade etmektedir. Bu açıdan büyük veri kavramı için bir tanım yapmak yerine veriyi ifade eden dört temel özellikten bahsetmek daha anlamlı olacaktır. Bunlara büyük verinin 4V'si denilir. İngilizce "Volume", "Velocity", "Variety", "Veracity" kelimelerini ifade etmektedir.

Hacim (Volume): Büyük Veri'yi ifade eden en önemli özelliktir ancak büyüklük kavramı tanımlanabilir olmakla beraber göreceli bir kavramdır. Örneğin bir metin verisi için 100 GB çok büyük bir kavramı ifade etse de video verileri için bu rakam çok da büyük değildir. Bu kavram daha çok petabayt ölçeğinden başlayan verileri ifade etmekle birlikte verinin işlenmesindeki zorluklar dolayısıyla daha küçük miktardaki veriler de bu kavram içerisinde yer alabilir.

Hız (Velocity): Bahsedilen büyük miktardaki verinin çok hızlı ve sürekli bir şekilde üretildiğini anlatır. Hatta dünyadaki mevcut verinin %90'ının son 2 yılda oluşturulduğu ifade edilmekte ve bu artan bir hızla devam etmektedir. Dünyanın her yerinde insanlar sürekli e-postalar atmakta, tweeter veya facebook üzerinde paylaşımlarda bulunmakta, uçaklar sürekli kalkış, iniş ve seyahatleri boyunca ürettikleri verileri göndermekte, milyonlarca insan bir yerden başka bir yere giderken konum bilgisi üretmektedir. Bu örneklerin sayısı arttırılabilir.

Çeşitlilik (Variety): Büyük verinin en önemli özelliklerinden biri de çeşitliliktir. Veriler video, resim, metin, sinyal vb. birçok farklı formatta üretilir.

Doğruluk (Veracity): En önemli özelliklerden biridir. Elimizde bulunan büyük miktardaki veri hatalı ya da eksik olabilir. Bu durum elde edeceğimiz bilgi ve sonuçları etkileyeceğinden dikkatle ele alınması gerekmektedir. Bahsedilen dört özellik büyük verinin yönetilmesini oldukça zor bir hale getirmektedir. İstenilen faydaların elde

edilebilmesi için hiçbir özellik ihmal edilemez. Ancak bu özelliklerin hepsini birden sağlayabilen sistemler bizim çözüm kümemizde yer alabilir.

2.2 Büyük Veri Tipleri

Büyük Veri kavramının tam olarak anlaşılabilmesi ve problemin boyutlarının görülebilmesi açısından somut bazı örnekler vermek gerekir.

CERN: Bilindiği gibi Cern Nükleer araştırmalar için oluşturulmuş bir organizasyondur ve büyük hadron çarpıştırıcısına ev sahipliği yapmaktadır. Burada 100 mega piksel kameralar ile proton çarpışmaları saniyede 40 milyon fotoğraf çekilerek kayda alınmaktadır. Bu da yaklaşık olarak her saniyede analiz edilecek 1 Petabayt veri demek. Bunun yanı sıra CERN yılda 35 Petabayt veri depolamayı ve bunları 20 yıl boyunca saklamayı planlanmaktadır.

Sosyal Medya: İnsanlarla ilgili toplanan verilerin %46'sı sosyal medya üzerinden toplanmaktadır. Tweeter'da her bir dakikada 100.000 tweet atılmaktadır. Facebook'da her bir dakikada 650.000 paylaşım yapılmaktadır. Bu, günde 144.000.000 tweet ve 936.000.000 facebook paylaşımı anlamına gelir. Bu yoğun kullanım sonucu sosyal medyanın ürettiği veri devasa boyutlara ulaşmıştır.

Mobil Cihazlar: Hayatımızın vazgeçilmezlerinden olan cep telefonları günümüzde birçoksensor ile donatılmış durumda ve sürekli internet bağlantısına sahiptir. Bu cihazlar devamlı olarak baz istasyonları ile haberleşmekte ve bulunduğu konum, yön, ses ve görüntü bilgilerini aktarmaktalar. 2013 rakamlarına göre sadece Türkiye'deki GSM abone sayısının 68 milyon olduğunu düşündüğümüzde üretilen verinin devasa boyutunu tahmin etmek zor olmayacaktır.

Nesnelerin İnterneti: Son zamanlarda adını daha sık duymaya başladığımız Nesnelerin İnterneti kavramı ilk kez 1999 yılında Kevin Ashton tarafından dile getirilmiştir. Nesnelerin İnterneti evinizdeki buzdolabından televizyona, bilgisayarınızdan alarm sisteminize günlük hayatta evde, işte, yolda yürürken kullandığınız her türlü cihazının internet üzerinden iletişime geçilebilir olması, bunları uzaktan yönetebilmeniz ve bu ağda yer alan cihazların bilgi paylaşımı ile akıllı bir ağ sistemi oluşturmasını ifade eder. Bugün internet üzerinde 11 milyar cihazın bulunduğu ve 2020 yılında bu rakamın 50 milyara ulaşacağı tahmin edilmektedir. Bu kadar cihazın ürettiği veriyi

düşündüğümüzde veri miktarlarının inanılmaz boyutlara ulaşacağını söylemek yanlış olmayacaktır.

Burada bahsedilen örnekleri çoğaltmak mümkündür, ancak temel olan veri miktarının ulaşacağı boyutların devasa nitelikte olmasıdır. 2020 yılında veri üretiminin günümüze göre 44 kat daha fazla olacağı öngörülmesinin temel nedeni de bu artışın hızla devam ediyor olmasıdır.

2.3 Dağıtık Mimaride Veri Saklama: Apache Hadoop

Apache HadoopDoug Cutting tarafından hazırlanmıştır. 2002 yılında başlamış bir içerik çekmek sistemi ve arama sistemi olan Apache Nutch, çekilen içeriğin saklanmasında ve aranmasında problemler çıktığı için dağıtık mimaride veri saklama ve işleme sistemlerine olan ihtiyacı ortaya koymuştur. Böylelikle Apache Hadoop ve Apache Solr projelerine başlanmıştır.

Apache Nutch projesi ilk zamanlarda web üzerindeki milyarlarca sayfayı yönetmesi mümkün görülmemiştir. Bu noktada, Google arama motorunun kullandığı dağıtık dosya sisteminin (Google File System, GFS) mimarisinin anlatıldığı 2003 tarihli yayın [24] ve Google'ın geliştirdiği dağıtık veri işleme modeli olan MapReduce programlama modelinin anlatıldığı 2004 tarihli yayından [25] esinlenilerek, Nutch için dağıtık bir dosya sistemi ve MapReduce gerçekleştirilmiştir.

Daha sonra veri işleme farklı alanlarda da kullanılabilir düşüncesi ile Hadoop adı altında bağımsız bir alt proje oluşturulmuştur. Doug Cutting'in Yahoo'ya katılmasıyla, Hadoop'un geniş ölçekte bir sisteme dönüştürülmesi için gerekli kaynaklar sağlanmıştır. 2008 yılında, Yahoo, arama indeksinin 10.000 çekirdekli bir Hadoop kümesi tarafından oluşturulduğunu duyurmuştur[26]. Hadoop, 2008 yılından beri üst seviye bir Apache projesi olarak yer almaktadır.

Hadoop'un büyük miktarda veri için dağıtık veri depolama ve işleme için sağladığı platform, tüm sektörde geniş bir ilgi görmüştür. Günümüzde, Oracle, Microsoft, HP gibi önde gelen pek çok bilişim firması, Hadoop desteği veya Hadoop ile entegrasyon çözümleri sunmaktadır. Bunun yanı sıra, birçok firmanın Hadoop dağıtımı vardır; Cloudera, Hortonworks, MapR gibi Hadoop firmalarının yanı sıra, IBM, Intel ve EMC gibi köklü firmalar da bunların arasında sayılabilir. Yahoo, Last.fm, Facebook, Twitter, New York Times, eBay, LinkedIn gibi pek çok şirket Hadoop kullanmaktadır [27 - 28].

Bunların arasında Facebook, Hadoop ve Hadoop ekosistemindeki ürünlerden Hive ve HBase'i veri tabanı ve gerçek zamanlı uygulamalar için kullanmaktadır. Twitter, Hadoop ve Hadoop ekosistemindeki Pig ve HBase'i veri analizi, görselleştirme, sosyal çizge analizi ve makine öğrenmesi gibi amaçlarla kullanmaktadır.

Hadoop, büyük verinin saklanması ve işlenmesi için maliyet etkin, ölçeklenebilir ve açık kaynak bir çözümdür. Hadoop, temelde iki ana kısımdan oluşur: HDFS (Hadoop Distributed File System) [29] ve MapReduce. HDFS, Hadoop'un dağıtık dosya sistemidir. Dağıtık olarak saklanmış dosyaları işlemek için Hadoop MapReduce ile gerekli programlama modelleri sağlanır. Hadoop mimarisi, verilerin bölümlenmesine ve büyük veri kümelerinin paralel bir şekilde işlenmesine izin verir. Hadoop ve ekosistemindeki ürünler için tercih edilen sunucular, genel olarak "orta sınıf" olarak adlandırılan sunuculardır. Veri depolama ve veri işleme yapısı sayesinde, bir Hadoop kümesi yeni sunucuların eklenmesiyle kolaylıkla ölçeklenebilir ve petabaytlarca verinin bulunduğu binlerce sunucuya erişilebilir.

2.4 Yapısallaştırılmamış Veri Tabanı: NoSQL

NoSQL, ilk olarak 1998 yılında Carlo Strozzi tarafından ortaya atılmıştır. NoSQL için "not only sql" tabiri kullanılmaktadır. Yani sadece sql değil denilmektedir. Bunun nedeni sql sorgularına alışmış geliştiriciler olarak aynı sorgu mantığının adapte edilmeye çalışılmasında yatmaktadır.

NoSQL, ilişkisel veritabanı sistemlerine alternatif bir çözüm olarak ortaya çıkan, yatay olarak ölçeklendirilen bir veri depolama sistemidir. İnternetin hızlanması ile beraber, sistemlerde çok fazla kullanıcının aktif rol alması, birçok ihtiyacın yanında veritabanı şemasının artan bir sıklıkla değiştirilmesi zorunluluğunu ortaya çıkardı. Hepimizin kullandığı ilişkisel veritabanlarındaki hiyerarşi önce tabloları ve her tabloya ait sütunları oluşturmak sonrasında ise bilgileri satır satır eklemektir. Fakat burada karşımıza çıkan bir sıkıntı tanımı olmayan alana bilgi kaydetmektir. Bu sıkıntıdan kurtulmak için yapmamız gereken öncelikle tabloyu tekrardan ele alıp yeni sütunlar eklemektir. Tabi bununla birlikte veritabanındaki tüm tablo ve ilişkileri etkileyecek durumlarda ortaya çıkabilmektedir. Bu işlemi gerçekleştirmek sistemi tekrardan ele almayı gerektirdiği için çok maliyetli olabilmektedir.

Özetle söylemek gerekirse İlişkisel Veri Tabanı Sistemlerinde maliyetin yükselmemesi adına yapıyı önceden çok iyi planlamak gerekmektedir. Fakat NoSQL veritabanları sistemlerinde bilgilerin kayıt altına alınması adına bu yapıyı sonradan kurmak çok daha az bir maliyet getirmektedir. Nedeni ise NoSQL sistemlerde tablo ve sütun kavramının olmamasıdır. NoSQL sistemlerde yeni bir alanaihtiyacınız olduğu durumlarda kaydı direk eklemeniz yeterli olmaktadır. Sistem sizin yerinize alanı oluşturur ve değeri kaydeder.

Tüm bu açıklamaların ardından "NoSQL veritabanları, ilişkisel veritabanlarından çok daha iyidir" gibi bir sonucu varmak yanlış olacaktır. Çünkü ikisi de yerine göre kullanılmalıdır. Örneğin: ilişkisel olayların çok yoğun gerçekleştiği bir bankacılık uygulamasında NoSQL bir veritabanı kullanmamız uygun değildir. NoSQL, Fire and Forget prensibi yani veriyi gönder gerisi ile uğraşma mantığı ile çalıştığı için bankacılık, alışveriş gibi para üzerinden işlem yapılan kritik uygulamalarda kullanılmamalıdır. Aksine verinin %100 önem arz etmediği durumlarda kullanıma daha uygundur.

NoSQL sistemlerinin ortaya çıkmasındaki en büyük sebep internetin hızlanması ile büyümekte olan ve yapısal olmayan verinin İlişikisel Veritabanı Yönetim Sistemlerinde (IVTYS) performanslı bir şekilde kaydedilememesidir. Veri büyüklüğü bir noktada bu veritabanlarının kısıtlarına takılmaktadırlar. Dikey büyüme maliyetli olduğundan, yatayda büyüyebilmek için daha fazla önbellek kullanımı, verilerin programatik olarak dağıtılması gibi teknikler denenmektedir. Bu uygulamaların başarısız olduğu durumlarda NoSQL sistemleri ortaya çıkmıştır.

Google, Big Table adını verdiği nosql veritabanı sistemi ile Amazon ise DynamoDB ismini verdiği nosql veritabanı sistemi kullanmaktadır. İnternetteki en büyük veri hacmine sahip olduğu bilinen Google firması, internet verilerini İlişkisel Veritabanları yerine BigTable'da tutmayı tercih etmektedir. Google firmasının bu tercihi yapma sebebi ise verilere ilişkisel veritabanı sistemlerine nazaran daha hızlı ve daha ucuza erişebiliyor olmasıdır. Daha ucuza erişebilmesini sağlayan önemli bir faktörde dikey büyüme yerine yatayda büyüme ile gereksiz bir ek maliyetten kurtulmalarıdır.

NoSQL veritabanlarının Avantajlarını saymak gerekirse;

- İlişkisel veritabanlarına göre yüksek erişilebilirlik imkânı sunarlar.

- Yatay olarak genişletilebilirler. Binlerce sunucu bir arada küme olarak çalışabilir ve çok büyük veri üzerinde işlem yapabilirler.
- Esnek yapılarından dolayı programlama ve bakım anlamında kolaylık sağlarlar.
- Çok fazla açık kaynak kodlu projelere ve bulut bilişim teknolojilerine uygun olduğu için maliyet olarak ilişkisel veritabanı yönetim sistemlerine göre daha avantajlıdır.

Her teknolojinin olduğu gibi NoSQL veritabanlarının da dezavantajları vardır. Bunlar;

- İlişkisel veritabanı yönetim sistemlerini kullanan uygulamaların NoSQL sistemlere taşınması başlangıçta zor olacaktır. Veriler başarılı bir şekilde taşınsa bile bağlantı (join) kullanan kodların değiştirilmesi gerekecektir.
- İlişkisel veritabanı yönetim sistemlerindeki sorgu tabanlı veri erişimi yerine NoSQL sistemlerdeki anahtar tabanlı veri erişimi sağlamak gerekmektedir. Buna göre bir yapılandırmaya gidilmesi zaman alabilmektedir.
- İlişkisel veritabanı yönetim sistemlerindeki işlem hareketleri kavramı, NoSQL veritabanı sistemlerinde bulunmadığı için veri kaybı söz konusu olabilmektedir.

NoSQL veritabanı sistemleri veri güvenliği konusunda ilişkisel veritabanı yönetim sistemleri kadar gelişmiş özelliklere henüz sahip değiller.

2.4.1 Dağıtık Mimaride Veri Tabanı: Apache HBase

Fikir ilk başta 2006 yılında San Fransisco'da bulunan ve doğal dil işleme projeleri geliştiren bir firma olan PowerSet firması tarafından ortaya atılmıştır. O zamanlar Google Big-Table makalesi yeni yayınlanmış ve PowerSet firmasındaki Chad Walter'in ilgisini çekmiştir ve şu ifadeleri kullanmıştır.

"Building an open source system to run on top of Hadoop's Distributed Filesystem (HDFS) in much the same way that BigTable ran on top of the Google File System seemed like a good approach because: 1) it was a proven scalable architecture; 2) we could leverage existing work on Hadoop's HDFS; and 3) we could both contribute to and get additional leverage from the growing Hadoop ecosystem."

2007 başlarında Apache Nutch projesinde Doug Cutting ile birlikte çalışan Mike Cafarela yazdığı java kodunu paylaşmıştır ve şu ifadeleri kullanmıştır.

“I’ve written some code for HBase, a BigTable-like file store. It’s not perfect, but it’s ready for other people to play with and examine.”

Apache HBASE[30] açık kaynak, ilişkisel olmayan Hadoop Distributed File System (HDFS) üzerinde çalışabilen java ile geliştirilmiş bir veri tabanıdır. Stun tabanlı ve hata tolerans özelliklerine sahiptir. Çok büyük veri kümesinde seyrek veri olarak adlandırılan istatistik gibi ufak bilgileri hızlı elde etmeye imkân sağlamaktadır.

HBase, Hadoop üzerinde gerçek zamanlı veri alma/yazma işlemleri için geliştirilmiştir. HBase ile milyonlarca satırın ve verinin kümeler yardımı ile sunulması sağlanmıştır.

Apache HBase bir NoSQL (Not only SQL) veri tabanıdır ve veri tabanından daha çok veri depolama sistemi olarak kullanılmaktadır.

HBase her ne kadar HDFS üzerinde çalışıyor olsada, HDFS ile aralarında farklılıklar vardır. Bunlar;

HDFS:

- HDFS Büyük ölçekli dosyaları saklamak için tasarlanmış, dağıtık mimariye sahip bir dosya sistemidir. Bu nedenle kaydedilen verilerde bireysel olarak arama yapılamamaktadır.
- Batch işlemleri gibi yüksek gecikmeye sahip işlemler için tasarlanmıştır.
- Verilere öncelikle MapReduce ile ulaşılabilmektedir.

HBASE:

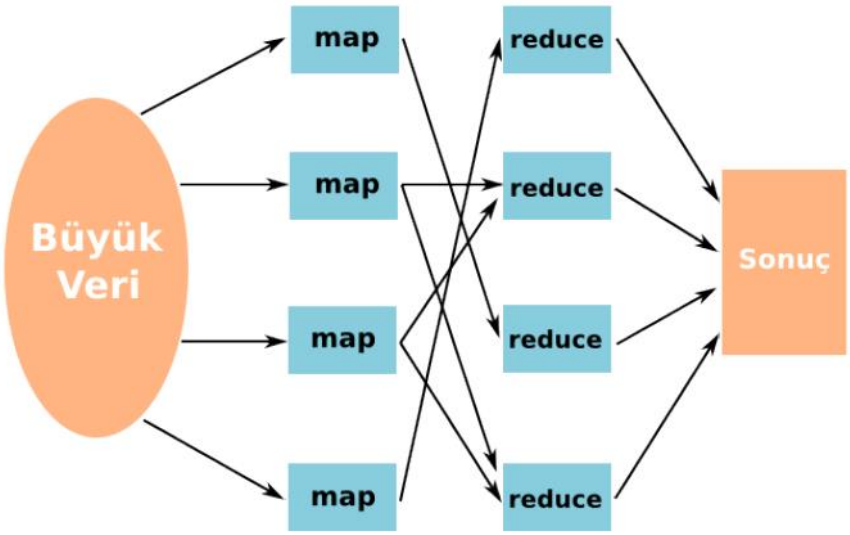
- HDFS üzerine kurulmaktadır ve büyük ölçekli kayıtlar arasında hızlı arama yapılmasını sağlar.
- Düşük gecikmeye sahip işlemler için tasarlanmıştır.
- Milyonlarca kayıt arasında tek bir satıra ulaşılmasına olanak sağlar.

2.4.2 Büyük Veride Veri İşleme: MapReduce Çatısı

MapReduce [31], verileri büyük kümelerde işleyebilmek için geliştirilmiş dağıtık programlama modelidir. Genel yapısı itibariyle MapReduce iki ana fonksiyondan oluşur. Birincisi Map; bir yığının tüm üyelerini sahip olduğu fonksiyon ile işleyip bir sonuç listesi döndürür. İkincisi Reduce; paralel bir şekilde çalışan iki veya daha fazla

Map'lerden dönen sonuçları harmanlar ve çözer. Map ve Reduce işlemleri paralel bir şekilde çalışırlar, aynı anda aynı sistemde olmaları gerekmez. Birçok programlama dili kullanarak (Java, C++, Python, Perl, Ruby, C vs.) MapReduce uygulaması geliştirilebilir. Google web sayfalarını indekslemek için MapReduce kullanmaktadır.

Map fonksiyonu çok büyük veri kümelerini bucket diye adlandırılan iki veya daha fazla kovaya böler. Sonrasında her bir kova belirli mantıkla ayrılmış kayıtları veya bir yazı satırı tutar. Her iş parçacığı (thread), işlemci veya sistem, her mantıksal kaydı işleyerek ara değer kümelerini hesaplamak için map fonksiyonunu ayrılmış kovalar üzerinde çalıştırır. Birden fazla kovada çalışmış olan bu fonksiyonların sonuçları benzer yapıda olmalı ki, sonunda bir araya getirildiğinde bir map fonksiyonu ile çalıştırılarak sonuç elde edilebilsin.



Şekil 2.1 Map Reduce çalışma mantığı

Devasa boyuttaki yapısal olmayan verinin işlenmesi çok zor bir konudur. Aynı zamanda çok hantal çalışan ve uzun süren programlar ortaya çıkmaktadır. MapReduce paralel sistemler üzerinde verinin işlenmesi ile alakalı çok başarılı çözümler sunmaktadır.

2.4.3 Web sayfaların içeriğinin çekilmesi

İnternette milyarlarca yazı, resim ve video bulunmaktadır ve bu bilgiler her an artmaktadır. Bu artış internetteki bilgilere kolay ve hızlı şekilde ulaşmak için etkili araçlara gereksinimi ortaya koymaktadır. Peki, nasıl oluyor da bu noktadan hareketle

hayatımıza giren arama motorları (örn. Google, Yahoo ve Yandex) bize ihtiyacımız olan bilgileri hızlı ve doğru şekilde bulabiliyor?

Arama motorlarının tamamında ortak adımlar vardır. Bu adımların başında içerik çekme yer almaktadır. İçerik çekmede işlemi internet sayfalarının taranması ve sayfa içeriğinin kaydedilmesidir. Ham içeriği kaydedilen sayfalar daha sonra analiz edilmekte ve sayfanın tüm bilgileri (başlık, içerik, diğer internet sayfalarına olan linkleri ve sayfanın hangi dilde yazıldığı) sayfa içeriğiyle beraber depolanmaktadır. Analiz edilen sayfalar hakkında nasıl bir analizden geçirildiği ve sayfa içeriğinin hangi sıklıkla tekrar çekildiği günümüz arama motorları arasındaki farklardandır.

Arama motorları mimarileri internetteki hızlı veri değişimini takip edebilmek için internet örümceği olarak isimlendirilen bir sistemler kullanmaktadır. Bu sistemler düzenli olacak şekilde, belirli kuralları kullanarak internetteki tüm sayfaları ziyaret etmektedir. Bilginin aranabilir olması için o bilgiye ihtiyaç duyulur mottosundan yola çıkarak, herhangi bir sorgu geldiğinde arama motorlarının kullanıcıya hızlı bir şekilde geri dönüş yapılabilmesi için bilgilerin önceden depolanması gerekmektedir. İnternet örümcekleri, kullanıcılar tarafından internet üzerindeki yazı, resim, video, müzik vb. her türlü bilgiye hızlıca erişilebilmesinde büyük rol almaktadır. 1994 yılında ilk internet örümceğinin ortaya çıkmasını takiben büyük başarı elde eden bu araç, geçen yıllar zarfında Lycos, Excite, Altavista, Google, Yandex, Bing gibi pek çok arama motoru tarafından kullanılmaya devam etmektedir.

Temelde internet sayfalarının içeriğinin çekilmesi için kullanılan internet örümcekleri birden fazla amaca dönük olarak da hizmet verebilmektedir. Örneğin, örümcekler belirli bir konuda (örn. çocuk kitapları ve sağlık yayınları, haber siteleri) internette yer alan kaynakların indeksinin oluşturulmasına yardımcı olmaktadır. Başka bir örnek olarak, internetteki içeriğin arşivlenmesini olası kılmaları ve böylelikle zaman içindeki değişimler hakkında analizlerin yapılmasına imkân sağlamaları verilebilir. İçeriği çekilecek kaynakların sınıflandırılabilmesi ve isteğe bağlı sınıfların içeriğinin çekilmesi de internet örümcekleri tarafından sağlanabilir. Örneğin, feed crawler olarak isimlendirilen internet örümcekleri sadece RSS/RDF tipindeki kaynakların içeriklerinin değişmeleri durumunda, sayfa içeriklerini çekmek için tasarlanmışlardır.

2.4.4 Büyük Hacimde Veri Elde Etme: Apache Nutch

Nutch açık kaynak bir arama motoru geliştirme mottosuyla başlayan, Apache Lucene [32] arama kütüphanesi temel alınarak geliştirilen ve Apache Software License ile dağıtılan bir arama sistemidir [33 - 35] Doug Cutting (Lucene ve Hadoop projelerinin sahibi) ve Mike Cafarella'nın ortak çalışması sonucunda geliştirilmiş olan Nutch'ın ilk sürümü Haziran 2003 tarihinde çıkarılmıştır ve Haziran 2005 tarihinden itibaren Lucene'nin yardımcı projelerinden biri olarak konumlandırılmıştır. Nutch, Hadoop mimarisi üzerinde dağıtık olarak çalıştırılabilmekte (MapReduce ile uyumlu dağıtık dosya sistemi) ve arama sistemleriyle rahatlıkla entegre edilebilmektedir.

Nutch, esnek yapısı ve ihtiyaca göre uyarlanabilirliği sayesinde, heterojen veri üzerinde işlem yapılabilmesini, arama arayüzünün isteğe bağlı olarak düzenlenebilmesini ve eklentiler sayesinde sürüm dışı yetkinliklerin kullanıcıya sunulabilmesini olası kılmaktadır.

Nutch, üç farklı senaryoda kullanım için düşünülebilir: Yerel dosya sistemi, İç İnternet (intranet) ve Evrensel İnternet. Her bir senaryo farklı özellikler taşıdığı için Nutch kullanımında farklı problemlerle karşılaşılabileceği öngörülmeli ve bu problemler için etkili çözümler üretilmelidir.

Nutch içeriği çekilecek olan web sitelerinin adreslerini istemektedir. Bu adreslere tohum denmektedir. Tohum kelimesi burada tam uymaktadır, çünkü verilen adreslerden yola çıkarak çekmekte olduğu sitelerde bulunan diğer sitelerin adreslerine dallanacaktır. Her bir dallanma olayına bir sonraki derinlik olarak adlandırılmaktadır. Eğer ilk verdiğimiz site adresleri içeri giren linkler olarak adlandıracak olursa, o sitelerde bulunan diğer sitelerin adresleri de dışarı çıkan linkler diyebiliriz. Böylelikle Nutch içerik çekme işlemini 3. derinlikte bile inanılmaz boyutlara ulaşmaktadır.

BÖLÜM 3

ARAMA VE İNDEKSLEME

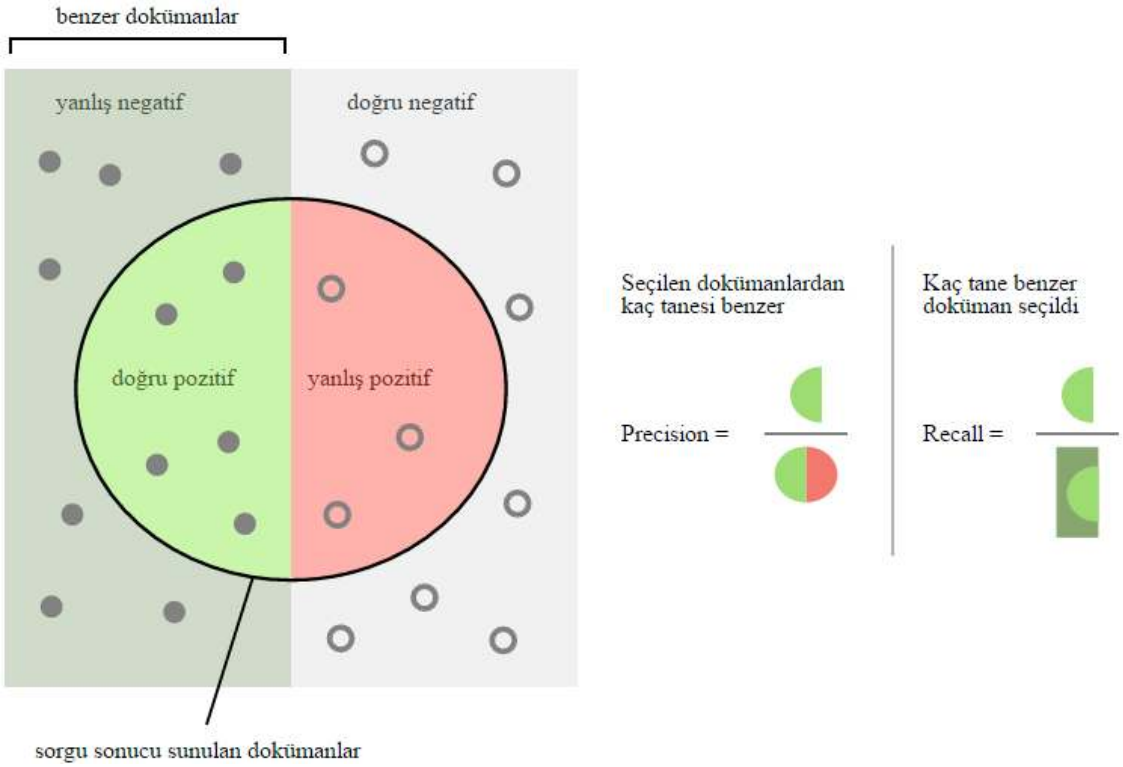
İnternet ve diğer kaynaklardaki veri miktarındaki hızlı artış ve bilgiye erişim araçlarının yaygınlaşması, bilgiye doğru, hızlı ve etkin erişimin önemini gün geçtikçe arttırmaktadır. Kullanıcıların bilgiye erişme isteklerini karşılamak üzere hazırlanan çözümler arasında, internet arama motorları, elektronik kütüphane sistemleri, kurumsal arama ve masaüstü arama gibi bilgiye erişim sistemleri sayılabilir. Arama, bilgiye erişim sistemlerinin temel görevi olmakla birlikte, bu alan, metin gruplandırma, metin sınıflandırma, özetleme, bilgi çıkarımı ve çoklu ortam bilgisine erişim gibi pek çok araştırma alanıyla da ilgilidir [36].

Bilgiye erişim sistemlerinde ele alınan veriler çoğunlukla metin türünde olmakla birlikte, bu konuda bir sınırlandırma yoktur. Görüntü, ses, video ve diğer veri türleri de bilgiye erişim sistemi içerisine dâhil edilebilir. Örneğin, Google arama motoru, metin araması dışında görsel arama gibi işlevler de sunmaktadır. Kullanıcılar, görselleri metinsel sorgularla arayabilmenin yanı sıra, mevcut bir görsel tanımlayarak benzer görseller arasında arama yapabilmektedir.

Bir bilgiye erişim sisteminin temel hedefi, kullanıcıya sorgusuyla ilgili mümkün olduğunca çok doküman sunulmasının yanı sıra, ilgili olmayan dokümanların da mümkün olduğunca az sunulmasıdır [37]. Bir arama motoru mimarisi tasarlanırken, kullanıcı sorgularına karşılık en ilgili dokümanların sunulması ile birlikte sorguların mümkün olduğunca hızlı bir şekilde işlenmesi de amaçlanır.

Bir arama motorunun sonuçlarının kalitesi değerlendirilirken, sıklıkla iki temel ölçüt kullanılır: Kesin isabet (*precision*) ve erişim isabeti (*recall*)[37]. Bu ölçütler tanımlanırken, kullanıcı sorgusuyla ilgili dokümanlar sistemde (İ) kümesi ile bu sorguya karşılık sunulan dokümanlar (S) kümesi ile gösterilebilir(Şekil 3.1). Kesin isabet, arama

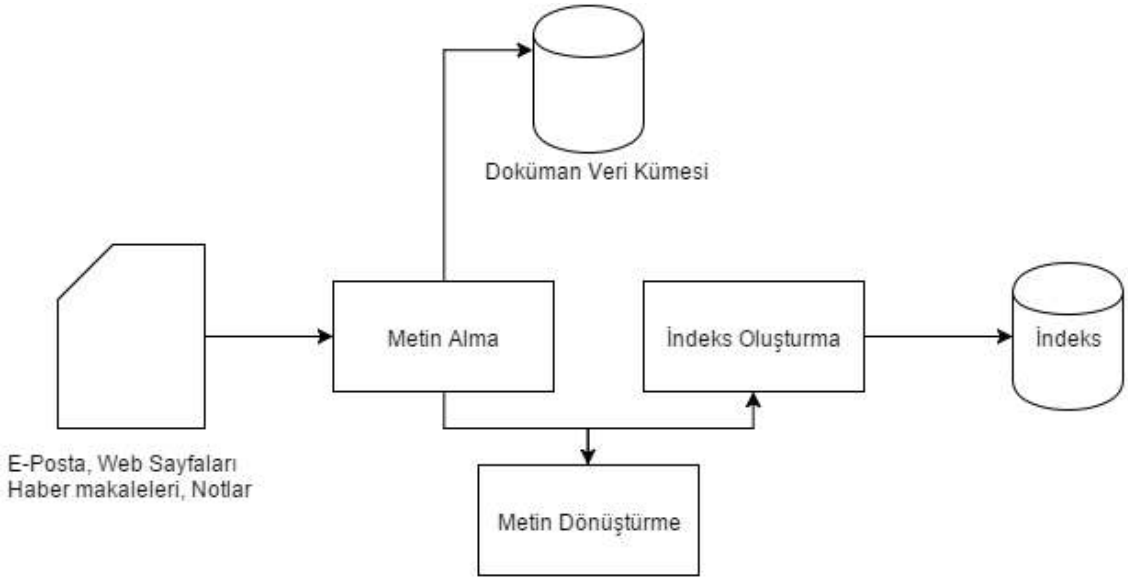
sonuçlarında sunulan dokümanlardan kullanıcı sorgusuyla ilgili olanların sayısının sunulan toplam doküman sayısına oranı (|İ ∩ S| / |S|) olarak tanımlanır. Erişim isabeti ise, kullanıcıya sunulan arama sonuçlarının, sistemde kullanıcı sorgusuyla ilgili belgeleri hangi oranda kapsadığına (|İ ∩ S| / |İ|) işaret eder.



Şekil 3.1 Precision / Recall gösterimi [50]

Bir arama motoru mimarisi indeksleme ve arama olmak üzere iki temel işlemi mevcuttur. Verilen doküman ya da girdi içeriği indekslenerek aranılabilir hale getirilir Daha sonra istenilen kriterlere göre bu indeks üzerinde arama yapılabilir.

İndeksleme aşamasının temel bileşenleri, metin alma, metin dönüştürme ve indeks oluşturmadır (Şekil 3.2) [38]. Metin alma bileşeni, dokümanların hangi kaynaktan alındığını gösterir. Metin dönüştürme, dokümanlardan indeks terimlerinin ve özelliklerinin elde edilmesi işlemidir. İndeks oluşturma bileşeni ise, dönüştürülen metni kullanarak hızlı aramayı sağlayacak indeks veya veri yapılarını oluşturur.



Şekil 3.2 Arama motorlarında indeksleme işlemi

3.1 İndeks ve Temel Arama Kavramları

Bir indeks, verinin tamamı ya da belirli bir kısmının sıralanmış ve düzenlenmiş halidir. İndeksler genelde bir yığın veriden aradığımız şeyi kolayca bulmamızı sağlar. Örnek olarak elimizde birkaç satırdan oluşan bir metinin bulunduğunu varsayalım:

Hayat bir nefestir, aldığın kadar,

Hayat bir kafestir, kaldığın kadar,

Hayat bir hevestir, daldığın kadar

Bu metine ait satırlara numara vermek bir indeksleme örneğidir.

Metin[1]: Hayat bir nefestir, aldığın kadar,

Metin[2]: Hayat bir kafestir, kaldığın kadar,

Metin[3]: Hayat bir hevestir, daldığın kadar,

Numaralanmış metinler artık daha kolay bulunabilecektir. Bir metinden bahsedilirken 3 numaralı metin denilerek tüm metin ifade edilmiş olacaktır. Metinlerin öz niteliklerine göre de indeksleme yapabiliriz. Aşağıdaki gibi her metine bir etiket verilebilir.

[Nefes]: Hayat bir nefestir, aldığın kadar,

[Kafes]: Hayat bir kafestir, kaldığın kadar,

[Heves]: Hayat bir hevestir, daldığın kadar

Birçok veritabanı yönetim sistemi benzer şekilde tablolardaki sütunları indekslememizi sağlamaktadır. Oluşturulan indeks her değer için tablodaki bir satır numarasına işaret eder. Bu sayede ilgili sütun ile yapılan aramalara tüm tablonun verileri gezilmeden hızlı yanıtlar dönülebilmektedir.

3.1.1 Evrilmiş İndeksler

Tam metin arama sistemlerinde Evrilmiş İndeksler kullanılır. Bunların temel çalışma mantığı kitap indeksleri ile aynıdır. Bu yöntemde, kitaplardaki dizinlere benzer biçimde, her bir indeks terimi için, terimin geçtiği dokümanlar, konumlar gibi ayrıntılı özellikler tutulabilir[18]. Örnek olarak indekslenmiş aşağıdaki gibi birçok metin olduğu düşünülürse;

hayat: 1,2,3

bir: 1,2,3

nefestir: 1

aldığın: 1

kadar: 1,2,3

kafestir: 2

kaldığın: 2

hevestir: 3

daldığın: 3

Evrilmiş indeks ile hangi kelimenin hangi metine ait olduğu kolaylıkla bulunabilmektedir. Örneğin "hevestir" kelimesi indeks üzerinde aranır ve 3numaralı metinde olduğu görülür. Aramalarda Boolean cebiri de kullanılabilmektedir. Örneğin aramak istenilen kelimeler "nefestir" VE "kadar" olduğunu düşünürsek. "nefestir" kelimesi 1 numaralı metinde geçerken "kadar" kelimesi 1, 2 ve 3 numaralı metinde geçmektedir. Kesişim kümesi VE işleminin sonucunu verecektir. Yani 1 numaralı

metinde hem "nefestir" hem de "kadar" kelimeleri bulunmaktadır. Benzer şekilde bir VEYA işlemi de birleşim kümesi alınarak bulunabilir.

3.1.2 Terim Sıklığı ve Ters Doküman Sıklığı

Evrilmiş indeksler bir kitap için düşünüldüğünde oldukça işe yarar yapılardır. Örneğin bir teknik terimi kitabın sonundaki indeksten aradığımızda bize hangi sayfalarda olduğunu numaraları ile bildirecektir. Evrilmiş indekste terim tekrar sayısı da kullanılabilir. Bu sayede milyonlarca sayfa ve milyonlarca doküman içinde bile aradığımız şeyi bulmak mümkün olacaktır. Aşağıda programlama dilleri ile alakalı dokümanlar indekslendikten sonra ortaya çıkan örnek bir evrilmiş indeks yapısı bulunmaktadır.

deri: 4:terim tekrarı(1000), 3:terim tekrarı(480), 2:terim tekrarı(90)

ayakkabı: 2:terim tekrarı(1200), 3:terim tekrarı(600)

topuk: 5:terim tekrarı(800), 1:terim tekrarı(100)

Yukarıdaki Evrilmiş İndeks'e göre "deri" konusunda bir arama yaptığımızda 4 numaralı dokümanın gelen önerilerde en üst sırada olmasını isteriz. Çünkü ilgili dokümanda 1000 adet "deri" ifadesi geçmiştir. Fakat bu yaklaşıma doküman büyüklükleri de eklenmesi gereklidir. 10 milyon kelimelik bir dokümanda 1000 defa "deri" ifadesi geçmesi ile 1000 kelimelik bir dokümanda 50 defa geçmesi gibi bir durumda daha az "deri" ifadesi geçen doküman daha önemlidir. Çünkü asıl olan doküman içinde aranan terime kaç terimde bir rastlandığıdır. Bunun için ilgili terimin dokümandaki Terim Sıklığı (Term Frequency) hesaplanmalıdır.

İkinci dokümanda terim tekrarı daha az olmasına karşın Terim Sıklığı (tf) daha yüksek olduğundan aradığımız terim için daha iyi bir sonuç önerisidir.

deri: 4:terim sıklığı(0.0050), 3:terim sıklığı(0.0001)

ayakkabı: 2:terim sıklığı(0.0120), 3:terim sıklığı(0.0010)

topuk: 5:terim sıklığı(0.0010), 1:terim sıklığı(0.0009)

ve: 1:terim sıklığı(0.0500), 5:terim sıklığı(0.0400), 3:terim sıklığı(0.0300), 4:terim sıklığı(0.0200),2:terim sıklığı(0.0100)

Yukarıdaki örnekte her terimin ilgili numaralı dokümandaki Terim Sıklığı (tf) değeri belirtilmiştir. "deri" ifadesi aratıldığında 4 numaralı doküman, en iyi sonuç önerisi olarak önerilecektir. Benzer şekilde "topuk" ifadesi aratıldığında ise 5 numaralı doküman en iyi sonuç önerisi olarak sunulacaktır. "deri ve ayakkabı" ifadesi aratıldığında ise öncelikle VEYA Boolean operatörü için tüm ifadelerin geçtiği dokümanlar çıkartılacak daha sonra Terim Sıklığı (tf) değerlerine göre sıralanacaktır.

deri: 4:terim sıklığı(0.0050), 3:terim sıklığı(0.0001)

ayakkabı: 2:terim sıklığı(0.0120), 3:terim sıklığı(0.0010)

ve: 1:terim sıklığı(0.0500), 5:terim sıklığı(0.0400), 3:terim sıklığı(0.0300), 4:terim sıklığı(0.0200),2:terim sıklığı(0.0100)

Öncelikle VEYA Boolean operatörü kapsamında kesişim kümesi alınarak elde edilen değer kümesi daha sonra Terim Sıklığına (tf) göre sıralanacaktır. En yüksek değere sahip dokümanın aranılan sonuca en yakın olması beklenir.

1: ve(0.0500) = 0.0500

5: ve(0.0400) = 0.0400

3: ve(0.0300) + deri(0.0001) + ayakkabı(0.0010) = 0.0301

2: ve(0.0100) + ayakkabı(0.0120) = 0.0220

4: ve(0.0200) + deri(0.0050) = 0.0205

Görüldüğü gibi "deri ve ayakkabı" ile ilgili yapılan aramada ilk iki sıradaki gelen sonuçlarda "deri" ve "ayakkabı" terimleri bulunmamaktadır. "ve", "veya" gibi dokümanlarda çok kez tekrar eden kelimeler sonuç sıralamasını olumsuz olarak etkilemektedir. Buradaki sorun için temel çözüm; doküman havuzundaki nadir kullanılan kelimelerin değerinin daha yüksek olması ve çok sayıda kullanılan kelimelerin değerinin daha az olmasıdır. Bunun için terimlerden dokümanlara evrilmiş bir bakış yapılarak her terimin doküman havuzunda kaç dokümanda geçtiği bulunur. Bulunan bu değere Ters Doküman Sıklığı (idf Inverse Document Frequency) denilir. Temel hesaplama metodu şu şekildedir:

log([Toplam Doküman Adeti] / [Terimin Geçtiği Doküman Adeti]) (3.1)

Yukarıdaki formüle göre örnekteki terimlerin Ters Doküman Sıklığı (idf) hesaplanabilir.

deri: log(5/2) = 0.3980

ayakkabı: log(5/2) = 0.3980

ve: log(5/5) = 0

Örnekteki arama sonuçları tf-idf'e göre tekrar hesaplanırsa;

2: ve(0.01 * 0) + ayakkabı(0.012 * 0.398) = 0.0048

4: ve(0.02 * 0) + deri(0.005 * 0.398) = 0.0020

3: ve(0.03 * 0) + deri(0.0001 * 0.398) + ayakkabı(0.001 * 0.398) = 0.0004

1: ve(0.05 * 0) = 0

5: ve(0.04 * 0) = 0

Sonuçların daha kabul edilebilir sınırlarda olduğu fark edilecektir. Günümüzde daha anlamlı sonuçlara ulaşma adına tf-idf benzeri birçok arama ve indeksleme algoritmaları bulunmaktadır.

BÖLÜM 4

RESİMLERİN ÖZ NİTELİKLERİNİN ÇIKARILMASI

4.1 Bulanık Renk Doku Histogramı (BRDH)

Bulanık Renk Doku Histogramı (BRDH - Fuzzy Color Texture Histogram FCTH)[23] algoritması renk histogramı ile doku histogramını tek bir histogramda birleştirerek toplamda 72 bytelık bir vektör ile resmi ifade edilmektedir.

4.1.1 Bulanık Mantık İle Renk Segmentasyonu

Bu aşamada resim renk bilgileri bulanık mantık kullanılarak segmente edilecektir [39]. Verilen resim ilk etapta maksimum 1600 olacak şekilde parçalara bölünür. Bölünen her bir parçanın içindeki piksellerin RGB değerlerinin ortalaması alınır. Daha sonra ortalaması alınan RGB değerleri HSV uzayına dönüştürülür ve HSV değerleri 10 bin ve 24 binlik iki adet bulanık mantık sistemi ile segmente işlemi yapılır.

4.1.1.1 10 bin Bulanık Mantık Histogramı

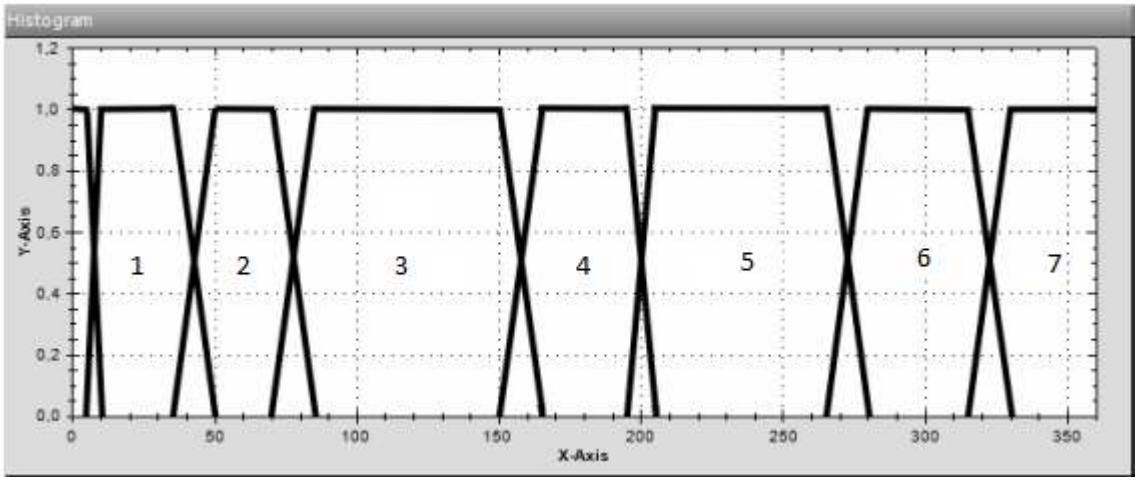
Hue kanalı 8 farklı bulanık alana bölünmektedir. Bunlar,

0. Kırmızı turuncu arası renkler
1. Turuncu
2. Sarı
3. Yeşil
4. Camgöbeği (Cyan)

5. Mavi
6. Eflatun (Magenta)
7. Mavi kırmızı arası renkler

Hue üyelik fonksiyonu

0. 0, 0, 5, 10 //Kırmızı turuncu arası
1. 5, 10, 35, 50 //Turuncu
2. 35, 50, 70, 85 //Sarı
3. 70, 85, 150, 165 //Yeşil
4. 150, 165, 195, 205 //Camgöbeği
5. 195, 205, 265, 280 //Mavi
6. 265, 280, 315, 330 //Eflatun
7. 315, 330, 360, 360 //Mavi kırmızı arası renkler

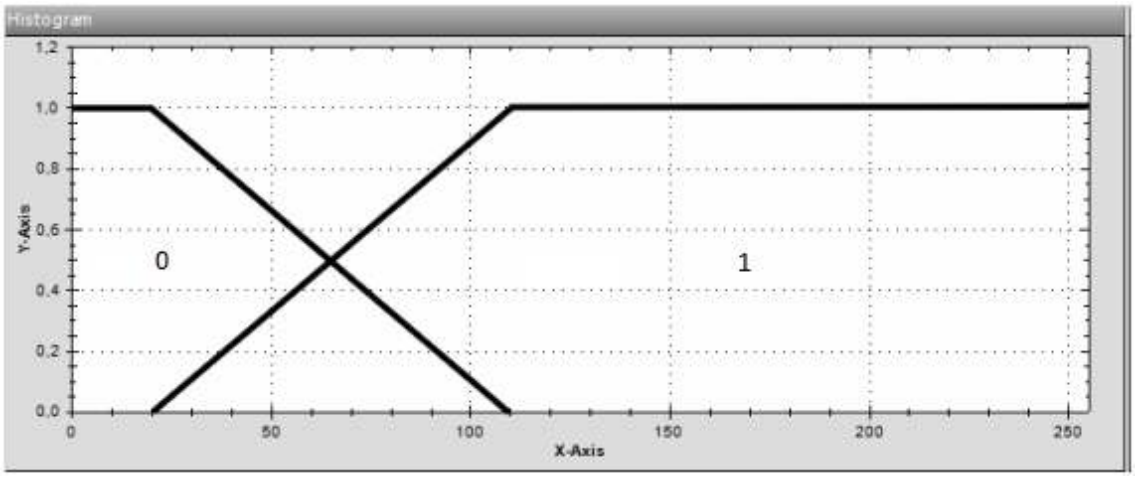


Şekil 4.1 Hue üyelik fonksiyonu

Saturation üyelik fonksiyonu

Saturation değerli aşağıdaki gibi 2 mantıksal alana bölünmektedir.

0. 0, 0, 10, 75
1. 10, 75, 255, 255

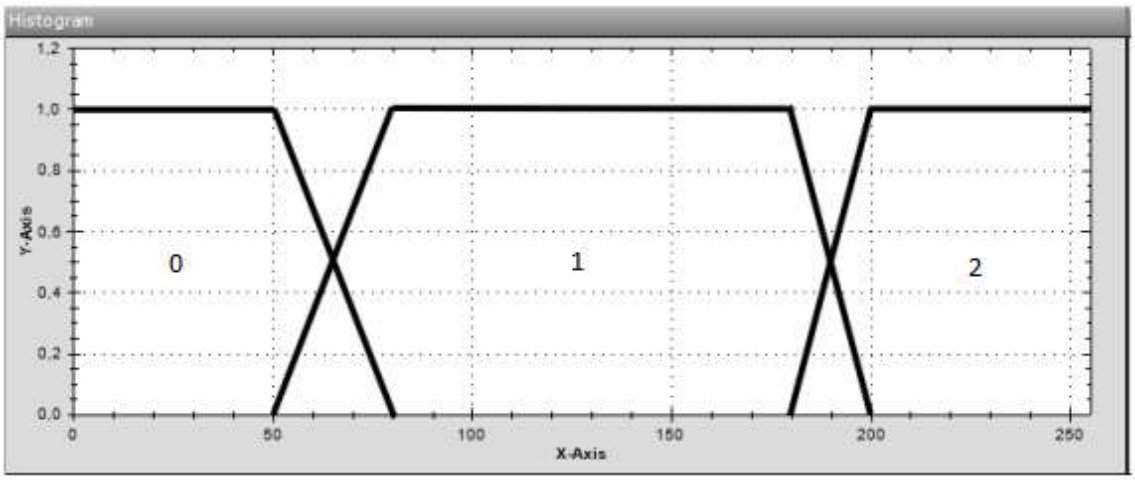


Şekil 4.2 Saturation üyelik fonksiyonu

Value üyelik fonsiyonu

Value değerli aşağıdaki gibi 3 mantıksal alana bölünmektedir.

0. 0, 0, 10, 75
1. 10, 75, 180, 220
2. 180, 220, 255, 255



Şekil 4.3 Value üyelik fonksiyonu

Belirlenen üyelik fonksiyonları ile 3 girişli (H,S,V) tek çıkışlı 10 binlik 48 bulanık mantık kuralları uygulanır [40]

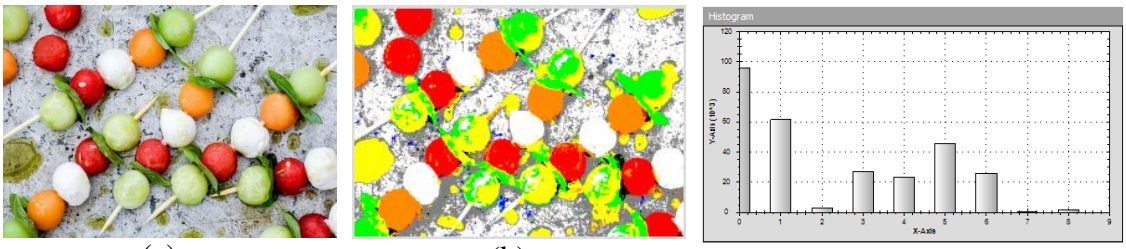
10 bin Bulanık Mantık kuralları

{0, 0, 0, 2}	{3, 0, 0, 2}	{6, 0, 0, 2}	{2, 1, 1, 5}
{0, 1, 0, 2}	{3, 1, 0, 2}	{6, 1, 0, 2}	{2, 1, 2, 5}

{0, 0, 2, 0}	{3, 0, 2, 0}	{6, 0, 2, 0}	{3, 1, 1, 6}
{0, 0, 1, 1}	{3, 0, 1, 1}	{6, 0, 1, 1}	{3, 1, 2, 6}
{1, 0, 0, 2}	{4, 0, 0, 2}	{7, 0, 0, 2}	{4, 1, 1, 7}
{1, 1, 0, 2}	{4, 1, 0, 2}	{7, 1, 0, 2}	{4, 1, 2, 7}
{1, 0, 2, 0}	{4, 0, 2, 0}	{7, 0, 2, 0}	{5, 1, 1, 8}
{1, 0, 1, 1}	{4, 0, 1, 1}	{7, 0, 1, 1}	{5, 1, 2, 8}
{2, 0, 0, 2}	{5, 0, 0, 2}	{0, 1, 1, 3}	{6, 1, 1, 9}
{2, 1, 0, 2}	{5, 1, 0, 2}	{0, 1, 2, 3}	{6, 1, 2, 9}
{2, 0, 1, 1}	{5, 0, 2, 0}	{1, 1, 1, 4}	{7, 1, 1, 3}
{2, 0, 2, 0}	{5, 0, 1, 1}	{1, 1, 2, 4}	{7, 1, 2, 3}

Bulanık mantık kurallarında bin'ler [41] deki çalışmadan faydalanılarak aşağıdaki gibi renkleri belirtmektedir.

0. Siyah
1. Gri
2. Beyaz
3. Kırmızı
4. Turuncu
5. Sarı
6. Yeşil
7. Camgöbeği
8. Mavi
9. Eflatun



(a) (b) (c)

Şekil 4.410 bin bulanık mantık sonucu a) orijinal resim b) bulanık mantık sonucu c) renk histogramı

4.1.1.2 24 bin Bulanık Mantık Histogramı

Elde edilen 10 binlik histogram ikinci bir bulanık mantık kurallarına tabi tutulmaktadır. Buradaki amaç elde edilen renklerin her birini 3 hue alanına genişletmektir. Örnek olarak sarı rengi: açık sarı, sarı ve koyu sarı renklerine genişletilecektir.

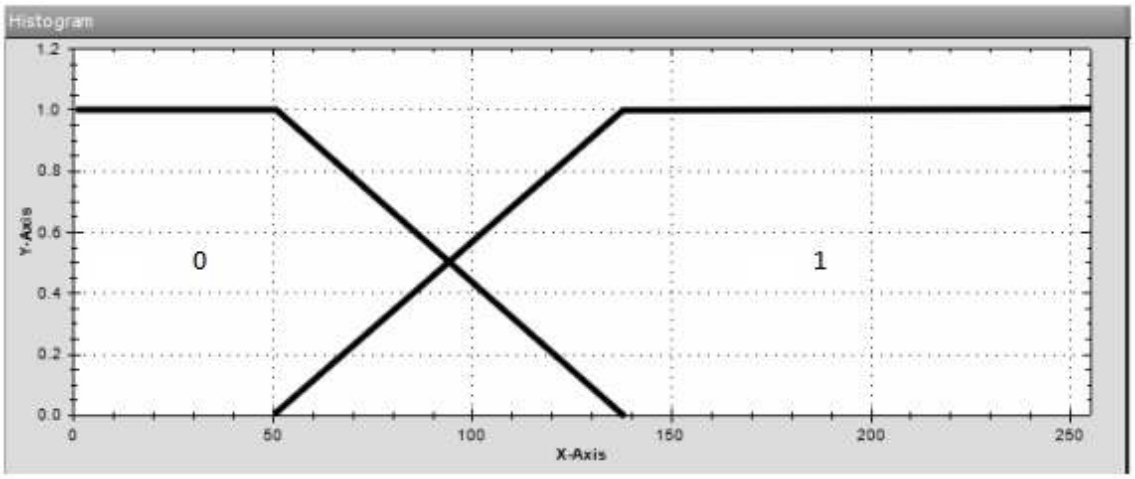
Uygulanan 2. bulanık mantık kuralları 2 girişli (S,V) ve tek çıkışlı 4 bulanık mantık kurallarından [40] oluşmaktadır.

Saturation üyelik fonksiyonu

0. 0, 0, 68, 188
1. 68, 188, 255, 255

Value üyelik fonksiyonu

0. 0, 0, 68, 188
1. 68, 188, 255, 255



Şekil 4.5 Saturation ve Value üyelik fonksiyonu

24 bin Bulanık mantık kuralları

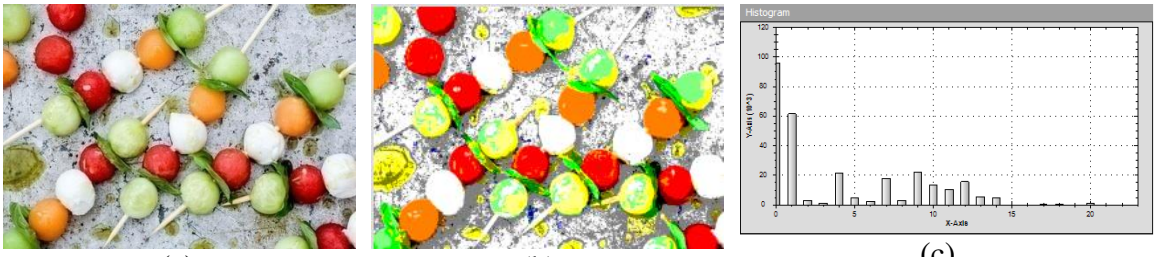
{1, 1, 1}, {0, 0, 2}, {0, 1, 0}, {1, 0, 2}

Bulanık mantık kurallarında bin'ler şunları ifade etmektedir.

0. Koyu renk
1. Rengin kendisi
2. Açık renk

10 binlik histogram 2. bulanık mantık kurallarına uygulanarak aşağıdaki 24 binlik histogram elde edilir. Burada siyah, gri ve beyaz renkler olduğu gibi aktarılmıştır.

0. Siyah	8. Açık Turuncu	16. Cam Göbeği
1. Gri	9. Koyu Sarı	17. Açık Cam Göbeği
2. Beyaz	10. Sarı	18. Koyu Mavi
3. Koyu Kırmızı	11. Açık Sarı	19. Mavi
4. Kırmızı	12. Koyu Yeşil	20. Açık Mavi
5. Açık Kırmızı	13. Yeşil	21. Koyu Eflatun
6. Koyu Turuncu	14. Açık Yeşil	22. Eflatun
7. Turuncu	15. Koyu Cam Göbeği	23. Açık Eflatun



(a) (b) (c)

Şekil 4.6 24 bin bulanık mantık sonucu a) orijinal resim b) bulanık mantık sonucu c) renk histogramı

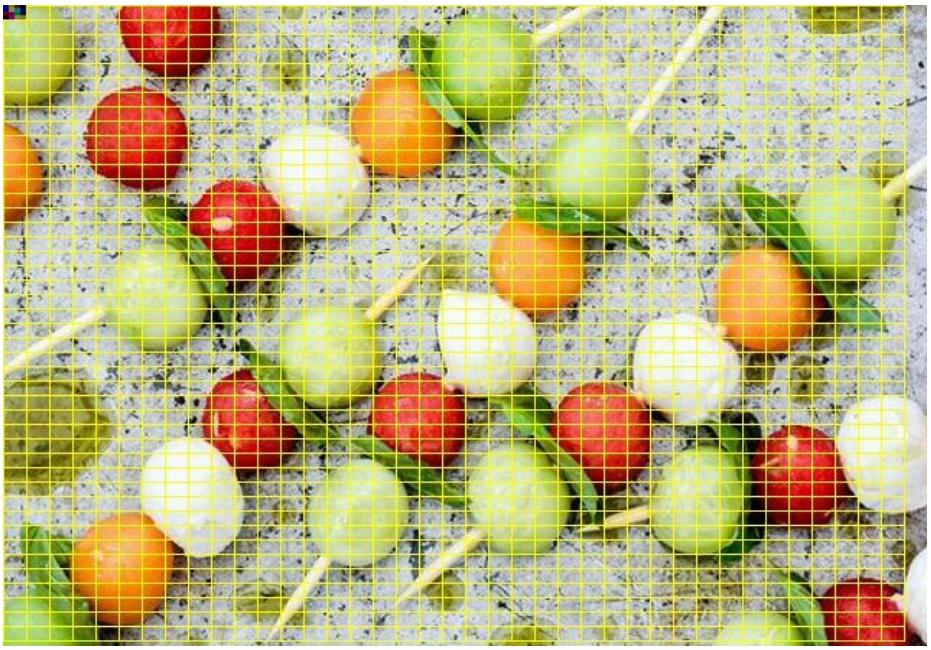
4.1.2 Bulanık Mantık Doku Segmentasyonu

Resmin doku bilgileri çıkartırken Wavelet Dönüşümü kullanılarak yüksek frekanslarda yatay, dikey ve diyagonal özellikleri hesaplanmıştır. Özelliklerin yüksek frekans bantlarından çıkarmanın nedeni doku özelliklerini yansıttığı içindir.

Resim ilk olarak RGB uzayından YIQ uzayına dönüştürülür ve burada Y değeri (Luminosity) kullanılır.

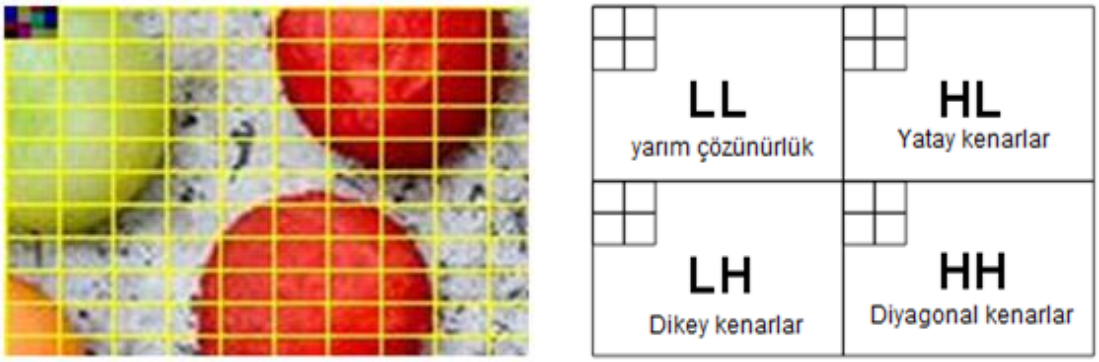
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} +0.299 & +0.587 & +0.114 \\ +0.595716 & -0.274453 & -0.321263 \\ +0.211456 & -0.522591 & +0.311135 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (4.1)$$

Şekil 4.7 YIQ renk uzayındaki Y değerleri



Şekil 4.8 1600 parçaya bölünmüş resim

1600 parçaya bölünmüş resmin her bir parçası 4x4 lük bloklara bölünür.



(a) (b)

Şekil 4.9 Örnek parça. a)orijinal resim kesiti b) frekans bantlarına ayrıştırılmış blok

Çıkarılacak özellikler (yatay, dikey ve diyagonal) Wavelet dönüşümün yüksek frekans bantlarındaki enerjilerdir. Bu, yüksek frekans bandında wavelet katsayıların ikinci seviye momentlerin karekköküdür[42]. Özellikleri çıkarmak için resmin YIQ uzayında Y değerine Haar Wavelet dönüşümü uygulanır. Böylelikle her bir blok 4 frekans bandına ayrıştırılır. (Şekil 4.9).

Her bir bant 2x2 katsayıya sahiptir, HL bandı katsayılar $\{C_{kl}, C_{k,l+1}, C_{k+1,l}, C_{k+1,l+1}\}$ise yatay özellik f_{HL}şu formül ile hesaplanır[42].

$$f_{HL} = \left(\frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} C_{k+i,l+j}^{2}\right)^{\frac{1}{2}} \quad (4.2)$$

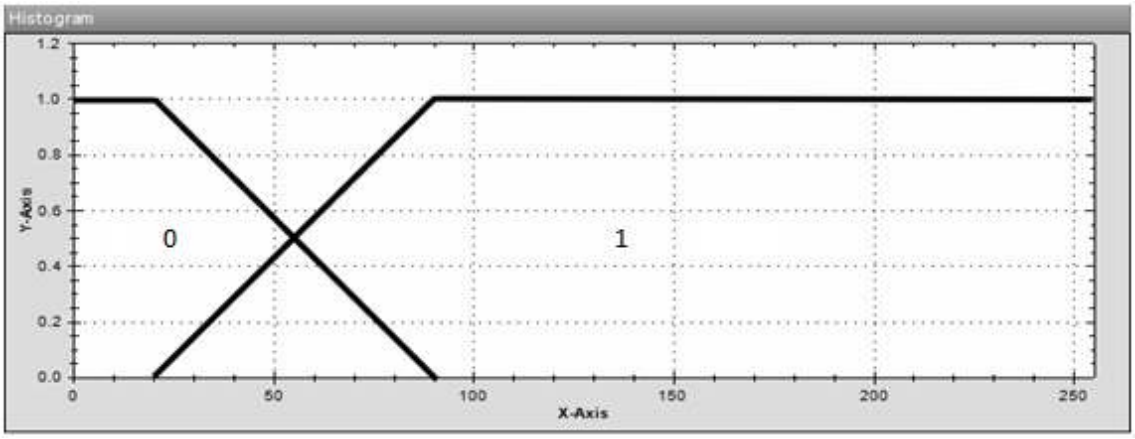
Diğer f_{LH} ve f_{HH} özellikleri LH ve HH bantları yardımı ile hesaplanır.

Daha sonraf_{HL}, f_{LH}, f_{HH} sırası ile yatay, dikey ve diyogonal özellikleri bulanık mantık sistemine girdi olarak kullanılır.

Uygulanan bulanık mantık kuralları 3 girişli f_{HL}, f_{LH}, f_{HH} ve tek cıkışlı 8 bulanık mantık kurallarından [40] oluşmaktadır.

f_{HL} ve f_{LH}üyelik fonksiyonu

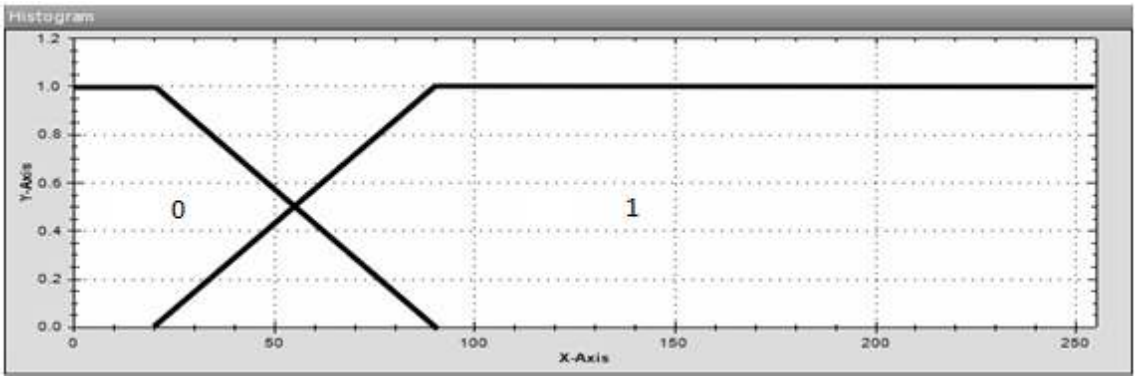
0. 0, 0, 20, 90
1. 20, 90, 255, 255



Şekil 4.10 f_{HL} ve f_{LH}üyelik fonksiyonu

f_{HH}üyelik fonksiyonu

0. 0, 0, 20, 80
1. 20, 80, 255, 255



Şekil 4.11 f_{HH} üyelik fonksiyonu

Bulanık mantık kuralları

{0, 0, 0, 0}	{1, 0, 0, 4}
{0, 0, 1, 1}	{1, 0, 1, 5}
{0, 1, 0, 2}	{1, 1, 0, 6}
{0, 1, 1, 3}	{1, 1, 1, 7}

Bulanık mantık kurallarında bin'ler şunları ifade etmektedir.

0. Düşük enerjili doğrusal alan
1. Düşük enerjili yatay alan etkinleştirme
2. Düşük enerjili dikey alan etkinleştirme
3. Düşük enerjili yatay ve dikey alan etkinleştirme
4. Yüksek enerjili doğrusal alan
5. Yüksek enerjili yatay alan etkinleştirme
6. Yüksek enerjili dikey alan etkinleştirme
7. Yüksek enerjili yatay ve dikey alan etkinleştirme

Sonuç olarak 8 binlik doku bilgisini barındıran bir histogram elde edilir.

Elde edilen 8 binlik histogram 24 binlik renk histogramı ile çarpılarak 192 binlik renk ve doku bilgilerini barındıran histogram elde edilir. Her bir parça 8 alana (3. bulanık mantık sonucu) ve her bir alan işe 24 alan ile (2. bulanık mantık sonucu) genişletilerek 192 binlik ($8x24 = 192$) histograma ulaşılmıştır.

Örnek:

1. bulanık mantık çıktısı: 3 (Kırmızı)

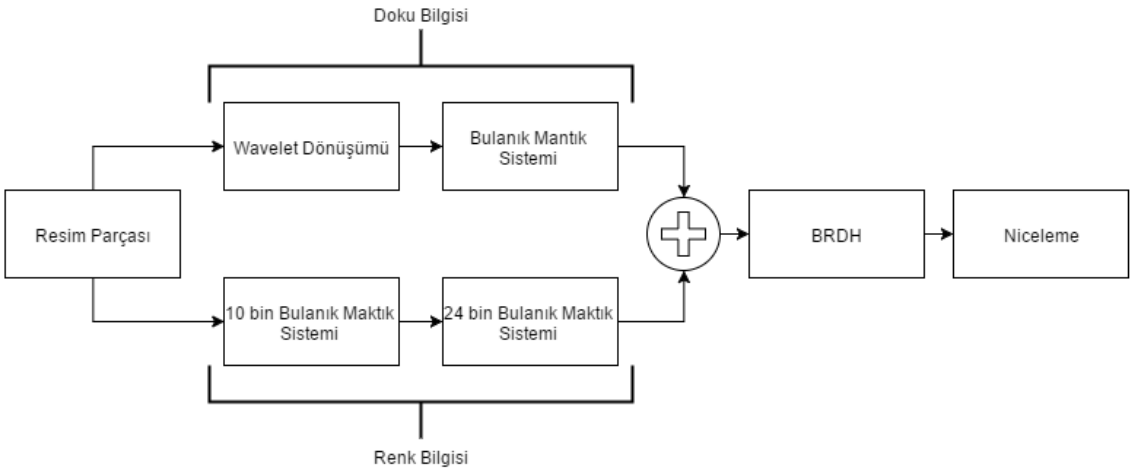
2. bulanık mantık çıktısı: 3 (Koyu Kırmızı)

3. bulanık mantık çıktısı: 1 (Düşük enerjili yatay alan etkinleştirme)

Bu durumda 3 bulanık mantık sisteminin birleşimi: $1x24 + 3 = 27$ değer 192 binlik histogramdaki 27. binin değerini vermektedir. 3^{fl} 3. bulanık sistemi ve 2^{fl} 2. bulanık sistemi ise;

$$3^{fl}x24 + 2^{fl} \tag{4.3}$$

Bulanık mantık doku segmentasyonu her bir parça için tekrarlanır ve son olarak {0,1} aralığında normalize edilir.



Şekil 4.12 Üç bulanık mantık sistemin birleştirilmesi

4.1.3 Niceleme

192 binlik histogramları niceleme işlemi için daha önce histogramları çıkartılmış 10000 resim kullanılmıştır. Bu histogramlar Gustafson Kessel Sınıflandırıcı [43] algoritması ile 8 alana sınıflandırılmıştır. Sınıflandırma sonucu aşağıdadır.

Çizelge 4.1 Gustafon Kessel normalizasyon tablosu (10^7ile çarpılmıştır)

Bin 0 - 23, 24-47							
13	931	2243	4312	8316	10143	17484	22448
Bin 48 - 71, 72 - 95, 96 - 119, 120 - 143							
23	1732	3911	6933	7912	9098	16179	18472

Çizelge 4.1 Gustafon Kessel normalizasyon tablosu (10^7ile çarpılmıştır) (devamı)

Bin 144 - 167, 168 - 192							
18	2373	4145	5391	6912	8198	9179	12472

Bu tablo bize şunu vermektedir. Örneğin: Hesaplanan histogramın{0 - 23} binleri arasındaki değerler tabloda hangi değere yakın ise o değerin indeksine {0 - 7} atanacaktır.

Sonuç olarak bu sınıflandırıcı binleri {0-7}.{0-1} (örn:7.025) olacak şekilde sınıflandırmaktadır. Bu da bize her bir resim için 192x3=576 bit büyüklüğüne sabitlenmiş bir değer vermektedir.

4.2 Hızlı Retina Anahtar Noktası (HRAN)

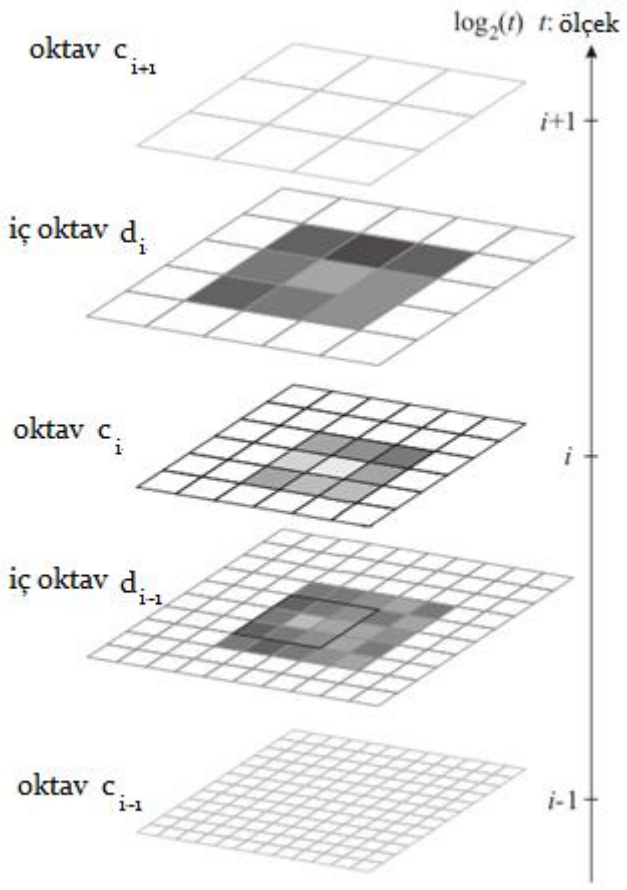
Hızlı Retina Anahtar Noktası (HRAN - Fast Retina Keypoint - FREAK) algoritması ikili öz nitelik tanımlayıcılı bir algoritmadır. 2012 yılında Alexandre Alahi[20] tarafından hazırlanmıştır. HRAN algoritması öznitelik çıkarma işlemini 4 temel adında yapmaktadır. Şöyle ki;

1. Anahtar nokta tespiti
2. Tespit edilen anahtar noktalara örüntünün uygulanması
3. Anahtar noktalar ile tanımlayıcının hesaplanması
4. Tanımlayıcılar kullanılarak arama işleminin yapılması

4.2.1 Anahtar Noktalarının Tespiti

Anahtar nokta tespitinde BRISK [19] algoritmasının anahtar nokta tespit metodu kullanılmıştır. Bu metot da, resimdeki anahtar noktaların tespiti AGAST [44] ve FAST [48] 9-16 filtresi ile yapılmaktadır. Ölçek uzay piramidin katmanları $i = \{0,1, \dots, n-1\}$ve n= 4 olmak üzere n adet oktav c_i, n adet iç oktav d_i den oluşmaktadır.Oktavlar kademeli olarak $c_0$olan orjinal resminin yarı yarıya ölçeklenerek oluşturulmaktadır.

Şekil 4.13'de görüldüğü gibi her bir $d_i$oktav $c_i ile c_{i+1}$ katmanlarının arasında konuşlanmaktadır.



Şekil 4.13 Ölçek uzay piramidi[14]

İlk iç oktav olan $d_0 c_0$ 'ın $\frac{1}{1,5}$ katı ölçeklenerek oluşturulmaktadır. Diğer iç oktavlar ise yarı yarıya kademeli şekilde küçültülerek ölçeklenmektedir. Bu nedenle t ölçekleme ise

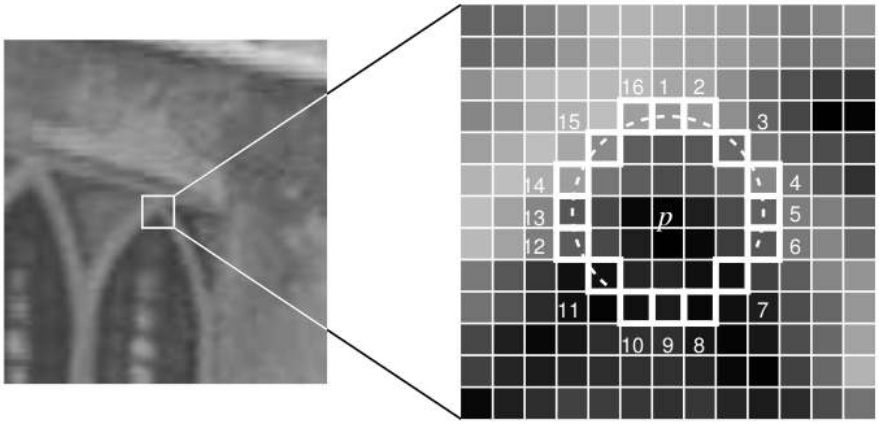
$$t(c_i) = \frac{1}{2^i} \text{ ve } t(d_i) = 1/(2_i x\ 1{,}5) \quad (4.4)$$

dir. N=4 için tüm oktavlar Çizelge 4.2'da verilmiştir.

Çizelge 4.2 HRAN oktavlar

Resim	Ölçek
c_0	1
d_0	$\frac{2}{3}$
c_1	$\frac{1}{2}$
d_1	$\frac{1}{3}$
c_2	$\frac{1}{4}$
d_2	$\frac{1}{6}$
c_3	$\frac{1}{8}$
d_3	$\frac{1}{12}$

FAST 9-16 köşe bulma filtresinde her 16 piksellik çemberde ardışık 9 pikseli koyu ya da açık ise merkez piksel anahtar nokta olarak işaretlenmektedir.



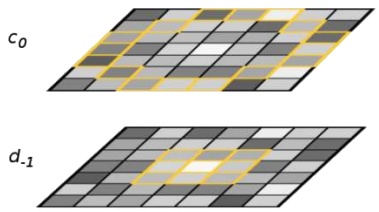
Şekil 4.14 FAST 9-16 filtre [17]

Potansiyel anahtar noktaları bulmak için her bir oktava ve iç oktava FAST 9-16 filtresi aynı T eşik değeri ile uygulanır.

FAST 9-16 filtresinde potansiyel anahtar nokta olarak işaretlenen alanların her bir pikseli için FAST[48] s değeri hesaplanır. Pikselin s değeri, kendisine komşu 8

pikselin(FAST 9-16 filtresinin içine sığan en büyük kare 3x3 alandır.) aynı katmandaki s değerinden, önceki ve sonraki katmanlardaki s değerlerinden büyük ise anahtar nokta olarak işaretlenir. Bu işlem tüm oktavlar için gerçekleştirilir.

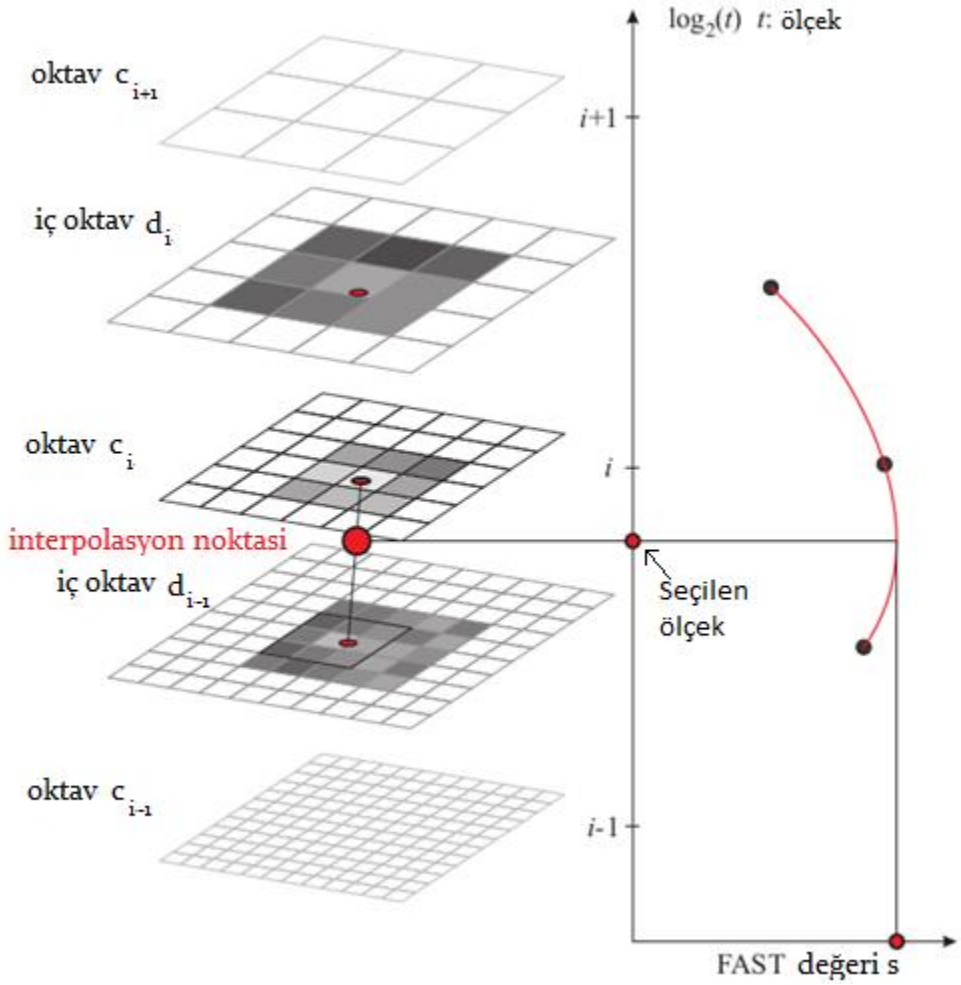
Burada c_0 birinci katman onduğu için $t(d_{-1}) = 0.75$ sanal ara katman direk olarak oluşturulmaz. Bunun için FAST 5-8 filtresic_0 katmanına aynı T eşik değeri ile uygulanır ve d_{-1} katmanı olarak kullanılır.



Şekil 4.15 C_0 katmanı

Daha sonra işaretlenen her bir anahtar noktalar için, bulunduğu katman ile önceki ve sonraki katmanlarda 3x3'lün alan iki boyutlu kuadratik fonksiyona fit edilerek maksimum değer bulunur.

Son olarak belirlenen maksimum değerler tek boyutlu kuadratik fonksiyona fit edilerek yerel maksimumlar seçilir. Bu maksimum değer anahtar nokta için ölçek olarak tanımlanır ve anahtar noktanın ölçekteki lokasyonu interpolasyon ile hesaplanır.(Şekil 4.16)

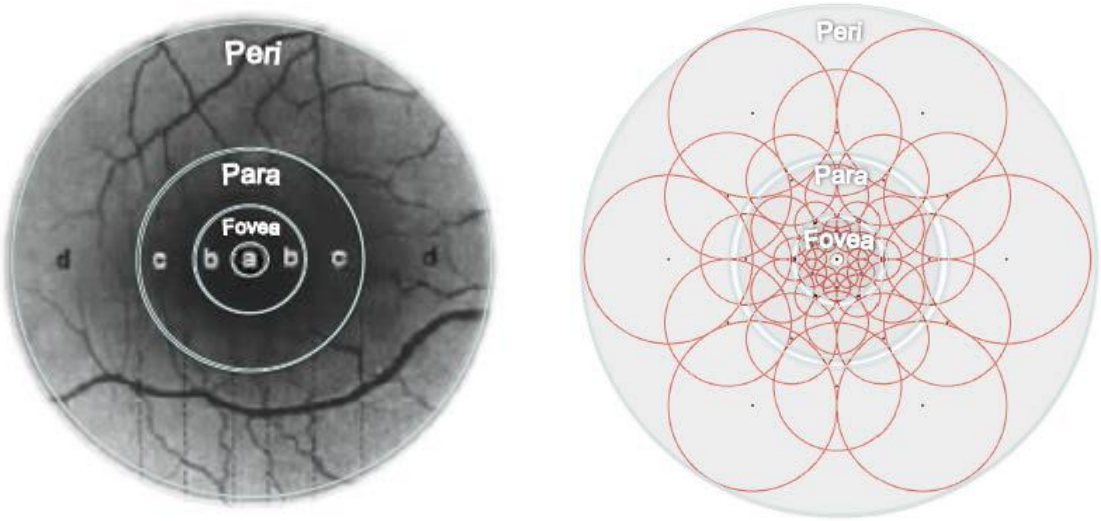


Şekil 4.16 Anahtar nokta için ölçek seçimi [15]

Yapılan bu işlem FAST [48] değerleri dikkate alınarak ölçek-uzay içerisinde maksimum noktaların tespitinde AGAST [44] algoritmasının çok boyutta uygulamasıdır.

4.2.2 Retina Örüntüsü Kullanımı

HRAN[20] insan gözünün retinasından esinlenmiş bir örüntüyü kullanmaktadır. Şekil 4.17'de de görüldüğü gibi merkezde çok yoğun fakat dışarı doğru hızla azalan dairelerden oluşmaktadır. Dairelere algılayıcı alan denmektedir.



(a) (b)

Şekil 4.17 Kullanılan HRAN örüntüsü a) insan retinası b)HRAN retina örüntüsü [20]

Literatürdeki bazı çalışmalar [21], [19], [22] göstermiştir ki, resim kesitlerin benzerliklerinin bulunmasında piksel ikililerin yoğunluk bilgilerinin karşılaştırılması ile yapılabilmektedir. Fakat piksel ikililerin belirlenmesinde ya da gürültüye dayanıklı hale getirilmesinde problemler yaşanmaya devam etmektedir. Nörobilimciler yaptıkları çalışmalarda ortaya koydukları sonuç, insan retinası resimlerin özelliklerini çıkarmada farklı boyutlarda Difference of Gaussian (DoG) kullandıklarıdır. HRAN algoritması da aynı strateji kullanarak resmin tanımlayıcısını oluşturmaktadır.

İlk olarak bir önceki maddede belirlenen her bir anahtar nokta, algılayıcı alanın yarıçapı($\sigma standartsapma$) ile orantılı olacak şekilde Gauss Çekirdeği ile yumuşatılır. Daha sonra yumuşatılan alanlar kullanılarak boşta algılayıcı alan kalmayacak şekilde ikililer oluşturulur. Bu ikililer ile sonraki madde de anlatıldığı gibi algılayıcı alanların yoğunluk farkları alınarak anahtar noktanın tanımlayıcısı elde edilecektir.

4.2.3 Anahtar Nokta Tanımlayıcısı

Anahtar noktaların tanımlayıcısı olan F, ikili dizi formatında tek bitlik gaussların farkının belli bir eşik değere uygulanmış halidir.

$$F = \sum_{0 \leq \alpha < N} 2^{\alpha} T(P_a) \tag{4.5}$$

Burada $P_a$algılayıcı alan ikililerini, N belirlenen tanımlayıcının uzunluğunu ve

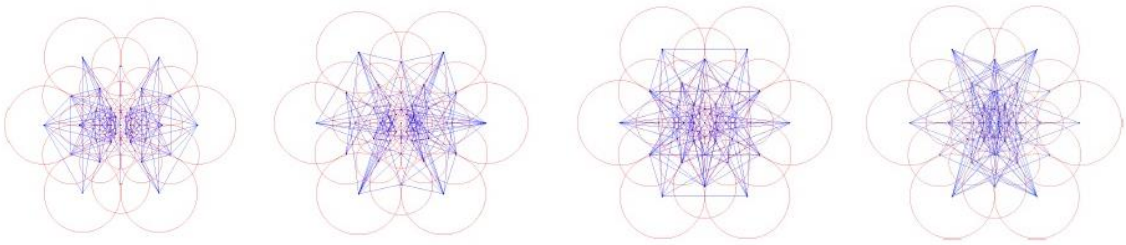
$$T(P_a) = \begin{cases} 1 & I(P^{r1}) - I(P^{r2}) > 0 \\ 0 & diğer durumlar \end{cases} \quad (4.6)$$

$I(P^{r1})$ ise $P_a$algılayıcı alan ikilisinin yumuşatılmış ilk algılayıcı alanı göstermektedir. Onlarca algılayıcı alan olduğu için yüzlerce ikilileri olacaktır. Fakat bu ikililerden birçoğu resmin karakteristiğini belirlemede yetersiz ya da gereksiz olacaktır. Bu ikililerin belirlenmesinde ORB'deki [22] kullanıma benzeyen bir algoritma geliştirilmiştir. Bu algoritma her bir ikililerin benzerliğini bulmak yerine birbiri ile ilintili olmayan ve en iyi sonucu veren ikilileri bulmaya yöneliktir. Algoritma şu şekildedir.

- Yaklaşık 500,000 anahtar noktayı barındıran bir matris D oluşturulur. Satırlar, her bir anahtar nokta için retina örüntüsünde bulunan tüm ikililer kullanılarak oluşturulan tanımlayıcıyı (F) içermektedir. Şekil 4.17'de gösterildiği gibi 43 adet algılayıcı alan ve yaklaşık 1000 adet ikili kullanılmıştır.
- Her bir sütunun ortalaması alınır. Burada birbirinden farklı özellikler elde etmek için yüksek varyant gereklidir. Burada 0,5 ortalaması olan en yüksek varyant vermektedir.
- Sütunlar varyant değerine göre büyükten küçüğe doğru sıralanır.
- Ortalaması 0,5 olan sütun seçilir ve seçilen sütun ile en düşük korelasyonu olanlar sırası ile alınır.

Bu algoritmada ilk seçilen ikililer örüntüdeki en dış halkaların örnek noktalarını, son seçilen ikililer ise örüntüdeki en iç halkaların örnek noktalarını karşılaştırmaktadır. Bu işlem insan retinasında da bu şekilde yapılmaktadır. İlk önce nesnenin yeri tespit edilir ve oraya odaklanılır, daha sonra algılayıcı alanların yoğun olduğu iç retina (fowea) ile bulunan nesne tasdik edilir.

Şekil 4.18'de her grupta 128 ikili olacak şekilde belirlenen ikililer 5 grupta toplanmıştır. Makalede ilk 512 ikilinin anahtar noktaların tanımlayıcısını çıkarmada yeterli olduğu, eklenen daha fazla ikilinin başarıyı etkilemediği belirtilmiştir [20]. Belirlenen 512 ikili ile F ikili dizisi hesaplanarak resmin anahtar noktalardaki tanımlayıcısı çıkartılır.



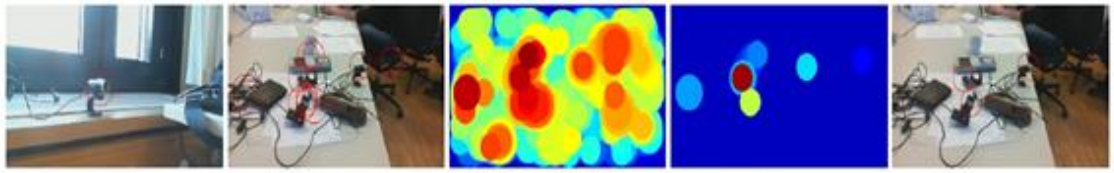
(a) (b) (c) (d)

Şekil 4.18 Anahtar nokta ikililerıa) ilk 128b) 128 -256 arasıc) 256 - 384 arasıd) 384 - 512 arası[20]

4.2.4 Tanımlayıcılar Arasında Arama İşlemi

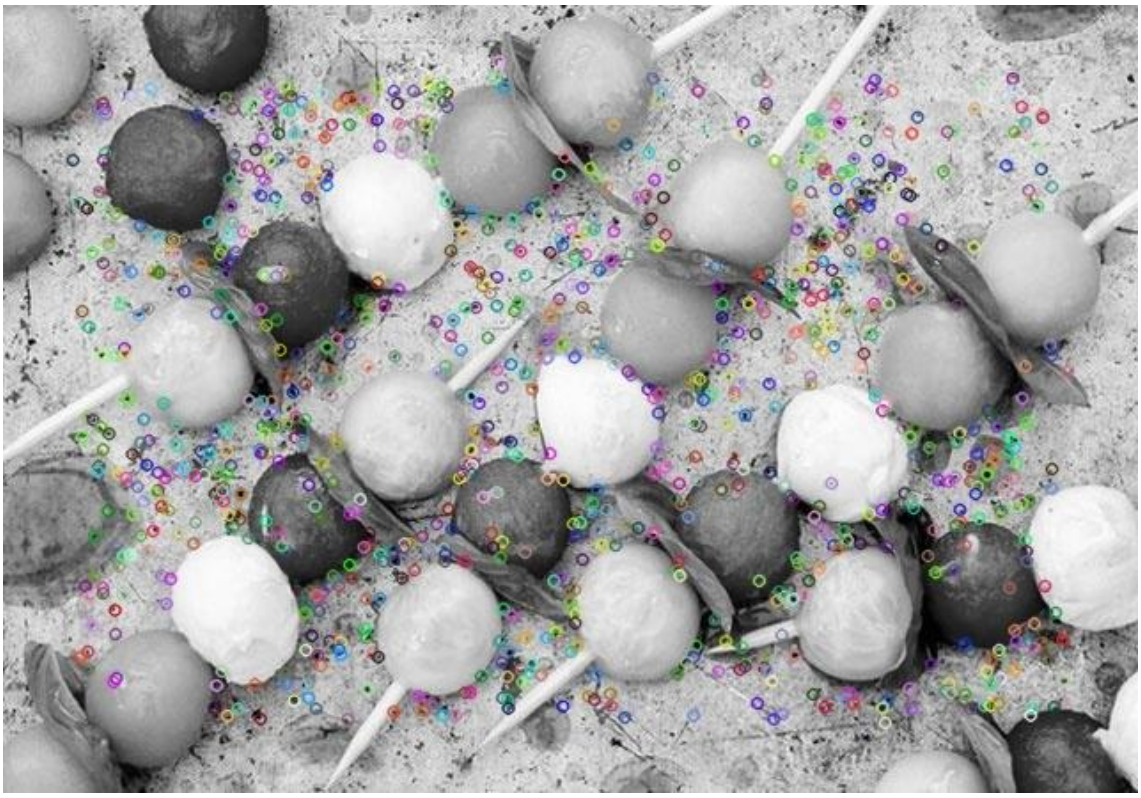
İnsan alana bakarken sabit bir davranış sergilemez. Gözler etrafa kesintili ve birbirinden bağımsız hareketler yapmaktadır. Bu hareketlere Saccade denmektedir. En iç alan olan Fovea nesnelerin tanınması ve benzerliğin bulunması için yüksek çözünürlükte bilgiler toplarken, en dış alan olan Perifoveal nesneleri tespit etmek için düşük çözünürlükte bilgiler toplamaktadır.

HRAN ile çıkartılan tanımlayıcılar arasında arama işlemi yapılırken retinanın bu işlemi taklit edilmektedir. İki tanımlayıcı karşılaştırılırken ilk 128 bitleri karşılaştırılır, eğer uzaklık belli bir eşik değerin altında ise sonraki 128 bit karşılaştırılmaya devam edilir. Belirtilen kademeli karşılaştırma ile ilk 128 bit karşılaştırmada aday noktalar %90 oranında elenmektedir.[20].Karşılaştırma işlemi hamming uzaklıklarının toplamı ile yapılmaktadır.



(a) (b) (c) (d) (e)

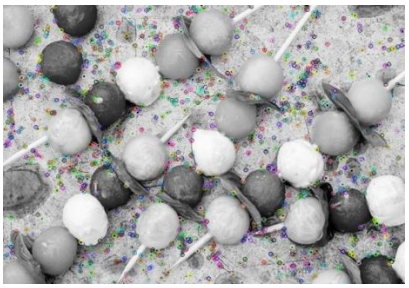
Şekil 4.19 Arama işleminin görselleştirilmesi a) aranılacak nesne b) 1. kademede en iyi sonuçlar c) 1. kademedeki sonuçların uzaklık haritası d) son kademeden sonraki arama sonuçları [20].



Şekil 4.20 HRAN anahtar özellikleri çıkartılmış örnek resim

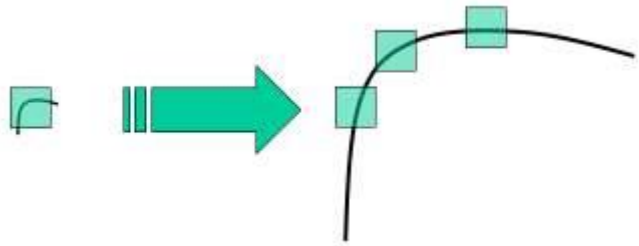
4.3 Ölçekten Bağımsız Öznitelik Dönüşümü (ÖBÖD)

Son yıllarda literatürde kenar hesaplama işlemini başarılı bir şekilde yapan birçok algoritma geliştirildi. Bunlardan birisi de Harris algoritmasıdır. Buna Şekil 4.21 ile örnek verebiliriz.



Şekil 4.21 Harris anahtar özellikleri çıkartılmış örnek resim

Bu algoritmalar kenar hesaplamalarını resmin dönmesinden etkilenmeden yapmaktadır. Çünkü resim döndüğü zaman kenar da aynı oranda dönecek ve kenar yine aynı kenar olarak kalarak hesaplanmasında problem oluşmayacaktır. Peki ya farklı ölçeklerdeki resimlerde durum aynı mı?



Şekil 4.22 Aynı pencere boyutunda ölçeklendirilmiş kenar bilgisi.

Kenar ölçeklendirilmiş resimde hala bir kenar olmayabilir. Şekil 4.22'de küçük pencere içindeki köşe, resim yaklaştırılınca aynı pencere içinde düzleştiği görülmektedir. Bu nedenle Harris algoritması ölçekten bağımsız değildir.

2004 yılında David Lowe [7] tarafından geliştirilen Ölçekten Bağımsız Öznitelik Dönüşümü (ÖBÖD - Scale Invariant Feature Transform - SIFT) algoritması, resimlerin boyutu, resimlerin aldığı kamera bakış açısı gibi durumlardan etkilenmeden resimlerin özelliklerinin çıkarılmasına olanak sağlamaktadır.

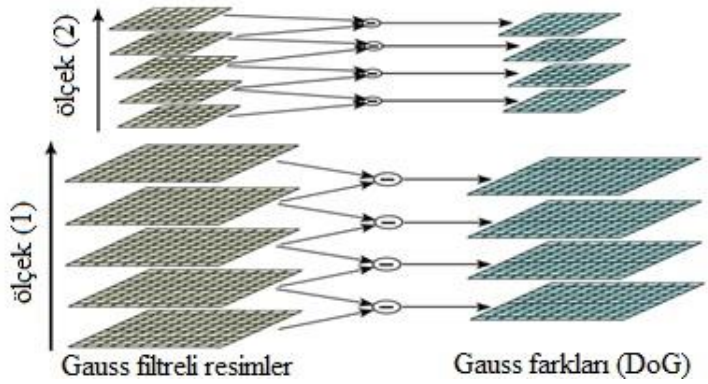
ÖBÖD algoritması temel olarak 4 adımdan oluşmaktadır.

- Ölçek uzayda uç noktaların hesaplanması. Bu aşamada minimum maksimum değerler hesaplanır.
- Anahtar noktaların belirlenmesi.
- Anahtar Noktaların döngüsel değişime uyum sağlaması.
- Anahtar Nokta tanımlayıcısının hesaplanması.

4.3.1 Ölçek-Uzay'da Üst Noktaların Belirlenmesi

Şekil 4.22'de gösterilen pencere boyutu farklı ölçeklerde anahtar nokta bulunmasında kullanılamayacaktır. Ölçek uzayda değişmeyen anahtar noktaların bulunması için DoG (Difference of Gaussian, Gauss Farkı) metodu kullanılmıştır. Bunun için ölçek uzay filtresi kullanılmıştır. Bu filtrede LoG (Laplacian of Gaussian) farklı σ değerlerin yardımı ile noktalar bulunmaktadır. Bir diğer ifade ile düşük σ değeri küçük köşeler için küçük ve yüksek σ değeri büyük köşeler için yüksek değerler verecektir. Böylelikle ölçek uzay alanında yerel maksimumların listesi (x, y, σ) elde edilir. Bu da bize muhtemel anahtar noktaların x, y korninatlarını vermektedir.

LoG işleminin maliyeti oldukça büyük olduğu için ÖBÖD algoritması LoG 'un yakınsamasını sağlayan DoG (Difference of Gaussians) kullanmaktadır.



Şekil 4.23 DoG - Gauss Piramidi [7]

L yumuşatılmış resim, I orijinal resim ve G değişken ölçek oranına sahip Gauss fonksiyonu ise, ölçek uzay fonksiyonu şu şekilde tanımlanır:

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y, k\sigma) \tag{4.7}$$

G ile gösterilen Gauss fonksiyonunun ise,

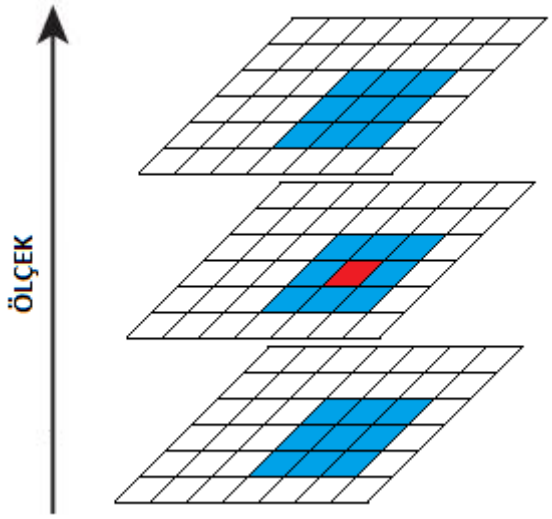
$$G(x, y, \sigma) = \frac{1}{2\pi^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{4.8}$$

Şeklinde tanımlanır.

DoG fonksiyonu olan *D(x, y, σ),*farklı standart sapmaya sahip Gauss filtrelerinden geçirilen ve k kadar ölçeklenmiş resimler arasındaki farkın sonucudur. Bu işlem Şekil 4.23 de gösterildiği üzere Gauss Piramidinde resim farklı ölçeklerde yumuşatılarak tekrarlanır.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{4.9}$$

DoG işleminden sonra ölçek uzay üzerinde yerel maksimum noktalar aranır. Bu işlem ilgili ölçekteki pikselin 8 komşusu ve bir önceki ve bir sonraki ölçeklerdeki aynı iz düşümündeki piksellerin karşılaştırılması ile yapılır. Yani toplamda 26 komşusu ile karşılaştırılır. Eğer Şekil 4.24'deki kırmızı nokta yerel maksimum ise o ölçekteki anahtar nokta, en iyi kırmızı nokta ile temsil edildiğini göstermektedir.

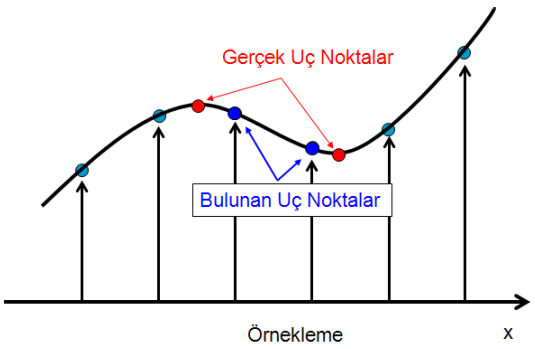


Şekil 4.24 Yerel maksimumların bulunması

4.3.2 Anahtar noktaların belirlenmesi

Aday anahtar bir önceki başlıkta belirlenmişti, bu başlıkta ise noktalar tekrar bir işlemden geçirilerek rafine edilecektir. Böylelikle yanlış olan anahtar noktalar elenecektir.

Önceki başlıkta belirlenen anahtar noktalarıyerel uç nokta olma ihtimalinden dolayı adayanahtar nokta olarak tanımlamaktayız. Maksimum ya da minimum hesabı piksel üzerinde doğru cevabı verecektir fakat pikseller arasında maksimum ya da minimum hesabı Şekil 4.25'de görüldüğü gibi bizi yanılgıya uğratabilir. Burada kırmızı nokta asıl uç nokta fakat mavi noktalar ise önceki başlıkta bulduğumuz anahtar noktalar.



Şekil 4.25 Gerçek ve hatalı uç noktalar

Bu nedenle yardımcı uç noktaların lokasyonları belirlenmesi gerekmektedir. Bu belirleme işleminde aşağıda belirtilen Taylor Serisi açılarak kullanılmaktadır. Uç noktaların lokasyonları,

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \quad (4.10)$$

ile hesaplandıktan sonra

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \quad (4.11)$$

elde edilir. DoG fonksiyonu olan D*(x,y,σ)* ve türevi, her uç nokta için hesaplanır ve |D(x)| < 0.03 ise elenir. Böylelikle DoG resminde düşük kontrastlı noktaları elemiş oluruz.

Daha sonra değersiz kenarları eleme işlemi yapılmaktadır. Burada belirlenen anahtar noktalarda birbirine dik 2 eğim oluşturulur. Bu durumda nokta etrafındaki resmin 3 olasılığı vardır. Bunlar;

Düz alan: 2 eğimde düşük ise bu durum oluşur

Kenar: Eğimlerden biri büyük (kenara dik olma durumu) diğeri ise küçük (kenar üzerinde olma durumu) ise bu durum oluşur

Köşe: 2 eğimde büyük ise bu durum oluşur.

Burada sadece köşeleri almak istediğimiz için diğerleri elenmektedir. Bu işlem Hessian matrisi ile elenir. *D*, DoG fonksiyonu olmak üzere, Hessian matrisi *H* ile gösterilir.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \quad (4.12)$$

a, büyük olan öz değeri, *b*, küçük olan öz değeri göstermek ve *r = a/b* olmak üzere, ÖBÖD kriteri,

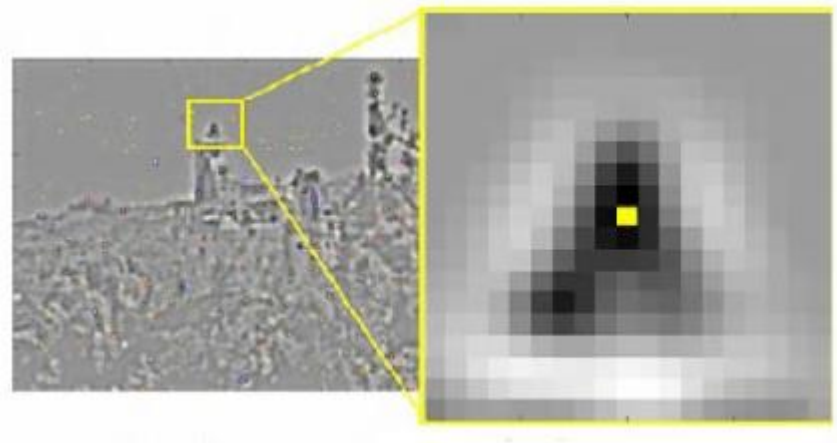
$$R = \frac{(r+1)^2}{r} \quad (4.13)$$

olarak belirtilmiştir.

4.3.3 Anahtar Noktaların Döngüsel Değişime Uyum Sağlaması

Bu adım, iki boyutlu düzlem üzerinde nesnedeki açısal değişime uyum sağlayarak anahtar noktayı kaybetmememizi sağlayacaktır. Bu evrede büyüklük ve yön hesaplanmaktadır.

Şekil 4.26'de gösterilen örnek anahtar noktayı ele alırsak, bu noktanın farklı açılarda döndürülmesiyle anahtar noktanın kaybolması şu şekilde engellenmektedir.



Şekil 4.26 Örnek anahtar nokta

Daha önce, σ ölçekli Gauss filtresinden geçirilmiş resim olan $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ kullanılarak gradyan büyüklüğü $m(x,y)$ ve açısı $\theta(x,y)$ hesaplanır.

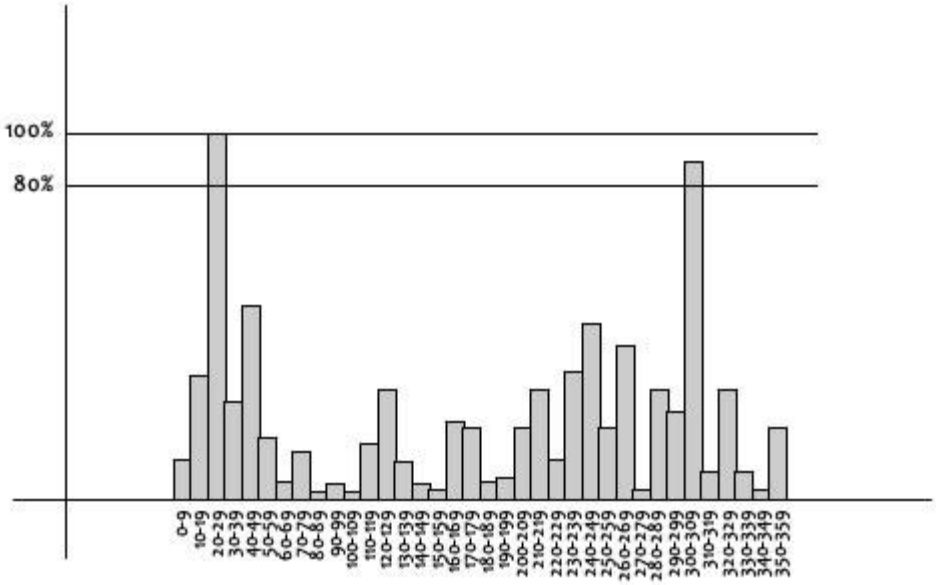
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (4.14)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/L(x+1,y) - L(x-1,y))) \quad (4.15)$$

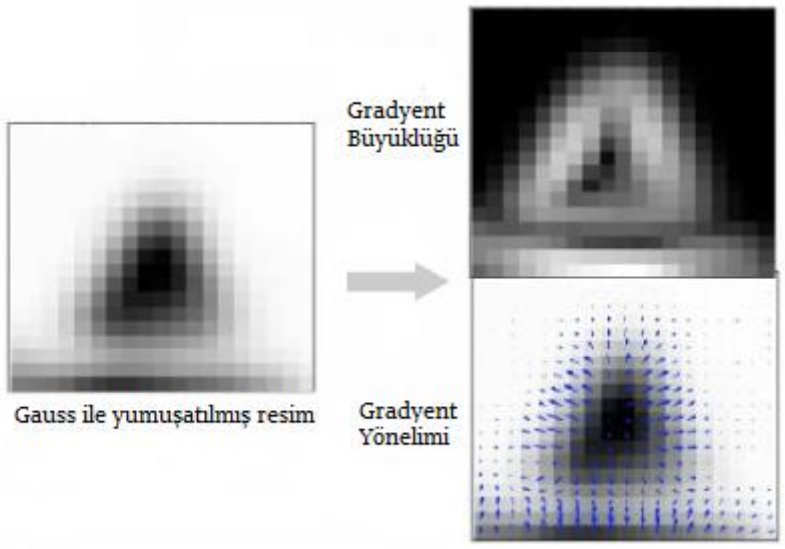
Büyüklük ve yön hesaplanması, belirlenen tüm noktalar için yapılır ve histogram oluşturulur. Bu histogram 36 binden oluşmaktadır (her biri 10 derecelik yönü kapsar) ve kendi gradyenti ile ağırlıklandırılmıştır.

Histogramda bazı binler uç noktaları göstermektedir. Örnek olarak Şekil 4.27'de bu değer 20-29 dereceler arasını kapsayan 3.bindir. Böylelikle anahtar nokta 3. bindeki açıda olacak şekilde atanır. Histogram içerisinde %80 üzerinde kalan binler de yeni anahtar noktanın açısal yönü olarak atanmaktadır. Yeni anahtar noktalar orijinal nokta ile aynı ölçek ve lokasyona fakat farklı döngüsel değerlere sahip olacaktır.

Sonuç olarak farklı döngülerle bir anahtar nokta birden fazla anahtar noktaya genişletilmiştir.



Şekil 4.27 Örnek döngüsel histogram



Şekil 4.28 Gradyent büyüklüğü ve yönelimi hesaplanmış örnek anahtar nokta

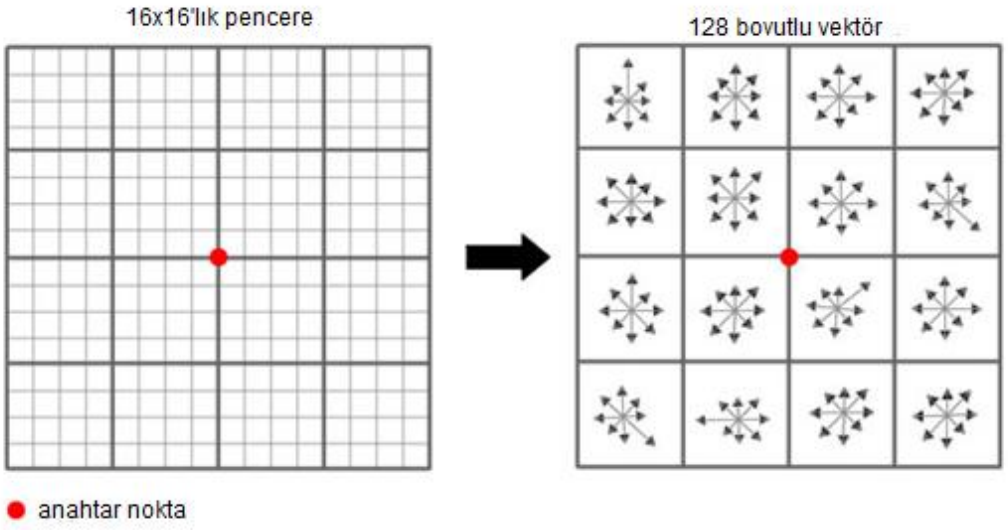
4.3.4 Anahtar Nokta Tanımlayıcısının Hesaplanması

Bu işlem ile daha önce belirlenen anahtar noktaların tanımlayıcılarını çıkartmış olacağız. Böylelikle hiçbir anahtar noktanın tanımlayıcısı birbirine benzemeyeceği için resimleri tanımlayıcıların üzerinden karşılaştırma işlemi yapabileceğiz.

Bunun için belirlenen anahtar nokta etrafında gradyan büyüklükleri ve açıları anahtar noktanın ölçeksel büyüklüğü kullanılarak $n \times n$'lik alanda örneklenir. Döngüsel

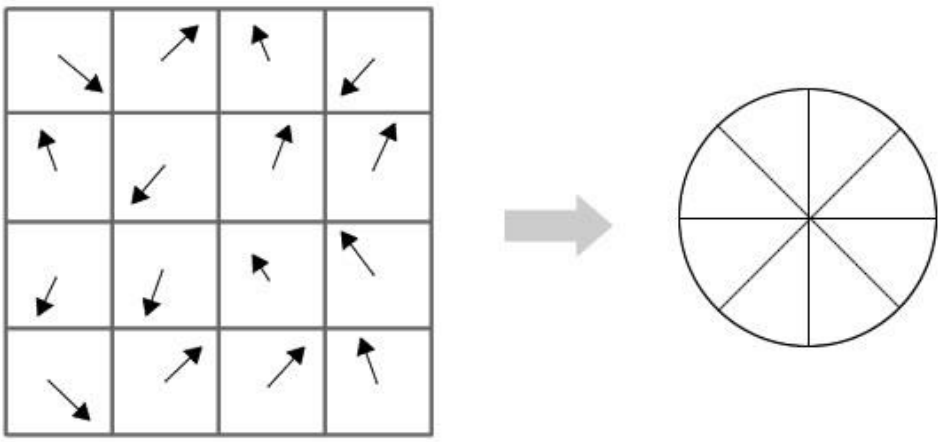
değişmezliği sağlamak için, anahtar nokta tanımlayıcının konumu ve gradyent açıları, anahtar nokta açısı temel alınarak döndürülür. 45 derecelik açılarla, anahtar noktanın etrafında bulunan *n x n*'lik alanın içinde kalan her bir nokta için tanımlanır.

Bu işlemde belirlenen anahtar noktanın etrafındaki 16x16 komşu pikseller Şekil 4.29'daki gibi seçilir ve bu pencere 4x4 olacak şekilde bölümlere ayrılır.



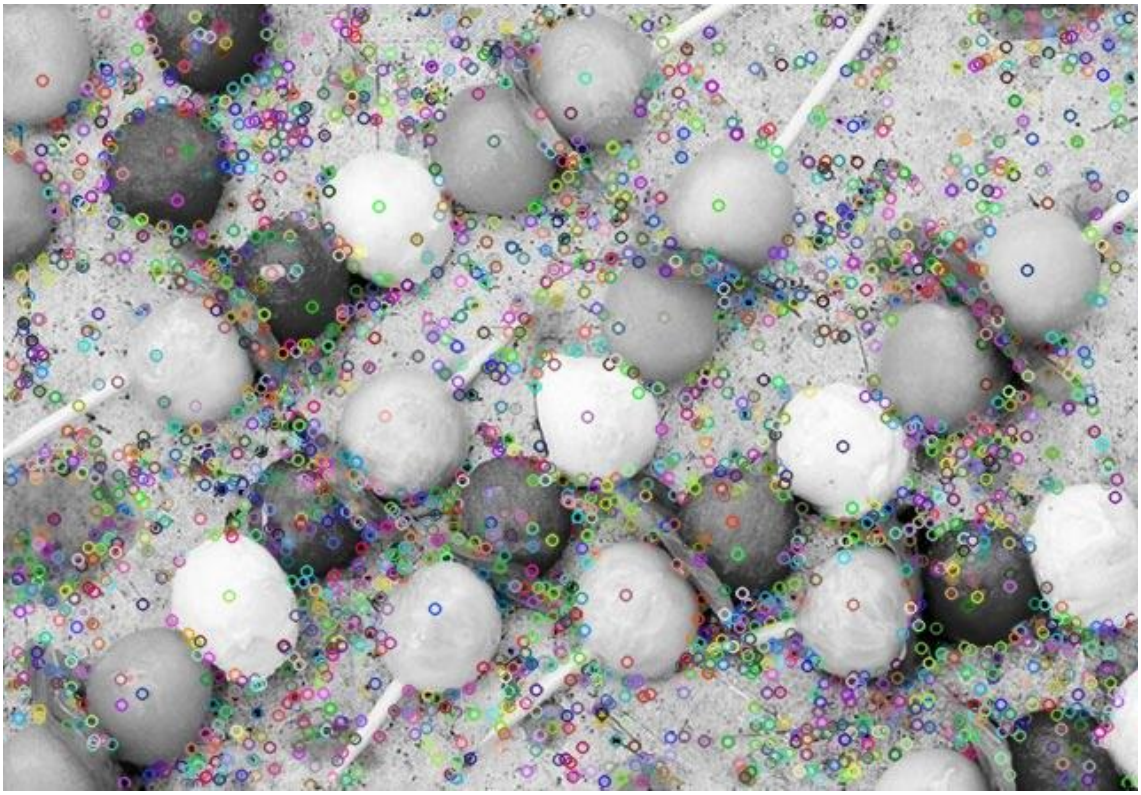
Şekil 4.29 Anahtar nokta tanımlayıcısının hesaplanması

Tüm 4x4 bölümlerin Şekil 4.30'daki gibi gradyent ve dönüsel değişimi hesaplanır. Bu değer 45 derecelik açılarca binlere (toplamda 8 bin) ayrılır ve histograma anahtar noktadan daha uzak olan gradyentler daha küçük değer alacak şekilde eklenir.



Şekil 4.30 Döngüsel değişimlerden 8 binlik histogram oluşturulması

Bu işlem tüm 16 pikselin için tüm 4x4 bölümleri için tekrarlanır. Sonuçta elimize her bir anahtar nokta için 4x4x8=128 büyüklüğünde bir özellik vektörü geçer. Benzerlik bulma işleminde bu vektörlerin aralarındaki uzaklıkların toplamı hesaplanarak yapılmaktadır.



Şekil 4.31 ÖBÖD anahtar özellikleri çıkartılmış örnek resim

4.4 Renk Haritalama

Resimlerin sahip oldukları renk dağılımı bulunarak bunun üzerinden arama yapıldığında; renk sayısının fazlalığından dolayı yeteri kadar başarılı sonuçlar alınamamaktadır. Renk haritası olarak adlandırdığımız bu algoritmada resmin baskın renklerini bulmak amaçlanmaktadır.

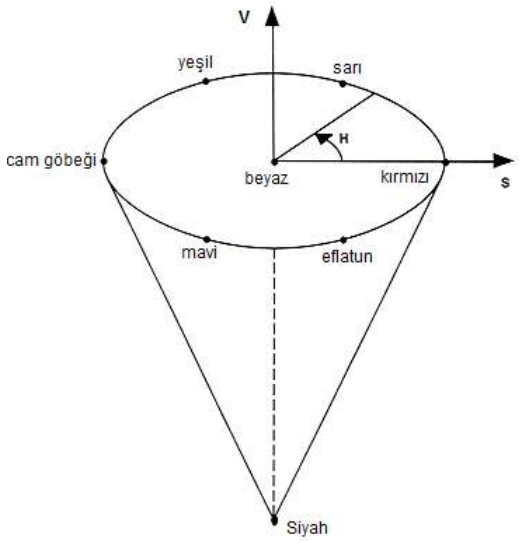
Bu işlem 2 aşamadan oluşmaktadır.

- Referans renklerin belirlenmesi
- Renklerin indirgenmesi

4.4.1 Referans Renklerin Belirlenmesi

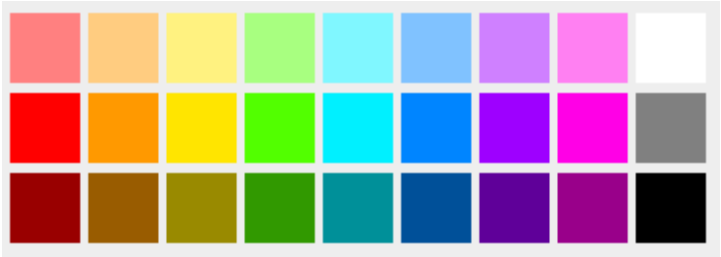
Referans renkleri belirlemede dikkate alınan nokta, belirlenen renklerin birbirine ne çok uzak nede çok yakın olmasıdır. İnsan gözü ile bakıldığında birbirinden farklı belli sayıda renkleri belirlememiz gerekmektedir.

Referans renkleri belirlemede HSV renk uzayı kullanılmıştır. Şekil 4.32'de de görüldüğü gibi Hue {0 - 360} derecelik bir aralığa sahiptir ve her açı bir rengi vermektedir. Saturation ise rengin doygunluğunu ve Value ise rengin parlıklığını vermektedir. S ve V'nin aralığı {0 - 100} arasında değişmektedir. Referans renkleri oluştururken Hue değerlerinden aralarında eşit açı olan n adet seçilir. Bu seçim işleminde S ve V değerleri maksimumu alınır (100). Daha sonra istenilen k değeri kadar V ve S değerleri 100'den 0'a doğru azaltılarak renkler oluşturulur.



Şekil 4.32 HSV renk uzayı

Bu işlemler sonrasında belirlediğimiz referans rengimiz Şekil 4.33'deki gibidir.



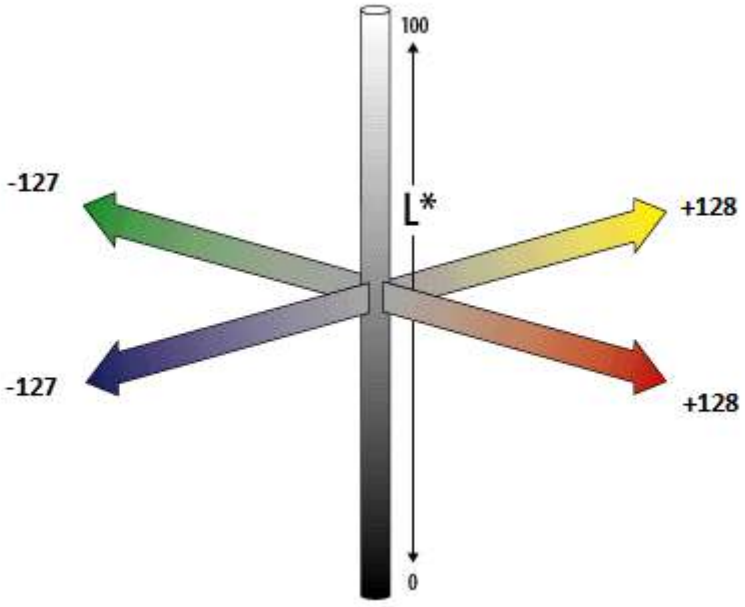
Şekil 4.33 Referans renkler

4.4.2 Renklerin İndirgenmesi

Bu aşamada belirlenen referans renkler baz alınarak resmin her bir pikseli en yakın renk ile değiştirilir. Bu işlem sırasında CieLab renk uzayı kullanılmıştır.

CieLab renk uzayı Uluslararası Aydınlatma Komisyonu tarafından insan gözünün renklerin birbirleri ile olan benzerlikleri dikkate alınarak uzaklığı belirlenen bir renk uzayıdır. Bu tezde benzerlik nihayetinde insan gözü ile değerlendirileceği için CieLab renk uzayını kullanmayı tercih ettik.

CieLab renk uzayı L,a,b olak üzere Şekil 4.34'de görüldüğü gibi 3 değerden oluşmaktadır.



Şekil 4.34 CieLab renk uzayı

Burada L değeri 0 iken siyah, 100 iken beyazı göstermektedir. a değeri ise -127 iken yeşil, +128 iken kırmızı rengi göstermektedir. b değeri ise -127 iken eflatun, +128 iken sarı rengi göstermektedir.

Resmin piksel değerleri referans renklerin CieLab uzayındaki en yakın olan ile değiştirilir. Bu değiştirme işleminde Ciede2000[45] renk uzaklığı algoritması kullanılmıştır.

Renk indirgeme işleminden sonra yeni resmin histogramı çıkartılır. Histogram normalize edilerek her bir rengin resmin içerisinde bulunma yüzdesi hesaplanır. Yüzdesi en fazla olan renk, resim için dominant rengini verecektir.

Şekil 4.35'de görüldüğü gibi, resmin en fazla renk oranına sahip 5 renk gösterilmiştir.

(a) (b)

Şekil 4.35 Renk haritası örnekleri. a) orijinal resim, b) renk haritası.

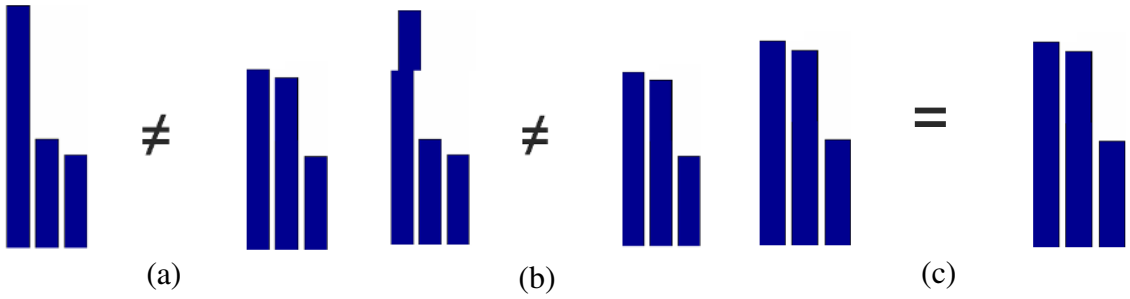
4.4.3 Renk Histogramların Benzerliklerinin Hesaplanması

Histogramları çıkartılmış 2 resmin birbirine ne denli benzediğinin hesaplanması ile resim arama motorunda resimlerin sıralanması yapılacaktır. Burada bizim için renklerin birbirine olan uzaklığı ne kadar önemli ise renklerin resim içerisindeki bulunma oranı da o denli önemli. Örnek vermek gerekirse, A resminde yeşil renk %50 orana sahip, B resminde bu oran %90, C resminde ise bu oran %51 olsun. Burada hepsi aynı renk olduğu için CieLab renk uzayında Ciede2000 uzaklığı 0 çıkacaktır. Fakat C resminin A resmine olan benzerliği, B'nin A resmine olan benzerliğinden daha fazladır. Bu nedenle uzaklık hesabı ağırlıklandırılarak yapılmıştır.

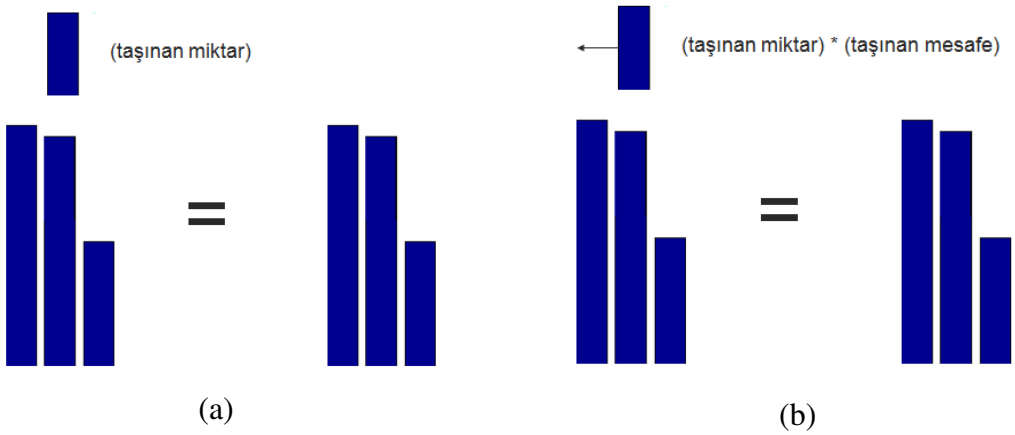
4.4.3.1 Toprak Taşıcısının Mesafesi

Toprak Taşıyıcısının Mesafesi (Earth Mover's Distance - EMD) algoritması çok boyutlu dağılımların belli bir özellik uzayında, özelliklerin birbirleri ile olan farklılıkları hesaplamaktadır. Bu farklılıklar; kullanılan özelliklerin birbirine olan uzaklığı ve zemine olan uzak ile hesaplanmaktadır. EMD bu farklılığı bireysel özelliklerden tam dağılıma yaymaktadır.

Örnek olarak 2 dağılım düşünelim. Bunlardan birisi belli bir alana yayılmış toprak kütlelerini, diğeri ise aynı alanda bulunan dağılmış çukurları göstermektedir. EMD çukurları toprak ile doldurmak için gerekli olan en az iş yükünü (maliyeti) hesaplamaktadır Şekil 4.37. Burada belirtilen iş yükü belli bir toprak miktarının belli bir mesafe taşınmasına denk gelmektedir. İşte buradaki mesafe zemine olan mesafe olarak nitelendirilmektedir.



Şekil 4.36 Histogramların birbirine benzetilmesi. a) birbirinden farklı iki histogram, b) histogramların birbirine olan farklılığı, c) birbirine benzeyen iki histogram



Şekil 4.37 Histogramları birbirine benzetirken ki maliyet hesabı. a) Benzetme işlemi için gerekli olan taşıma miktarı. b) taşınacak olan miktarın taşınacak mesafe ile çarpılıp taşıma maliyetinin hesaplanması.

EMD hesaplaması yaygın olarak bilinen taşımacılık problemi [49] çözümünden ilham almıştır. Diyelim ki her birinde belli sayıda ürün olan ve bu ürünleri sınırlı sayıda ürün alabilen müşterilere dağıtması gereken birçok tedarikçimiz var. Her bir tedarikçi ve müşteri ikilisi için, birim ürünün taşınması için gerekli olan maliyet hesaplanmış olsun. Buradaki taşıma problemi tedarikçilerin ürünleri tüm müşterilerin ihtiyacını cevap verecek şekilde en az maliyetle dağıtmasıdır.

$P = \{(p_1, w_{p_1}), (p_2, w_{p_2}), \ldots\ldots (p_m, w_{p_m})\}$ m kümesinin imzaları dersek, p_i kümeyi, w_{p_i} kümenin ağırlığını vermektedir.

$Q = \{(q_1, w_{q_1}), (q_2, w_{q_2}), \ldots\ldots (q_m, w_{q_n})\}$ n kümesinin imzaları dersek, q_i kümeyi, w_{q_i} kümenin ağırlığını vermektedir. $D = [d_{ij}]$ zemin mesafe matriksini, d_{ij} ise p_i ve q_j arasındaki zemin mesafesini göstermektedir. p_i ve q_j arasındaki f_{ij} akışı için toplam maliyeti minimum yapan $F = [f_{ij}]$ değerini bulmak istiyoruz.

$$Work(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij} \tag{4.16}$$

$$f_{ij} \geq 0 \;\; 1 \leq i \leq m, 1 \leq j \leq n \tag{4.17}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \;\; 1 \leq i \leq m \tag{4.18}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j} \;\; 1 \leq j \leq n \tag{4.19}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj}\right) \tag{4.20}$$

(4.17) ürünlerin sadece P den Q ya hareket etmesini sağlamaktadır.

(4.18) ve (4.19) P kümesinden gönderilecek ürünlerin miktarını ağırlıkları oranınca gönderilmesini ve Q kümesi ağırlığından fazla ürünü almamasını sağlamaktadır. (4.20) ise taşınacak ürünün maksimum sayıda olmasını sağlamaktadır. Bu miktara toplam akış adı verilmektedir. Taşıma problemi çözülüp optimum akış olan F'yi bulduktan sonra EMD, toplam akışta normalize edilmiş iş olarak nitelendirilir.

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{4.21}$$

4.4.3.2 Histogram benzerliğinin bulunması

Histogramların birbirine ne kadar benzediğinin bulunması işleminde Toprak Taşıcısının Mesafesi [46] algoritması kullanılmıştır. Bu algoritma ile A histogramının B Histogramına benzetilmesi için gerekli olan maliyeti hesaplamaktadır. Maliyeti büyük olan histogramlar birbirine o denli benzememektedir. Maliyet hesaplamasında makalede zemin mesafesi olarak adlandırılan değer binlerin histogramdaki hareket maliyetini vermektedir. Bizde hareket maliyetini renklerin Ciede2000 uzaklığı farkının karesi ile

resim içerisindeki yüzdelerin farklarının karesinin toplamının karekökü şeklinde belirledik.

$$\rho = \sqrt{(d_1 - d_2)^2 + Ciede2000(x_1 - x_2)^2} \quad (4.22)$$

Bu şekilde resimleri, histogramları birbirine benzetmek için maliyeti düşük olandan büyük olana göre sıraladığımızda, resimleri renk benzerliğine göre de sıralamış olmaktayız.

BÖLÜM 5

RESİM ARAMA MOTORU MİMARİSİ

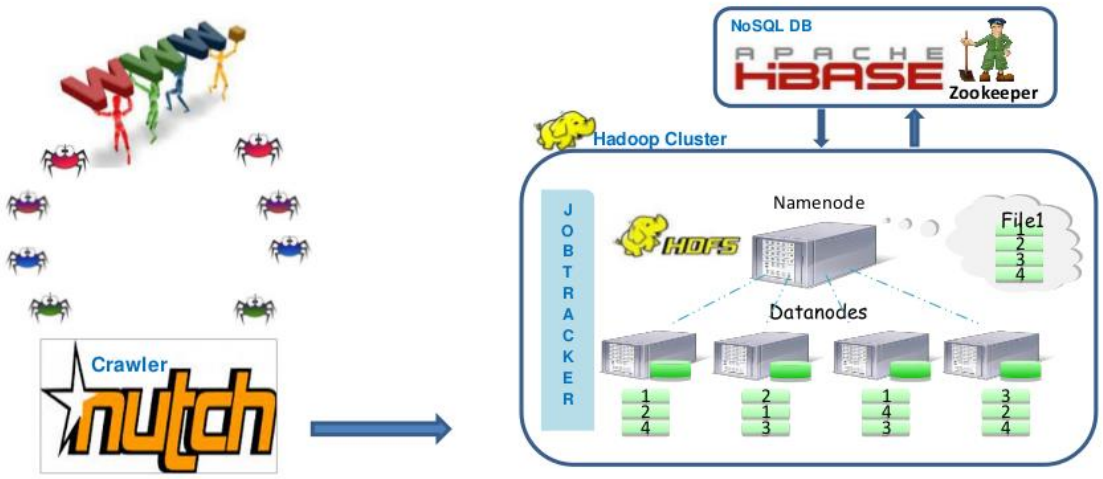
Tüm mimaride 4 adet 2 çekirdekli ve 8 GB ram kapasitesine sahip bilgisayarlar kullanılmıştır. Resimlerin saklanması ve işlenmesi için gerekli olan dağıtık dosya sistemi Apache Hadoop ve resim öz niteliklerinin indekslenerek aranabilir yapmak için gerekli olan Apache Solr sistemleri bu bilgisayarlar üzerine kurulmuştur.

Resim arama motoru mimarisi içerik çekme, resimleri işleme, indeksleme ve arama olmak üzere 4 aşamadan oluşmaktadır.

5.1 İçerik Çekme

İçerik çekme işleminde çekilen verilerin saklanması için Apache Hadoop üzerinde HDFS dosya sisteminde çalışan Apache HBASE veritabanı kullanılmıştır.

İçerik çekme işleminde belirlenen 20 adet haber sitesinin ana sayfaları tohum url olarak Nutch içerisinde kullanılmıştır. Nutch verilen bu tohum url'lere uğrayarak ham halinde web sayfaları çekilmektedir. Çekilen sayfalar html kodlarından ayrıştırılarak yazılar ve sayfada bulunan dış linkler HBASE veritabanına kaydedilmektedir. Bir sonraki döngüde bir önceki sayfanın dış linklerine uğramaktadır ve burada da aynı işlemler tekrarlanmaktadır. (Şekil 5.1)



Şekil 5.1 İçerik çekme mimarisi

Tezde bu işlem günde 2 defa olmak üzere 3 derinlik sabitlenmiştir. Bir diğer kısıtlamamız ise sitelerden bulunan dış linkler yine ilk başta verilen 20 adet haber sitesinin herhangi birine ait olmasıdır. Böylelikle sadece belirtilen haber sitelerinde içerik çekme işlemi tamamlanmıştır. Tüm veri çekme işlemi 5 gün yapılmış toplamda yaklaşık 200bin resim toplanmıştır. Resim çekme işleminde minimum kenar uzunluğu 300px altında olan resimler ise elenmiştir. Çekilen resimler hangi siteden çekildiğinin bilgisi korunarak daha sonra işlenmek üzere HBASE veri tabanına ikili kaydedilmiştir.

5.2 Resimleri İşleme

Resim işleme adımı içerik çekme adımından ayrıştırılarak her iki adımında birbirine paralel olarak çalışması sağlanmıştır. Resim işleme adımı uzun süren bir iş olduğu için içerik çekme işlemini engellememesi istenmiştir.

Resim işleme adımında Apache HBASE veri tabanına kaydedilen resimler MapReduce işleri ile alınıp daha önce belirlenen görüntü işleme algoritmaları yardımı ile öz nitelikleri çıkartılmıştır. Şekil 5.2'de Resim işleme mimarisi gösterilmiştir.

Şekil 5.2 Resim işleme mimarisi

Kullanılan görüntü işleme algoritmaları sırası ile BRDH, HRAN, ÖBÖD ve Renk haritalamadır. Belirlenen bu algoritmalar yine MapReduce yapısı altında çalıştırılarak resimlerin öz nitelikleri çıkartılmış ve çıkartılan bu öz nitelikler Apache Solr'da indekslenmiştir.

Burada BRDH algoritması biraz farklı şekilde kullanılmıştır. Resme BRDH algoritması uygulanarak elde edilen özellik vektörü, 516 bit büyüklüğünde float veri tipine sahiptir. Biz bu vektörün ham halinin yanı sıra vektörün LSH (Local Sensetivity Hashing) algoritması ile karmaşası alınıp dizgi karşılığını Apache Solr üzerinde indeksledik. Bu işlemin sebebini indeksle ve arama başlığında bulabilirsiniz.

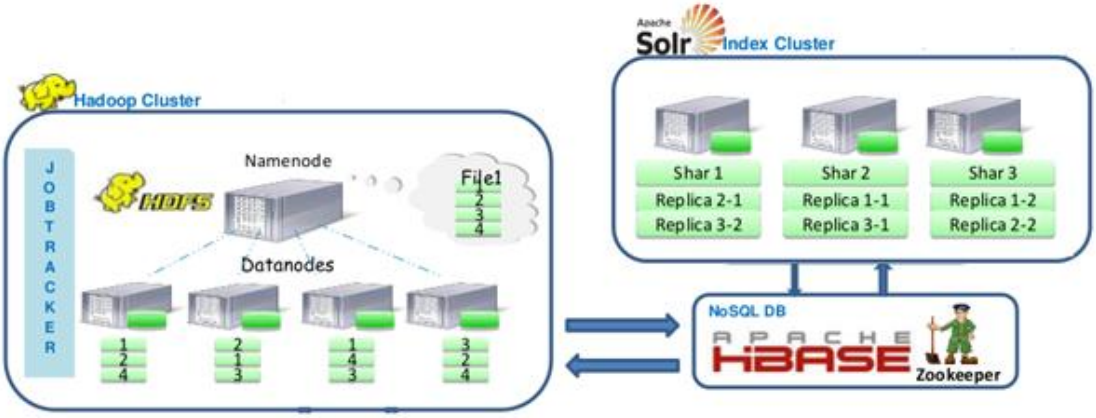
Locality Sensetive Hashing (LSH):

İnternet üzerinde indeksleme ve sıralama yapılırken karşılaşılan sorunlardan birisi doküman içeriklerinin pek çok kopyasının olmasıdır. Bu nedenle bir dokümanın içeriğinin orijinal olduğu ya da daha önceden indekslenip indekslenmediğinin bulunması gerekir. Ayrıca benzer içeriğe sahip dokümanların kümelenmesi için doküman yakınlığını bulan algoritmaların kullanılması gerekir. Bu konuda özellikle Hashing teknikleri kullanılmaktadır. Yaygın olarak kullanılan teknik ise Locality Sensitive Hashing (LSH)'dir. Bu yöntem aslında olasılıksal bir boyut indirgeme yöntemidir. Biyometrik arama için de kullanılan bu yöntem arama motorlarında eş ya da benzer içeriğe sahip dokümanların bulunması, spam sitelerinin tespiti ve kümeleme için kullanılmaktadır.

5.3 İndeksleme

İndekslenen veriler sorgu sırasında kullanılmaya olanak sağlamaktadır. Resimleri indeksleme ve arama işlemi yapabilmek için 3.1'de anlatılan Apache Lucene projesi kullanılarak oluşturulan, türetilen Apache Solr[47] Projesini kullandık. Apache Solr mimarisinde 2 türlü veri saklama yöntemi vardır. Bunlar proje içerisinde stored ve indexed olarak adlandırılmışlardır. Aralarındaki en önemli fark indekslenmeyen sadece stored olarak saklanan alan üzerinde daha önce anlatılan indeks mekanizması çalıştırılamaması ve bu alan baz alınarak arama yapılamamasıdır. Bir alan hem indexed hemde stored olabilmektedir.

Tez içerisinde HRAN ve ÖBÖD algoritması kullanılarak çıkartılan öz nitelikler Solr içerisinde stored olarak saklanmıştır. BRDH ve Renk bilgileri ise indexed olarak saklanmıştır. Şekil 5.3'de indeksleme mimarisi gösterilmiştir.

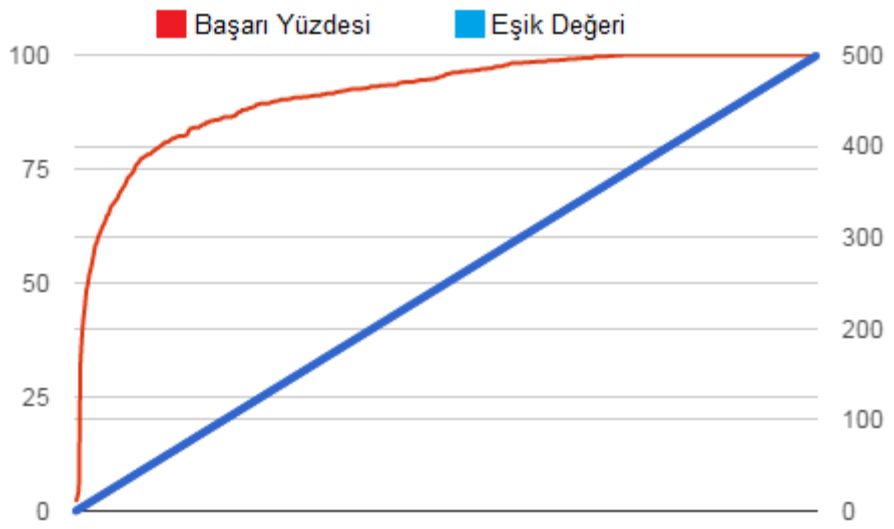


Şekil 5.3 İndeksleme mimarisi

5.4 Arama

Arama işleminde en başarılı sonuçları ÖBÖD algoritması vermektedir. Buradaki başarı precission ve recall değerleri ile ölçülmüştür. Sadece ÖBÖD algoritması kullanarak yaklaşık 5000 resim içerisinde arama yaptığımızda, sorgu sonucu takriben 20dakikada dönmektedir. Bu sayı 200binlere çıktığında bir sorgu beklenmeyecek kadar uzun sürmektedir. Her ne kadar ÖBÖD benzerlik bulmada başarılı olsa da, cevap zamanı arama işlemini başarısız kılmaktadır. Bu nedenle daha uygulanabilir bir çözün için kademeli arama mimarisi kullanmaya karar verildi.

Bu mimaride başarısı düşük fakat hızlı olan BRDH algoritması ile ön eleme yapıldı. Ön eleme işleminde dikkat edilmesi gereken en önemli konu, eleme sırasında benzer resimleri de eleme riskidir. Bu riski en aza indirmek adına bazı deneyler yaptık. Bu deneylerde "BRDH algoritması sonuçlarının ilk kaç resmi alalım ki benzer resimleri elememiş olalım?" sorusuna cevap aradık. Burada Şekil 5.3'de görüldüğü gibi BRDH algoritmasının ilk 500 resmini almaya karar verdik.



Şekil 5.4 BRDH başarı yüzdesi

BRDH sorgu sonucu ilk sıralarda beklediğimiz resimleri getirmiyor olabilir, zaten nihai sonuca BRDH ile varmayacağız ama şunu biliyoruz ki ilk 500 içerisinde beklediğimiz resimlerin tamamı mevcut. Bu nedenle eşik değeri olarak 500 seçtik. Kademeli aramanın ilk aşaması olan BRDH algoritması ile resimleri eleme işleminde, indekslenen tüm BRDH vektörlerinin ve sorgu ile gönderilen resmin BRDH vektörünün karmaşası kullanıldı. Bu karmaşalar baz alınarak Solr sorgusu çalıştırıldı. Sorgu ile indekslenen karmaşalar arasında sorgulanan resmin karmaşasına 0.9 oranında benzeyen 500 adet resim getirilmiştir. BRDH algoritması öz nitelik vektörü çıkartılırken son aşamada histogramı 8 bulanık mantıksal alana bölmektedir. Bu nedenle aynı alana sahip olan vektörlerin karmaşalarıda belli oranda birbirine benzeyecektir. Sonuç olarak ilk aşamada BRDH algoritması yardımı ile sorgu kümesini 500e indirgemiş ve sıralamış olmaktayız. Bu indirgeme işlemi indekslenmiş veri üzerinde yapıldığı için 0.2 saniyede gerçekleşmektedir.

Belirlenen 500 resim ÖBÖD algoritması ile tekrar sıralanmış ve bu şekilde sonuca ulaşılmak istenmiştir. Sonuç zamanı bu şekilde 50 saniyeye düşmektedir. Fakat hala bir arama motoru için yeterli bulunmamıştır. Çizelge 5.3'de algoritmaların sorgu sonucunun dönüş zamanları gösterilmiştir.

Çizelge 5.3 Sorgu zamanları ve başarı yüzdeleri

Algoritma	**Ortalama Sorgu Zamanı**	**Başarı Yüzdesi**
BRDH	0.2 saniye	58
ÖBÖD	> 1 saat	*
BRDH + ÖBÖD	50 saniye	88
BRDH + HRAN	4 saniye	66
BRDH + HRAN + ÖBÖD + Renk Haritalama	15 saniye	92

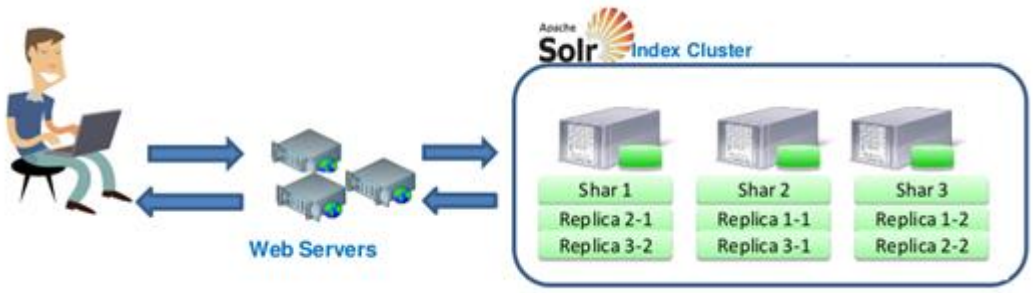
***İşlem çok uzun sürede geldiği için sunucular bellek hatası vermekte, sonuç alınamamaktadır**

İşlem yükünü başarıyı düşürmeden azaltmak gerekmektedir. Bu işlem yükü azaltma sırasında gayet başarılı olan HRAN[20] algoritmasını kullanmaya karar verdik. Daha önce anlatıldığı gibi HRAN algoritması belirlenen anahtar noktalarını insan retinasından ilham alarak tanımlamaktadır. Tanımlama işlemi sonunda her resim için, ikili vektör oluşturmaktadır. Bu, hem indekslenen verinin azalması hem de benzerlik hesabının daha az maliyetli olmasını sağlamaktadır. Böylelikle HRAN algoritması bize ikinci bir resim sayısını indirgeme olanağı sağlamaktadır. Burada Brisk [19] algoritmasının anahtar nokta tespiti kısmı ile tespit ettiğimiz anahtar noktaları, HRAN algoritması ile tanımlayarak tüm kümeyi indeksledik.

Bu işlemler ile 500'e indirgenen resim kümesi HRAN ve ÖBÖD algoritmaların başarıları birleştirilerek, belli bir eşik değeri yardımı ile 60'a indirgendi. Sonuç olarak elimize doku olarak birbirine en çok benzeyen resimler sıralanmış olarak geçmektedir.

İlk 60 resmin tamamı sorgu resmi ile benzer olmayabilir. Doku olarak benzeyenler zaten ilk sıralarda yer almaktadır fakat doku olarak yakalanamayan ancak renk dağılım olarak birbirine benzeyen resimler ise renk haritalama algoritması ile sıralanmıştır.

Kademeli aramada son indirgeme işlemi ile sorgu sonucu yaklaşık 15 saniyede dönmektedir. Bu mimari ile resim sayısı 200bin den 2milyon'a çıkarsak bile ilk eleme olan 500 resme indirgeme, indekslenen veriler üzerinden yapıldığı için sorgunun cevap zamanı yine çok düşük olacaktır.



Şekil 5.5 Resim arama mimarisi

İndekslenmiş tüm resim kümesine daha önceden belirlenen 30 resim ve bu 30 resmin n adet benzer resimleri eklenmiştir. Daha sonra eklenen 30 resim tek tek sorgulanarak beklenilen n adet resimlerin ilk sıralarda olma oranı dikkate alınarak başarı hesaplanmıştır. İlk n sıradaki resimlerin tamamı beklenen n adet alakalı resim gelmesi gerekmektedir. Sorgulanan her bir resim için bu n değeri farklıdır.

Resim sonuçlarının başarıları aşağıdaki formüllerle değerlendirilmiştir.

Benzer resimlerden kaç tanesi elde edildi? Cevabını almak için:

$$Erişimisabeti(Recall) = \frac{|\{benzer\ resimler\} \cap \{elde\ edilen\ resimler\}|}{|\{benzer\ resimler\}|} \quad (5.1)$$

$$= \frac{DP}{DP+YN}$$

30 resim sorgulanarak erişim isabeti değeri hesaplanmıştır.

2. bir test sorgusunda 0,03 eşik değeri, doku benzerlik oranına uygulanarak "Elde edilen resimlerden kaç tanesi benzer?" cevabı aranmıştır.

Elde edilen resimlerden kaç tanesi benzer? Cevabını almak için:

$$\text{Kesin isabet(Precision)} = \frac{|\{\text{benzer resimler}\} \cap \{\text{elde edilen resimler}\}|}{|\{\text{elde edilen resimler}\}|} = \frac{DP}{DP+YP} \quad (5.2)$$

Her bir sorgu resmi için hesaplanan değerler ve uygulanan farklı eşik değerlerin sonuçları aşağıdadır

Sorgu 1

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{7}{7} = 1$$

Sorgu 2

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{5} = 1$$

Sorgu 3

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{7}{7} = 1$$

Sorgu 4

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{4} = 1$$

Sorgu 5

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$

Sorgu 6

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{4} = 1$$

Sorgu 7

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{6}{6} = 1$$

Sorgu 8

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{7}{7} = 1$$

Sorgu 9

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 10

Veri setinde 10 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{10}{10} = 1$$

Sorgu 11

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$

Sorgu 12

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{6} = 0.8$$

Sorgu 13

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{4} = 0.75$$

Sorgu 14

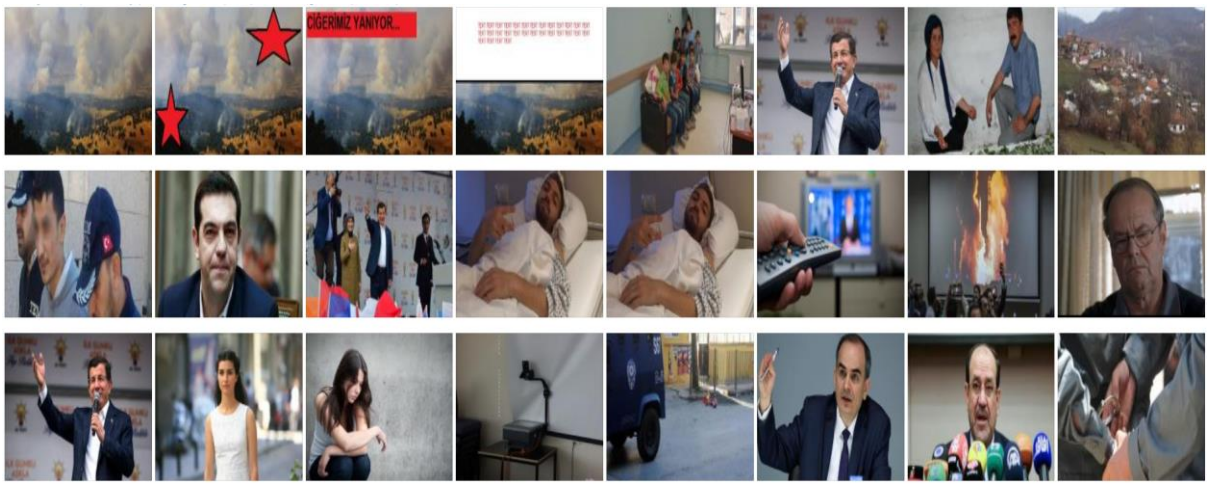
Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 15

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{5} = 0.8$$



Sorgu 16

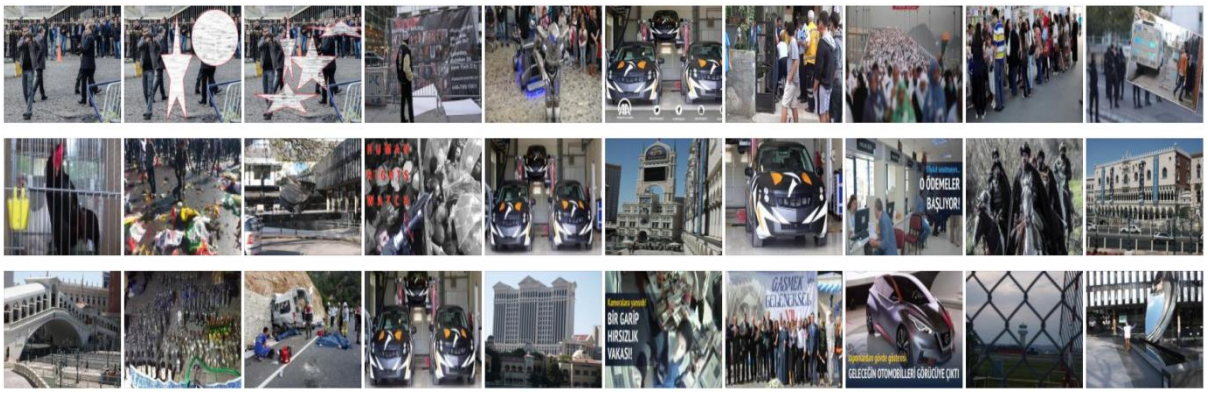
Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 17

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{4} = 0.75$$



Sorgu 18

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{5} = 1$$

Sorgu 19

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{6}{6} = 1$$

Sorgu 20

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{4}{4} = 1$$

Sorgu 21

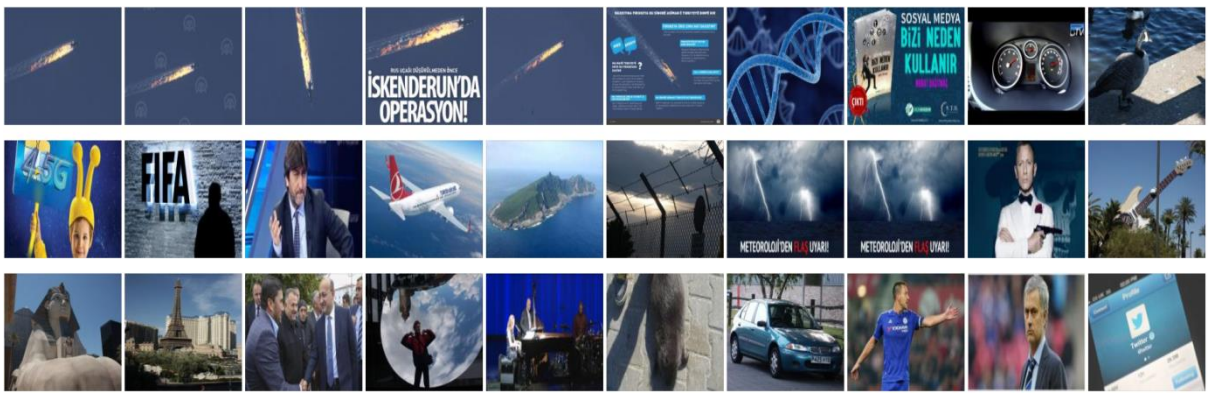
Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{5} = 1$$

Sorgu 22

Veri setinde 6 adet benzer resim bulunmaktadır.

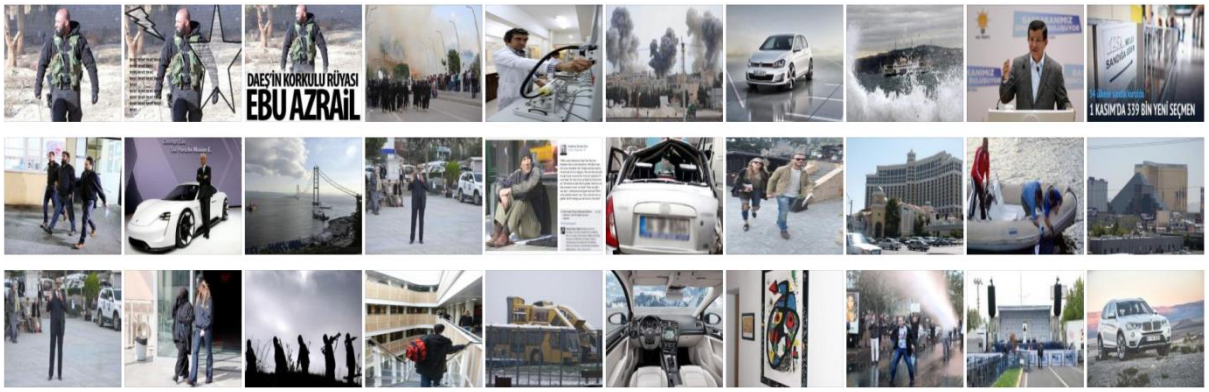
$$\text{Recall} = \frac{6}{6} = 1$$



Sorgu 23

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$



Sorgu 24

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$

Sorgu 25

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$

Sorgu 26

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{6} = 0.8$$

Sorgu 27

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{3} = 1$$

Sorgu 28

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{3}{4} = 0.75$$

Sorgu 29

Veri setinde 11 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{11}{11} = 1$$

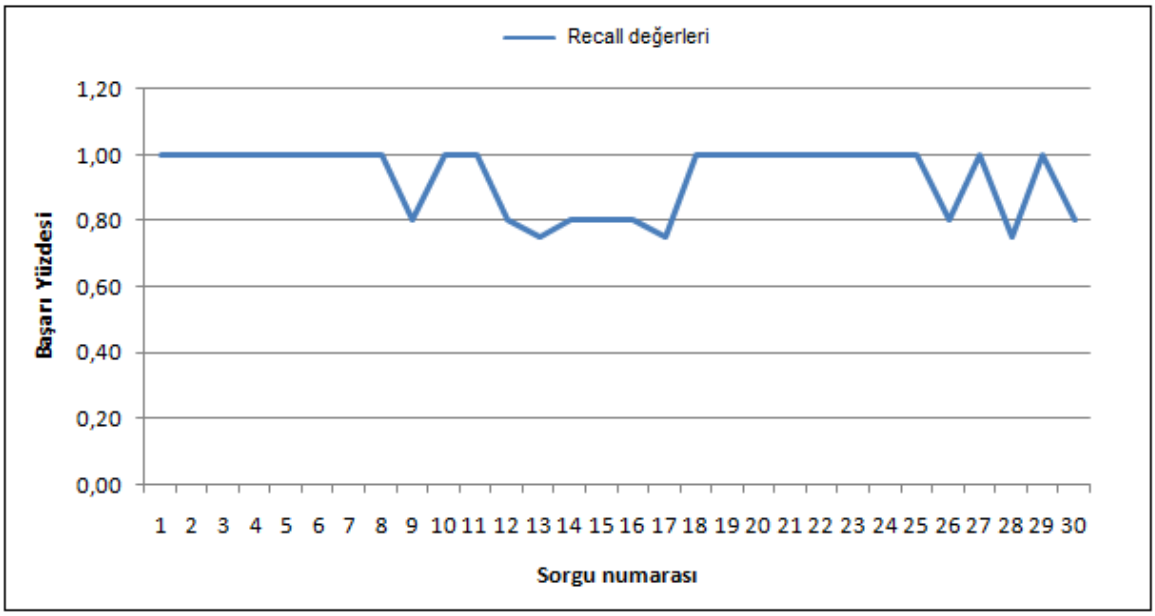
Sorgu 30

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Recall} = \frac{5}{6} = 0.8$$

Çizelge 5.4 Hesaplanan recall değerleri

Sorgu No	Recall %
Sorgu 13, 17, 28	0.75
Sorgu 9, 12, [14 -16], 26, 30	0.8
Sorgu [1-8], 10, 11, [18-25], 27, 29	1
Ortalama	**92**



Şekil 5.6 Hesaplanan recall değerleri

0.03 Eşik değeri ile hesaplanan değerler:

Sorgu 1

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{6}{6} = 1, \text{Recall} = \frac{6}{7} = 0.85$$

Sorgu 2

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{5}{5} = 1, \text{Recall} = \frac{5}{5} = 1$$

Sorgu 3

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{7}{7} = 1, \text{Recall} = \frac{7}{7} = 1$$

Sorgu 4

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{4} = 1$$

Sorgu 5

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$

Sorgu 6

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{4} = 1$$

Sorgu 7

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{6}{6} = 1, \text{Recall} = \frac{6}{6} = 1$$

Sorgu 8

Veri setinde 7 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{6}{6} = 1, \text{Recall} = \frac{6}{7} = 0.85$$

Sorgu 9

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{1}{1} = 1, \text{Recall} = \frac{1}{5} = 0.2$$

Sorgu 10

Veri setinde 10 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{10}{10} = 1, \text{Recall} = \frac{10}{10} = 1$$

Sorgu 11

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$

Sorgu 12

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{5}{5} = 1, \text{Recall} = \frac{5}{6} = 0.8$$

Sorgu 13

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{4} = 0.75$$

Sorgu 14

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 15

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 16

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{5} = 0.8$$

Sorgu 17

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{4} = 0.75$$

Sorgu 18

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{5}{5} = 1, \text{Recall} = \frac{5}{5} = 1$$

Sorgu 19

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{6}{6} = 1, \text{Recall} = \frac{6}{6} = 1$$

Sorgu 20

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{4} = 1$$

Sorgu 21

Veri setinde 5 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{5} = 0{,}8$$

Sorgu 22

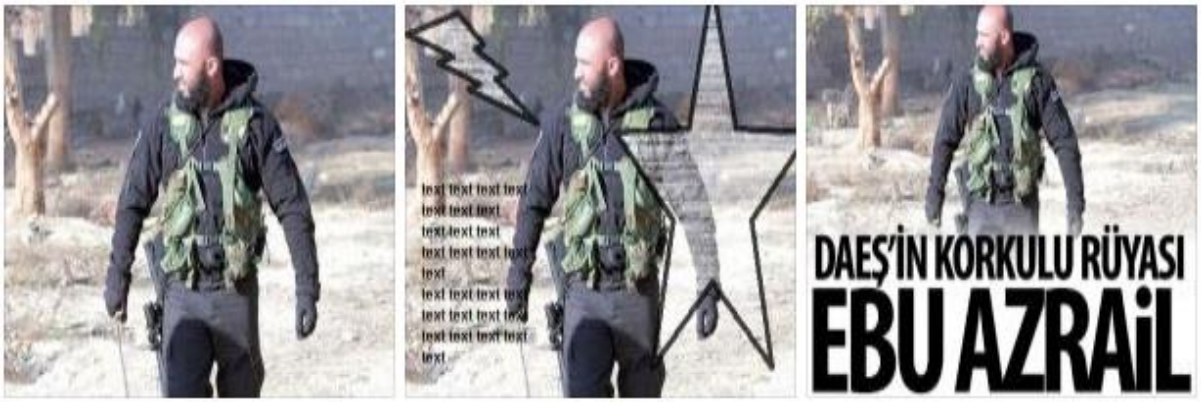
Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{4}{4} = 1, \text{Recall} = \frac{4}{6} = 0.66$$

Sorgu 23

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$



Sorgu 24

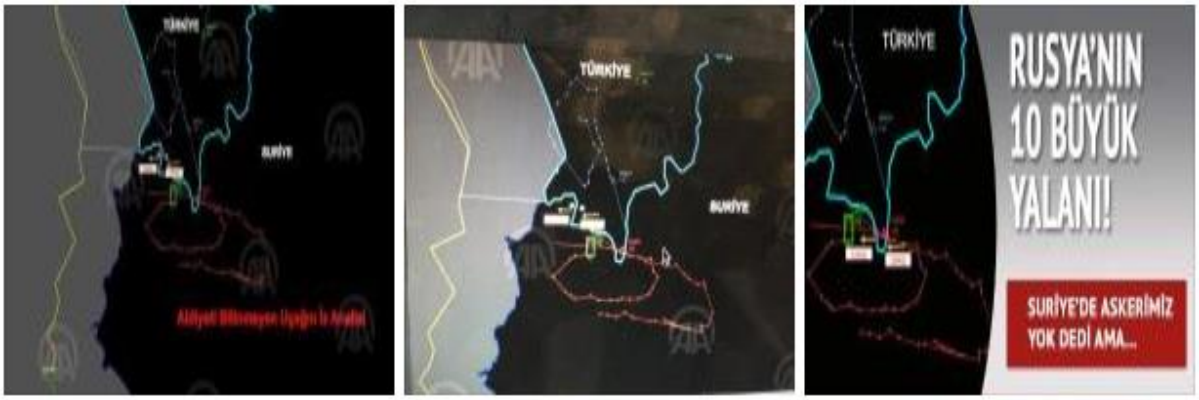
Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$

Sorgu 25

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$



Sorgu 26

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{5}{5} = 1, \text{Recall} = \frac{5}{6} = 0.8$$

Sorgu 27

Veri setinde 3 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{3}{3} = 1, \text{Recall} = \frac{3}{3} = 1$$

Sorgu 28

Veri setinde 4 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{2}{2} = 1, \text{Recall} = \frac{2}{4} = 0.5$$

Sorgu 29

Veri setinde 11 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{11}{11} = 1, \text{Recall} = \frac{11}{11} = 1$$

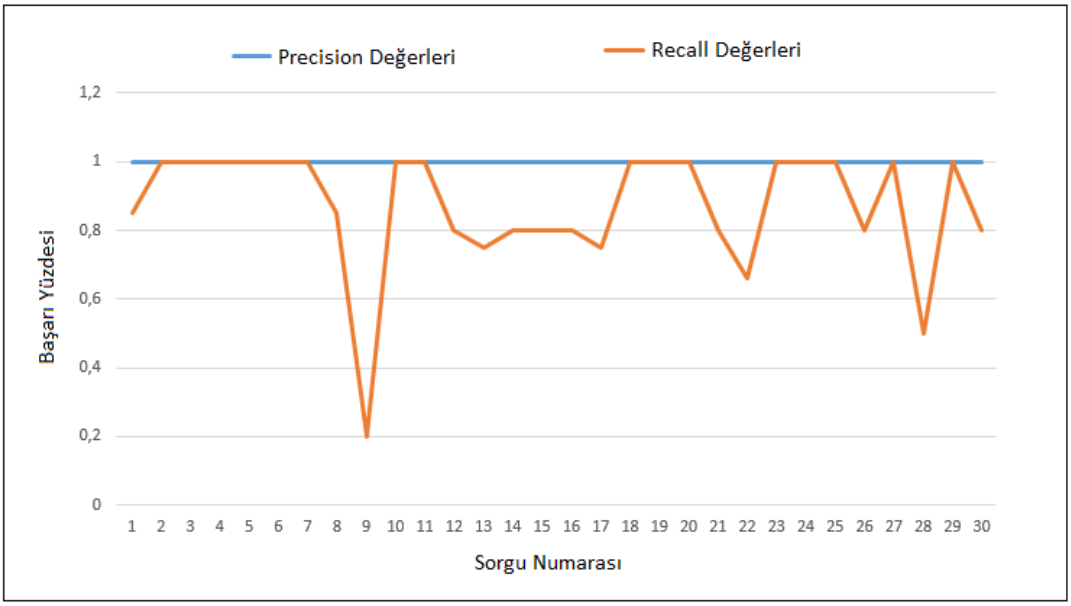
Sorgu 30

Veri setinde 6 adet benzer resim bulunmaktadır.

$$\text{Precision} = \frac{5}{5} = 1, \text{Recall} = \frac{5}{6} = 0.8$$

Çizelge 5.5 Eşik değer uygulanarak hesaplanan recall ve precision değerleri

Sorgu No	Precision %	Recall %
Sorgu 9	1	0,2
Sorgu 28	1	0,5
Sorgu 22	1	0.66
Sorgu 13, 17	1	0,75
Sorgu 12, 14, 15, 16, 21, 26, 30	1	0.8
Sorgu 1, 8	1	0,85
Sorgu [2-7], 10, 11, 18, 19, 20, 23, 24, 25, 27, 29	1	1
Ortalama	**100**	**87**



Şekil 5.7 Precision ve Recall değerleri

Çizelge 5.6 Farklı eşik değerlerindeki Precision ve Recall değerleri.

Eşik değeri	Precision %	Recall %
0,02	95,6	88,9
0,021	97,6	88,9
0,022	96,9	87,9
0,0225	98,6	87,9
0,025	99	88
0,03	100	87,4

BÖLÜM 6

SONUÇ VE ÖNERİLER

Bu çalışmada büyük sayıda resim kümesi içerisinde sorgulanan resme içerik olarak benzeyen resimleri bulma problemi için Büyük Veri teknolojileri ve birden fazla resim işleme algoritmalarının birlikte kullanılması önerilmiştir. Sonuçların makul bir zaman içinde dönmesi için kademeli aramanın indekslenmiş veriler içerisinde kullanılabilirliği incelenmiştir.

Veri seti internette bulunan haber sitelerinden sağlanmıştır. 20 haber sitesinin manşet haberleri günde 2 defa olmak üzere bir hafta boyunca taranmıştır. Taranılan haberlerde bulunan, haber ile ilişkili resimler Apache HBASE veri tabanına kaydedilmiştir. Bu şekilde yaklaşık 200bin resimlik bir veri seti oluşturulmuştur. Daha sonra kaydedilen resimlerin öz nitelikleri, MapReduce çatısı altında görüntü işleme algoritmaları yardımı ile çıkartılmıştır. Yapılan bu işlem MapReduce çatısı ile dağıtık mimaride görüntü işleme algoritmalarının kullanılabilir olduğu görülmüştür.

Kademeli arama mimarisi ile ilk etapta veri setinden alakasız resimler elenir. Sonraki etapta daraltılan veri seti daha başarılı görüntü işleme algoritması ile benzerliğe göre sıralanır. Bu noktada ilk eleme işlemi kullandığımız diğer algoritmalara göre çok daha hızlı olan Bulanık Renk Doku Histogramı (BRDH - FCTH) algoritması ile yapılmıştır. Bu aşamada 200bin resim 500 resme indirgenerek sorgulanan resim ile net olarak ilişkisi olmayan resimler elenmiştir.

Eleme işlemi sonrasında elde ettiğimiz 500 adet resim, ilk önce Hızlı Retina Anahtar Noktası (HRAN – FREAK) algoritması kullanılarak benzerliğine göre sıralanmıştır. Bu sıralamada doku olarak benzerliği yakalanamayan resimler Ölçekten Bağımsız Öznitelik Dönüşümü (ÖBÖD – SIFT) algoritması ile yeniden sıralanmıştır. Sıralama

sonrasında doku olarak benzerliği bulunamayan fakat renk dağılımında belirgin benzerlik bulunan resimlerde Renk Haritalama algoritması yardımı ile benzerlik sıralamasında yerini almıştır. Sıralanan 500 resmin ilk 60 resmi kullanıcıya sunulmuştur. Tüm bu arama işlemi, sadece ÖBÖD ile yapılınca 1 saatten uzun sürerken kademeli arama mimarisi ile yaklaşık 15 saniyede tamamlanmaktadır.

Sunulan resimlerden üst sıralarda bulunanlar, sorgulanan resme daha çok benzemesi beklenmektedir. Başarı ölçümü, beklenilen resimlerden kaç tanesinin ilk sıralarda yer aldığına bakılarak hesaplanmıştır. Yaptığımız deneylerde çalışmamızın başarısı eşik değeri uygulanmadığı durumlarda %92olarak hesaplanmıştır. Farklı eşik değerlerindeki sonuçlar tablo halinde verilmiştir. Precision değerinin %100 olduğu durumda Recall değeri %87 olarak hesaplanmıştır.

Tüm bu bilgiler ışığında Büyük Veri teknolojileri ile kademeli arama mimarisinde BRDH, HRAN, ÖBÖD ve Renk Haritalama algoritmaları büyük sayıda resim kümesi içerisinde sorgulanan resme içerik olarak benzeyen resimleri bulma problemi için tercih edilebilir olduğu görülmektedir.

Yapılan çalışmada yukarıda değinilen sonuçlar elde edilmekle beraber, çalışma sonrasında ileriye dönük üzerinde çalışılabilecek yeni konuların da ortaya çıktığı görülmüştür. Çalışmamız internette bulunan haber sitelerini kapsadığı için her resmin içerisinde bulunduğu ya da ilişkilendirildiği bir haber olmaktadır. Bu bilgi ile resim arama sırasında, aranılan resmin haber metni ile resim kümesindeki resimlerin haber metinlerinin benzerlik değerleri de arama sonucuna bir ağırlık olarak eklenmesi ileri dönük bir çalışma olarak ortaya çıkmıştır. Diğer çalışma alanları ise arama sırasında aranılan resimde insan yüzü geçiyor ise yüz tanıma algoritmalarının kullanılması ve resim üzerine eklenen yazıların tespit edilmesidir.

KAYNAKLAR

[1] Statistic Brain, Google Annual Search Statistics, http://www.statisticbrain.com/google-searches/, 30 Nisan. 2013

[2] Sparrow, B., Liu, J.,ve Wegner, D. M., (2011). "Google Effects on Memory: Cognitive Consequences of Having Information at Our Fingertips." Science,333: 776-778

[3] Dhale, V. D., Mahajan, A. R., ve Thakur, U. (2012). "A Survey of Feature Extraction Methods for Image Retrieval",International Journal of Advanced Research in Computer Science and Software Engineering Volume 2 Issue 10, 2: 1-8

[4] Eakins, J. P. ve Graham, M. E. (1999), "Content-based Image Retrieval: A report to the JISC Technology Applications Programme" , Technical report, Institute for Image Data Research, University of Northumbria at Newcastle .

[5] Yong, R., Huang, T. S.,ve Chang, S. F., (1999). "Image Retrieval: Current Techniques, Promising Directions, and Open Issues", Journal of Visual Communication and Image Representation, 10: 39-62

[6] Bay, H., Tuytelaars, T. ve Gool, V., (2006). "Speeded-Up Robust Features(SURF)",Computer Vision and Image Understanding, 110: 346-359.

[7] Lowe, D., (2004). "Distinctive Image Features from Scale-Invariant Keypoints.", Internation Journal of Compututer Vision,60: 1–28.

[8] Yin, D., ve Liu, D.,(2013). "Content-Based Image Retrial Based on Hadoop",Mathematical Problems in Engineering, 2013: 7

[9] Raju U.S.N., (2015)."Content-Based Image Retrial Based on Hadoop", Big Data (BigData Congress), 2015 IEEE International Congress, 1: 661-664

[10] Youssef, S.M., (2012). "An efficient content-based image retrieval system integrating wavelet-based image sub-blocks with dominant colors and texture analysis", Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference, 3: 518-523

[11] Gu, C., ve Gao, Y.,(2012). "A Content-Based Image Retrieval System Based on Hadoop and Lucene", Cloud and Green Computing (CGC), 2012 Second International Conference, 1: 684-687.

[12] Heczko, M., Hinneburg, A., Keim, D., ve Wawryniuk, M., (2004). "Multiresolution similarity search in image databases", Multimedia Systems, 10: 28-40

[13] Wang, X., Wu, J. ve Yang, H., (2010).“Robust image retrieval based on color histogram of local feature regions”, Multimedia Tools Applications, 49: 323–345.

[14] Hazra, D., (2011). “Texture recognition with combined GLCM, wavelet and rotated wavelet features”, International Journal of Computer and Electrical Engineering, 3: 146-150.

[15] Harris, C. ve Stephens, M., (1988).“A combined corner and edge detector.”, Alvey Vision Conference, 147–151.

[16] Mikolajczyk, K.,ve Schmid, C., (2001).“Indexing based on scale invariant interest points”, International Conference on Computer Vision, 1: 525-531

[17] Mikolajczyk K., ve Schmid, C., (2003). “Aperformance evaluation of local descriptors.”, IEEE Conference on Computer Vision and Pattern Recognition, 2: 257-263

[18] Ke, Y.,ve Sukthankar, R., (2004). “PCA-SIFT: A More Distinctive Representation for Local Image Descriptors.”, IEEE Conference on Computer Vision and Pattern Recognition, 2: 506-513

[19] Leutenegger, S., Chli, M. ve Siegwart, R. Y., (2011). “BRISK: Binary robust invariant scalable keypoints”, IEEE International Conferance of Computer Vision, 2548–2555.

[20] Alahi, A., Ortiz, R.ve Vandergheynst, P., (2012). “FREAK: Fast retina keypoint”, IEEE Conference CVPR, 510–517.

[21] Calonder, M., Lepetit, V., Strecha, C. ve Fua, P., (2010).“Brief: Binary robust independent elementary features.”, European Conference on Computer Vision, 4: 778-792.

[22] Rublee, E., Rabaud, V., Konolige, K. ve Bradski, G., (2011) “ORB: An efficient alternative to SIFT or SURF,” IEEE International Confenrence of Compututer Visison, 2564–2571.

[23] Chatzichristos, A.S.ve Boutalis, S. Y., (2008).“FCTH:Fuzzy Color and Texture Histogram - A Low Level Feature for Accurate Image Retrieval.” International Workshop on Image Analysis for Multimedia Interactive Services, 1: 191-196.

[24] Ghemawat, S., Gobioff, H.,ve Leung, S. T., (2003).“The Google File System”, SOSP, 37: 29-43

[25] Dean, J.,ve Ghemawat, S., (2004).“MapReduce: Simplified Data Processing on Large Clusters”, Communications of the ACM, 51:107-113

[26] Baldeschwieler, E., (2008).Yahoo! Launches World’s Largest Hadoop Production Application, https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html, 11 Ekim 2015

[27] White, T., (2012), Hadoop - The Definitive Guide, O'reilly, 3. Baskı

[28] Holmes. A., (2012), Hadoop in Practice, Manning.

[29] Hortonworks, HDFS (Hadoop Dıstrubuted File System), http://hortonworks.com/hadoop/hdfs/#section_1, 11 Ekim 2015

[30] Apache, Apache HBase Projesi, http://hbase.apache.org/, 13 Eylül 2015

[31] Wikipedia, MapReduce Çatısı, https://en.wikipedia.org/wiki/MapReduce/, 9 Haziran 2014

[32] Apache, Apache Lucene Projesi, http://lucene.apache.org/, 13 Eylül 2015.

[33] Khare, R., Cutting, D., Sitaker, K. ve Rifkin, A. (2004).“Nutch: A Flexible and Scalable Open-Source Web Search Engine.”, CommerceNet Labs, 1: 0-16

[34] Apache, Apache NutchProject, http://nutch.apache.org/, 13 Eylül 2015.

[35] Apache Nutch, FAQ Nutch Wiki, https://wiki.apache.org/nutch/FAQ/, 10Eylül 2015

[36] Büttcher, S., Clarke, C. L. A., ve Cormack, G. V., (2010). Information Retrieval: Implementing and Evaluating Search Engines, MIT Press, New York.

[37] Yates, R. B., ve Neto B. R., (2011), Modern Information Retrieval: The Concepts and Technology Behind Search, Addison-Wesley.

[38] Croft, B., Metzler, D.,ve Strohman, T.,(2010), Search Engines: Information Retrieval in Practice, Addison-Wesley.

[39] Chatzichristofis, S.,ve Boutalis, Y., (2007). “A Hybrid Scheme for fast and accurate image retrieval based on color descriptors”, IASTED International Conference on Artificial Intelligence and Soft Computing,1: 280-285.

[40] Zimmerman, H.J., (1987), Fuzzy Sets, Decision Making and Expert Systems, Kluwer Academic Publication, Boston MA

[41] Konstantinidis, K., Gasteratos, A., ve Andreadis, I., (2005). “Image Retrieval Based on Fuzzy Color Histogram Processing”, Optics Communications, 248: 375-386.

[42] Wang, J. Z., Li, J.ve Wiederhold, G., (2001). “SIMPLIcity: Semantics-Sensitive Integrate, Matching for Picture Libraries”, IEEE transactions on pattern analysis and machine intelligence, 23: 947-963.

[43] Gustafson, E. E., Kessel, W. C., (1979).“Fuzzy Clustering with a Fuzzy Covariance Matrix”, IEEE CDC, 1: 761- 766.

[44] Mair, E., Hager, C. D., Burschka, D., Suppa, M. veHirzinger, G., (2010). “Adaptive and generic corner detection based on the accelerated segment test.” European Conference on Computer Vision(ECCV),1: 183-196

[45] Sharma, G., Wu, W.ve Dalal, E. N., (2005). “The CIEDE2000 color-difference formula:Implementation notes, supplementary test data, and mathematical observations”, Color Research and Application, 30: 21–30.

[47] Apache, Apache Solr Project, http://lucene.apache.org/solr/, 22 Mart 2014.

[46] Rubner, Y., Tomasi, C.veGuibas, L. J., (2000). “The earth mover’s distance as a metric for image retrieval.”, International Journal of Computer Vision, 40: 99-121

[48] Rosten, E.,ve Drummond, T., (2006).“Machine Learning for High-Speed Corner Detection.” European Conference on Computer Vision, 1: 430-443

[49] Hitchcock, F. L., (1941). “The distribution of a product from several sources to numerous localities.” Journal of Mathematics and Physics, 20:224-230.

[50] Wikipedia, Precission and Recall, https://en.wikipedia.org/wiki/Precision_and_recall, 22 Ekim 2015.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Mehmet Zahid YÜZÜGÜLDÜ

Doğum Tarihi ve Yeri : 1984 / Konya

Yabancı Dili : İngilizce

E-posta : zyuzuguldu@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Mühendisi	Atılım Üniversitesi	2009
Lise	Fen Bilimleri	Konya Meram Anadolu Lisesi	2004

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2009	THY Teknik A.Ş.	Yazılım Uzmanı
2013	TÜBİTAK	Araştırmacı

Proje

1. Ulusal Arama Motoru / TÜBİTAK

2. Bulut Bilişim ve Büyük veri Analizi / TÜBİTAK