ÖZET

Yüksek Lisans Tezi, Nesne Tanıma ve Aralarındaki Topolojik İlişkilerin Semantik Düzeyde İncelenmesi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı

Son zamanlarda teknolojinin gelişmesiyle fazlalaşan dijital görüntülere istenildiğinde erişebilmek için görüntülerin belli özelliklere göre analiz edilmeleri ve sınıflandırılmaları gerekmektedir. Ayrıca yapay zeka alanındaki gelişmeler robot endüstrisindeki son yıllarda gören ve nesneleri tanımaya çalışan robotlar üzerindeki çalışmalara yol göstermiştir. Görüntü analizi yapabilen akıllı sistemler ve robotlar geleceğin teknolojileri olarak kabul edilmektedirler.

Bu tezde görüntü tanımından itibaren ele alınmış, işleme aşamaları, sınıflama yapmada ve akıllı görme sistemlerinde kullanılan nesne tanıma sistemleri incelenmiştir. Nesnelerin farklı özelliklerine tanıma yapan sistemlerden örnekler verilmiş, nesneler arası topolojik ilişkiler ve bunların mantıksal ve semantik olarak ifadeleri incelenmiştir. Temel geometrik şekiller üzerinde sınıflama yapan bir de uygulama geliştirilmiştir.

Bu tez 2005 yılında yapılmıştır ve 70 sayfadan oluşmaktadır.

ANAHTAR KELİMELER: Görüntü işleme, nesne tanıma, bilgisayarla görme, şekil tanıma, topolojik ilişkiler.

ABSTRACT

M.Sc. Thesis, “Object Recognition and Determining the Topological Relations in Between Semantically”, Trakya University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering.

To access the digital images which are increased in a great amount in the last years with the growth of the technology when needed, they have to be analysied regarding some of their features and classified. Besides, the development in artificial intelligence has led to the intensive research on robots having ability to see and recognize the objects around them in robotics industry. Smart systems and robots which are capable of doing image analysis are accepted as the technologies of the future.

In this thesis, image concept is held from description, processing techniques, object recognition systems used in classification and smart vision systems are examined. Recognition systems using different spesifications of the objects are exampled, topological relations between objects and their logical and semantic models are examined. Also, a basic geometrical shape classification application is implemented.

This thesis is done in 2005 and consists of 70 pages.

KEYWORDS: Image processing, object recognition, computer vision, shape recognition, topological relations.

TEŞEKKÜR

Bu çalışmamı hazırlamamda benden yardımlarını esirgemeyen danışman hocam Dr.Cavit Tezcan'a, her yönüyle mükemmel bir insan ve değerli bir hoca olan ve bana her zaman destek olan Dr.Yılmaz Kılıçaslan'a, yazılım konusunda bana çok yardımcı olan, iyi bir mühendis ve arkadaş, Ersin Aksoy'a ve bir ömür her zaman yanımda olacağını bildiğim sevgili Dr. Serkan Gökçay'a ve aileme çok teşekkür ediyorum.

İÇİNDEKİLER

1. GİRİŞ

Nesne tanıma sistemleri içinde barındırdıkları veritabanlarındaki modeller sayesinde gerçek dünyadaki bir görüntüden yola çıkarak, nesneleri tanımaya çalışan sistemlerdir. Nesne tanıma problemi önceden bilinen nesne modelleri üzerine kurulu bir etiketleme sorunu olarak da yorumlanabilir. Nesne tanıma sorunu ayrıştırma sorunu ile doğrudan bağlantılıdır. İyi bir ayrıştırma yapmadan iyi bir tanıma sistemi oluşturulamaz. Ayrıştırma konusu da görüntü işlemenin bir üst konusudur. Nesne tanıma sistemleri görüntü ayrıştırmada görüntü işleme tekniklerine de ihtiyaç duyacağından nesne tanıma konusu alt başlıklarıyla birlikte ele alınmıştır. Tezin kapsamında görüntü işleme, nesne ayrıştırma ve nesne tanımlama problemleri ve çözümleri incelenmiştir. Bir görüntü işleme ve ayrıştırma teknikleri uygulaması ile temel geometrik şekil tanıma programı geliştirilmiştir.

Son yıllarda, gelişen teknolojiye paralel olarak akıllı sistemlerin endüstride kullanılmaya başlanmasıyla birlikte bu konuya olan ilgi oldukça artmış ve yapay görme sistemleri uygulamaları her alana yayılmaya başlamıştır. Nesne tanıma sistemleri yapay görme sistemlerinin alt bir koludur ve tıp, güvenlik, tarım gibi alanlarda kullanılan sistemler geliştirilmeye başlanmaktadır. Tez içerisinde farklı alanlarda kullanılan nesne tanıma örnekleri araştırılmıştır.

Ayrıca yapay zeka alanının alt alanlarından olan görüntü yorumlanması ve içeriğinin semantik ifadesine giriş olan, nesneler arası topolojik ilişkilerin modellenmesi konusu da incelenmiştir.

Tez toplam 5 bölümden oluşmaktadır.

İkinci bölümde temel görüntü işleme teknikleri ve dijital görüntülere ilişkin kavramlar üzerinde durulmuştur.

Üçüncü bölümde görüntü ve nesne tanıma sistemlerinin genel yapıları anlatılmış ve nesneler arası topolojik ilişkiler ve bunların semantik gösterimleri incelenmiştir.

Dördüncü bölüm uygulamaya ayrılmıştır. Temel görüntü işleme tekniklerini uygulayan bir program ve temel geometrik şekilleri tanıyan bir sistem geliştirilmiştir.

Beşinci bölümde yapılan çalışmanın sonuçları incelenmiştir.

2. GÖRÜNTÜ İŞLEMEDE TEMEL BİLGİ VE TEKNİKLER

2.1. Görüntü Nedir?

Görüntü, iki boyutlu ışık şiddeti fonksiyonudur. Bu fonksiyon f(x,y) şeklinde gösterilir. Burada x ve y uzaysal koordinatları, (x,y) noktasındaki f'nin sayısal değeri ise parlaklık değeri veya görüntünün ilgili noktadaki renk seviye değeridir. Herhangi bir (x,y) koordinatındaki f'nin genliğine, görüntünün o noktadaki yoğunluğu denir.

İnsan algılama sistemi; görüntü yakalama, gruplama ve analiz konusunda bilinen en karmaşık sistemdir. İnsan görme sistemi gözlerimizle başlar. Işığın çok kanallı dalga boyları her biri birer algılama sistemi olan gözlerimiz yardımı ile algılanır. Görülebilen spektrum tanımı; insan gözünün görebileceği elektro manyetik dalga boyu aralığını tanımlar. Buna karşın bir arının görebildiği spektral aralık ultraviyole bölgede başlar ve yeşil dalga boylarında sona erer.

Spektrum uzunluk ölçme birimleri ile ölçülebilen periyodik davranış sergileyen enerji dalgalarını temsil eder. Görülebilen alana ait dalga boyları 0.4μm-0.7μm arasındadır.

Gözlerimizle görülebilen alandaki elektro manyetik dalgaları algılayabiliriz ve beynimiz yardımı ile yorumlanabilir görüntü haline dönüştürebiliriz. Gözün ana bileşenleri; Kornea, göz bebeği, mercek, retina ve optik sinirlerdir.Kornea gözün dış kısmında olup geçirgen, kubbe formunda olup, ışığa odaklama fonksiyonuna sahiptir. Göz bebeği kendisini tutan kaslar yardımı ile ışık göze ulaştığında gözün açılıp kapanmasına yarar. Göz bebeği göz merceğini örter. Kaslar yardımı ile mercek göze giren ışığın şiddetine göre kalınlaşır veya incelir.

Gözlerin farklı kontrastlara adapte olabilme yeteneği parlaklık adaptasyonu olarak adlandırılır. İki parlaklık düzeyleri arasında ayırım yapabilme yeteneğine ise zıtlık duyarlılığı adı verilir. Bu da gözün etrafını çevreleyen parlaklık düzeylerine

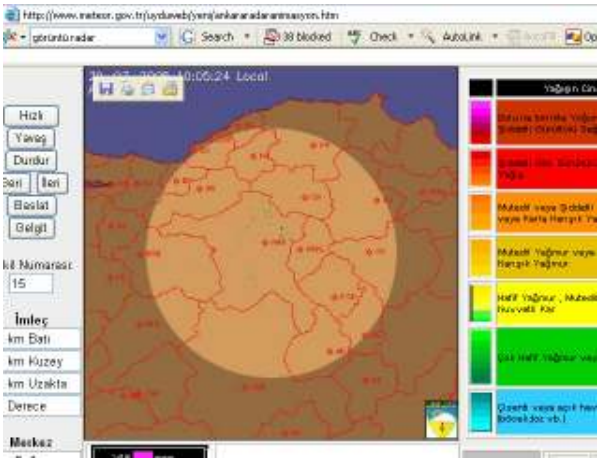
bağlıdır. Güneşli bir günde farları yanan bir aracın farlarını görmek güçtür, fakat gece değildir.

Özet olarak; sayısal görüntü işleme için görme sistemlerimizin altında yatan temel mekanizmaların bilinmesi oldukça önemlidir. Kısaca göz bir fotoğraf makinası gibi düşünülebilir ve beynin görme bölümleri de karmaşık bir sayısal görüntü işleme sistemi olarak düşünülebilir.

Görüntü işleme yaşam var oldukça söz konusu olmuştur. İnsanlar ve hayvanlar gözleri ile analog temele dayanan görüntü işleme yapmaktadırlar. Bu olay beyin yardımı ile (akıllı sistem) gerçek zamanlı, paralel ve çok spektrumlu olarak oluşmaktadır.

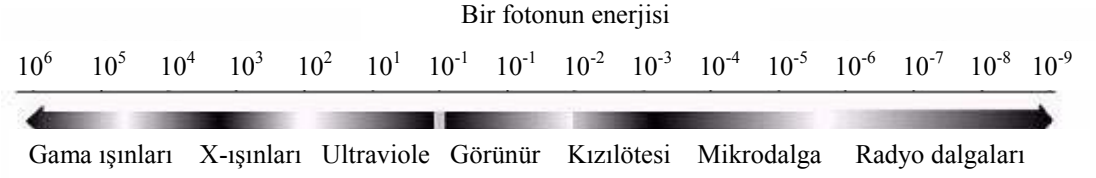
2.2. Görüntü Elde Etme

Gelişen teknoloji ile birlikte bugün pek çok farklı kaynaktan görüntü elde etmek mümkün olmuştur. Elde edilen görüntülerin elde edilme amaçlarına uygun olarak kullanılmaları için dijital görüntülere dönüştürülmeleri yani sayısallaştırılmaları gerekmektedir. Bu kaynaklara örnek olarak radar görüntüleri, sismik görüntüler, tıp alanında MR görüntüleri, ultrason görüntüleri örnek gösterilebilir. Aşağıdaki şekilde www.meteor.gov.tr sitesinden alınan, renklendirilerek anlaşılır hale getirilmiş ve hava durumunu gerçek zamanlı olarak takip etmek için geliştirilmiş sistemden alınan bir örnek görülmektedir.

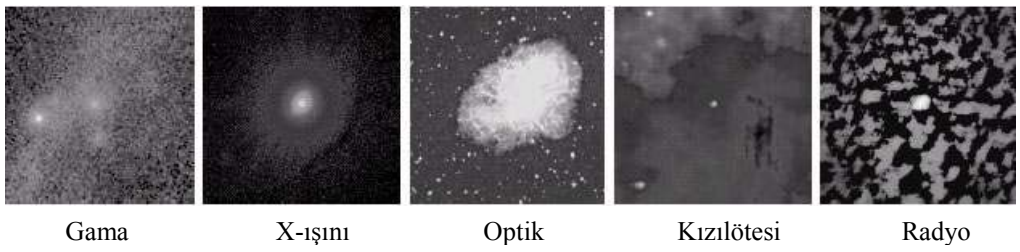


Şekil 2.1. Anlık hava durumunu gösteren radar görüntüsü

Özellikle uzaktan algılama sistemleri konusunda başarılı sonuçlar veren projeler günümüzde uzay araştırmalarında, hava ve deprem tahmin raporlarında, ulusal güvenlik amaçlı olarak kullanılmaktadır. Bu görüntülerin yakalanması elektromanyetik spektrum üzerindeki dalga boylarının farklı olması sayesinde mümkündür ve farklı amaçlar için kullanılabilirler.

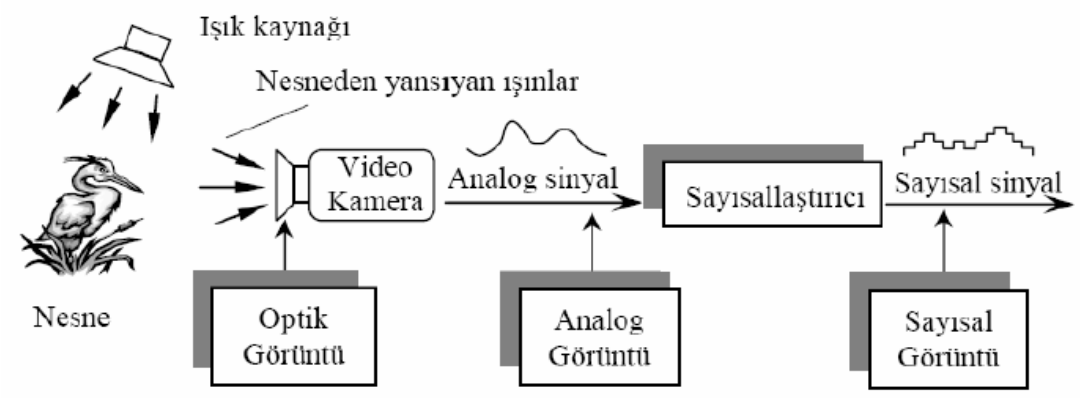


Şekil 2.2. Elektromagnetik spektrum



Şekil 2.3. Aynı kaynaktan (Yengeç Takımyıldızı) farklı ışınlarla elde edilen görüntüler

Fiyatları bol sıfırlı rakamlarla ifade edilen çeşitli optik ekipmanlarla donatılmış cihazlar sayesinde elde edilen bu görüntülerin anlamlandırılması aşamasında mutlaka sayısal sinyallere dönüştürülmeleri gerekmektedir. Ancak bu sayede hava durumunu gösteren şekilde olduğu gibi daha anlaşılabilir, daha önemlisi ve bu tezin konusunu da olan istenen nesnelerin sistem tarafından tanınması mümkün olacaktır.



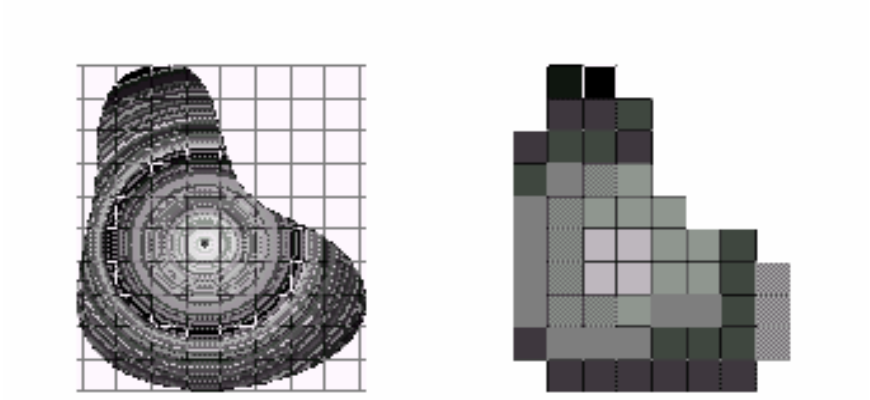
Şekil 2.4. Görüntü yakalama ve sayısallaştırma aşamaları

Bir görüntü fonksiyonunu, f(x,y), bilgisayarda işlemeye uygun hale getirebilmek için yani görüntüyü dijitalleştirmek için fonksiyonu hem uzaysal koordinatlar olarak, hem de genlik olarak sayısallaştırmak gerekir. Uzaysal koordinatların sayısallaştırılmasına örnekleme ve genliğin sayısallaştırılmasına da niceleme denir. Bu ifadeye Shanon'un Örnekleme ve Niceleme Teoremi de denir. Sayısal görüntüde normal fotoğrafik görüntü öncelikle bireysel parlaklık değerlerine sahip noktalara bölünmelidir. Görüntünün sayısallaştırılması, kameradaki görüntünün optik-elektrik mekanizma ile elektriksel sinyallere dönüştürülmesi işlemidir.

Resim fonksiyonu f(x), konum ve yoğunluk göz önüne alındığında sürekli bir fonksiyondur. Sayısallaştırma aşamasında tarama; resim elemanlarının (piksel) konumlarının belirlenmesidir. Değer atama, yoğunlukların belirlenmesi aşamasıdır. Yoğunluk dizisi(IG) doğal sayılar kümesinin elemanlarından oluşur.

Örneğin; IG = 0, 1, 2, … 255 = 8 bit

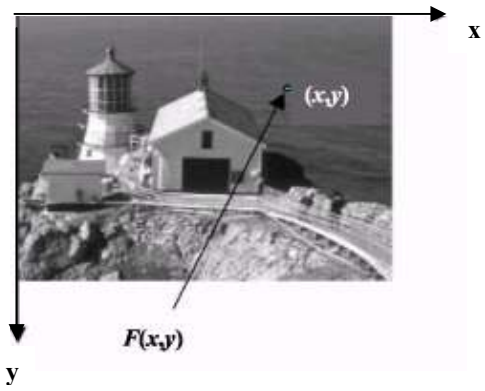
Analog bir kaynaktan elde edilen görüntü fonksiyonunun sayısal hale dönüştürülmesi resmin taranması ve değer atanması aşamalarının sonucudur.



Şekil 2.5. Örnekleme ve niceleme sonucunda sayısal görüntü

2.3. Sayısal Görüntüler

Bir sayısal görüntü genellikle dikdörtgen şeklinde piksel serisinden oluşacak biçimde örneklenir. Her pikselin görüntü üzerinde belirli bir koordinatı, (x, y), vardır.



Şekil 2.6. Sayısal görüntü temsili ve eksenleri

Bir sayısal görüntü, satır ve sütun indisleri görüntü içerisinde herhangi bir noktayı tanımlayan elemanlardan meydana gelmiş bir matris olarak göz önüne alınabilir. Bu matrisin her bir elemanının sayısal değeri, kendisine karşılık gelen noktalardaki renk seviye değerine eşittir. Bu sayısal dizinin veya matrisin her bir elemanına görüntü elemanı, resim elemanı veya piksel denir.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Şekil 2.7. Görüntü piksellerinin matris olarak temsili

2.4. Görüntü İşleme

Görüntü işleme, genel terim olarak resimsel bilgilerin değiştirilmesi ve analizi demektir. Yada sayısallaştırılmış görüntülerin bilgisayarlar aracılığıyla işlenmesi olarak da ifade edilebilir.

Sayısal görüntü elde edildikten sonra, diğer adım ön işleme işlemidir. Bu aşamada, alınan görüntü bir sonraki aşamada hatasız ve kolay işlenebilmesi için daha belirgin ve anlaşılır hale getirilir. Bu işlemlerden bazıları:

- Görüntüyü belirginleştirmek
- Görüntüde bulunan kirlilikleri filtrelemek
- Görüntü üzerindeki yapısal bozuklukları yok etmek veya minimize etmektir.

Sayısal görüntü işleme temelde parlaklık, kontrast, renk vb. görüntüye ait bilgilerin değiştirilmesi; manyetik alan, görüntüleme sırasında hatalı donanım ayarlarının kullanılması vb. nedenlerle oluşan görüntü kirliliklerin (gürültü) giderilmesi, detayların daha belirgin hale getirilmesi gibi görüntü kalitesinin iyileştirilmesine yönelik işlemleri ifade eder ve sonuçta yeni bir görüntü elde edilir. Sayısal görüntü analizinde ise genellikle yapılan işlemler sonucunda yeni bir görüntü elde edilmez, ancak görüntüye ait sınıflandırmalar yapılabilir, görüntüyle ilgili istatistikler üretilir. Sayısal görüntü analizinde nesnelere ait parametrelerin (şekil, uzunluk, alan, açı, nisbî konum, dokusal yapı, gri-ton değeri, RGB renk değerleri vb.) ölçülmesi söz konusudur.

Pek çok farklı görüntü işleme tekniği bulunmasına rağmen burada örnek olarak sadece bir filtreleme ele alınmıştır. Sayısal görüntü işleme başlı başına çok kapsamlı bir konudur.

Görüntü işlemede temel adımlar şunlardır:

- Görüntü edinme
- Görüntü iyileştirme- filtreleme
- Görüntü onarma
- Renkli görüntü işleme
- Dalgacıklar
- Sıkıştırma
- Morfolojik işlemler
- Ayrıştırma

- Gösterim ve tanımlama

Resim iyileştirme konuları; bilinen veya önceden bulunan istatiksel veya geometrik oluşum modelleri tarafından deforme olmuş resmi tamir etmekte kullanılan teknikleri içerir. Biçimi bozulmuş resmi tamir etmek ve resim üzerindeki gürültüleri filtrelemek iyileştirme uygulamalarına örnektir.

Resim arttırımı konuları; genellikle insan görüşü için en iyi görüntülü resme sahip olmak için kullanılır. Örnek olarak zıtlık germe bir arttırım uygulamasıdır. Değişiklik yapmadan her tür resmin değerini arttırmakta kullanılan genel arttırım teknikleri yoktur. Örnek olarak uydu resmi ile tıbbi resim için uygulanan teknikle istenilen sonuç elde edilemeyebilir.

Resim iyileştirme ve arttırımı konularının her ikisi de aynı amaca sahiptir. Her ikisinin de amacı orjinalden daha iyi resme sahip olmaktır. Fakat onların resme yaklaşım tarzları farklıdır. İyileştirme teknikleri bozulmaya uğramış resim ile ilgilenir ve arttırım teknikleri görsel görünüm ile ilgilenir.

Resim sıkıştırma ise önemli bilgiyi kaybetmeden resim verisini azaltmayı amaçlar. Amaç için gereksiz bilgi varsa elenir. Ne kadar verinin gerekli olduğuna karar vermede insan görüşü rol oynar. Bu yüzden resim sıkıştırma tekniği de resim işleme konularına girer.

2.4.1. Görüntü filtreleme

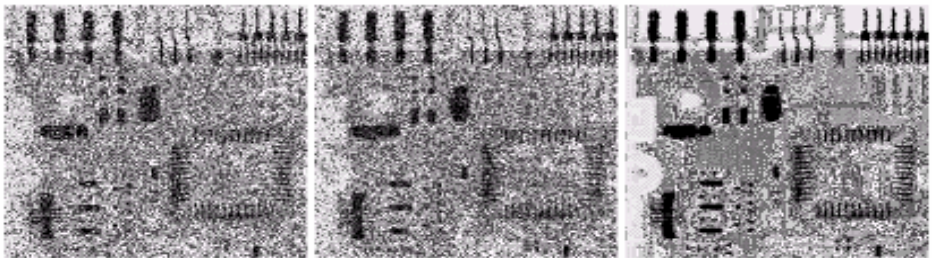
Konumsal filtreler gürültüleri (rahatsız edici faktör) filtrelemek için etkin bir şekilde kullanılırlar. Sıralı ve ortalama olmak üzere iki türlüdürler. Sıralıda, seçilen komşulukta gri renk tonları artan bir şekilde sıralanır. Bu sıradan gürültü olabilmesi için en küçük olasılığa sahip piksel seçilerek bu pikselin merkezine seçilen değer atanır. Ortalama filtrelerde ise, belirlenen ortalama kritere göre ortalama değer komşu piksellerden hesaplanır ve pikselin merkezine değer olarak atanır. Sıralı filtreler genellikle tuz ve biber gürültüleri, negatif üssel gürültüleri ve Rayleigh dağılımlı

gürültüleri ortadan kaldırmak, ortalama filtreler ise düzgün dağılımlı ve normal dağılımlı gürültüleri ortadan kaldırmak için kullanılırlar. En etkili sıralı filtre ortanca filtresidir. Bu filtre ile seçilen komşulukta gri renk değerleri artan bir biçimde sıralanır.

5	5	6			5 5 6
3	4	5	(3,3,4,4,**5**,5,5,6,7)	→	3 5 5
3	4	7			3 4 7

Şekil 2.8. Ortanca filtresi matris gösterimi

Aşağıdaki görüntü "Mr. Joseph E. Pascente.Lixi.Inc." ten alınmıştır.



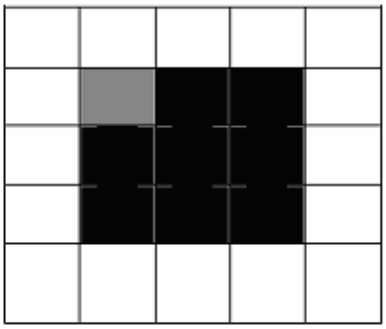
a) Tuz ve biber gürültülü devre şeması b) 3x3 ortanca maskesi ile gürültü azaltma c) 5x5 ortanca maskesi ile gürültü azaltma

Şekil 2.9. Ortanca filtresi uygulama

2.4.2. Gri düzey

Görüntü üzerindeki aydınlatma değerlerinin farklı seviyelerde olması, piksel düzeylerinin farklı olmasındandır. Bu şekilde ifadelerde görüntü siyah-beyaz renk tonlarından meydana geliyorsa, görüntü üzerindeki her bir nokta gri-düzey cetveli üzerindeki renk değerleriyle ifade edilir. Görüntü üzerindeki noktalar farklı olduğundan, her bir aydınlatma düzeyi için gerekli bitlerin yerleşimi farklıdır

Dört bitlik yani 16 farklı gri-ton aydınlanma değeri için her bir pikselin üzerinde bulunacak gri-seviye parlaklık değeri şu şekildedir.



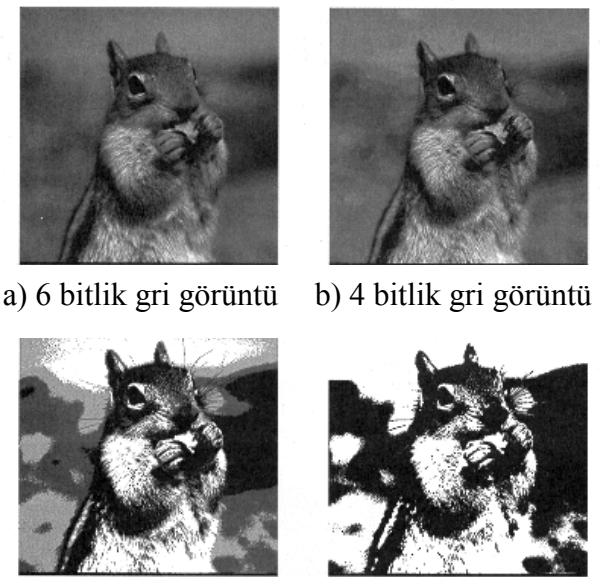
15	15	15	15	15
15	7	0	0	15
15	0	0	0	15
15	0	0	0	15
15	15	15	15	15

Şekil 2.10. 16 bitlik gri-düzey ifadesi

Değişik düzeylerin oluşturduğu görüntüler, gri-düzey veya gri-düzey cetveli ile ifade edilebilirler. Çünkü gri düzey görüntülerle işlem yapmak renkli görüntülerle olduğundan çok daha kolaydır. Piksel başına düşen bit sayıları; burada 4 bit/piksel yani bir pikselin değerini belirtmek için 4 bit kullanılmıştır. 0'dan 15'e kadar 16 gri-düzey değerleri mevcuttur.

Renkli görüntüler RGB adı verilen 24 bitlik renk uzayıyla ifade edilirler. RGB görüntüyü gri ölçekli görüntüye çevirmek için aşağıdaki denklem kullanılır.

Gri ölçek yoğunluğu=0.333R+0.333G+0.333B



a) 6 bitlik gri görüntü b) 4 bitlik gri görüntü

c) 2 bitlik gri görüntü d) 1 bitlik gri görüntü

Şekil 2.11. Aynı görüntünün farklı bit sayılarıyla ifadesi

2.4.3. Temel gri seviyesi dönüşümleri

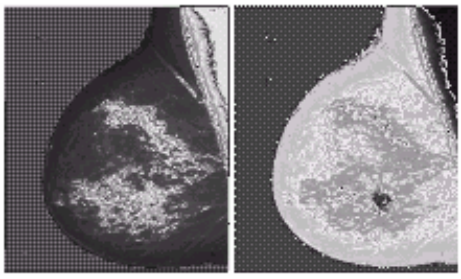
- Resmin negatifinin alınması
- Logaritmik dönüşümler
- Üstel fonksiyonlar

2.4.3.1. Resmin negatifi

Gri düzey görüntülerde görüntünün negatifinin alınması 8 bitlik gri görüntülerde her pikselin değerinin 255 ten çıkarılması ile olur. Formülü;

$$F(x)= 255- x$$

şeklindedir.



a) orijinal resim b) negatif resim

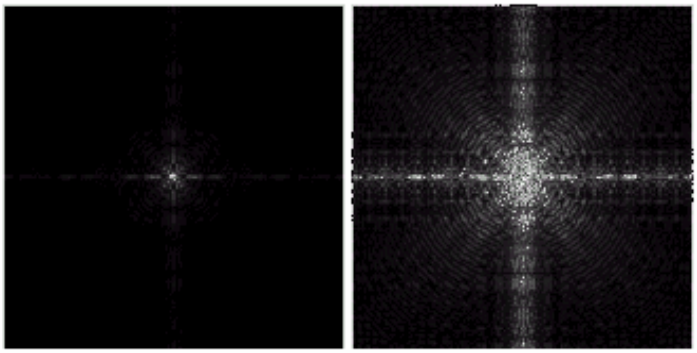
Şekil 2.12. Resmin negatife dönüşümü

2.4.3.2. Logaritmik Dönüşümler

Logaritmik dönüşümler, görüntünün piksellerinin parlaklık değerlerine göre farklı görüntülere dönüştürülmesidir. Formülü;

$$s = c- \log (1 + r)$$

şeklindedir.

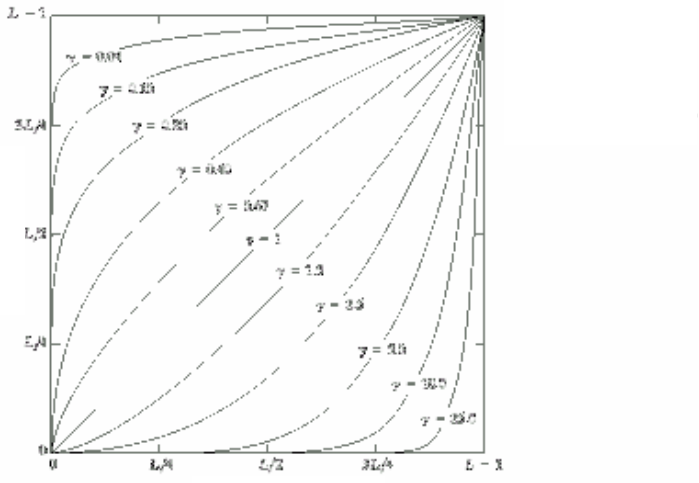


a) Fourier dönüşümü **b)** c=1 için log dönüşümü

Şekil 2.13. Farklı logaritmik dönüşümler uygulanan resim

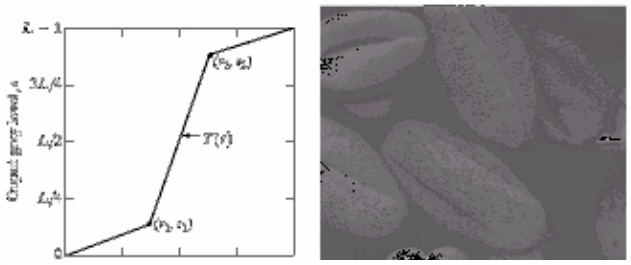
2.4.3.3. Üstel fonksiyonlar

Gri düzey görüntülerin üstel fonksiyonları her pikselin gama değeri adı verilen bir değerle üstsel olarak çarpılması sonucu hesaplanır. Gama düzeltimi adı verilen bu teknik, resmin açık yada koyu gri ton seviyelerinin değiştirilmesine izin verir.

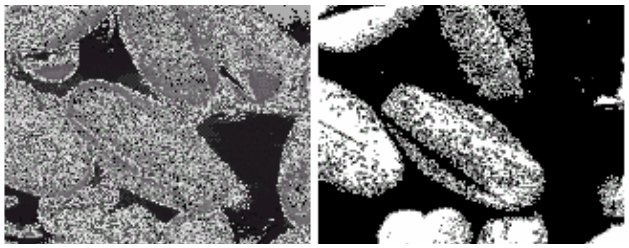


Şekil 2.14. Gama değerlerinin değişim grafiği

Parçalı-doğrusal dönüşüm fonksiyonları olarak belirtilen bu işlemler zıtlık artırma ve bit düzlemine göre dilimlemedir. Her biriyle ilgili örnekler aşağıda gösterilmiştir**.**

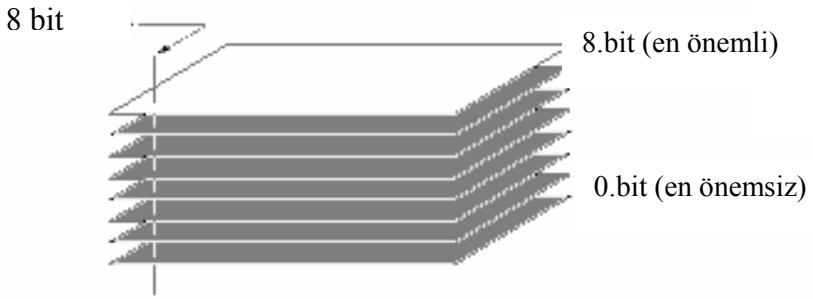


a) Dönüşüm fonksiyonu b) Orijinal resim

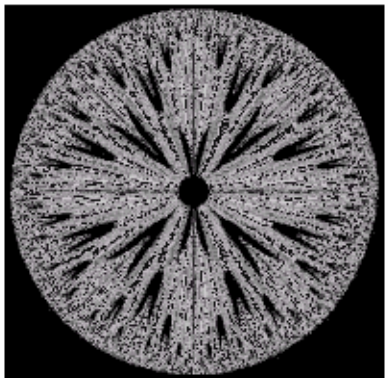


c) Zıtlık germe sonucu oluşan resim d) Tam zıtlık oluşturma

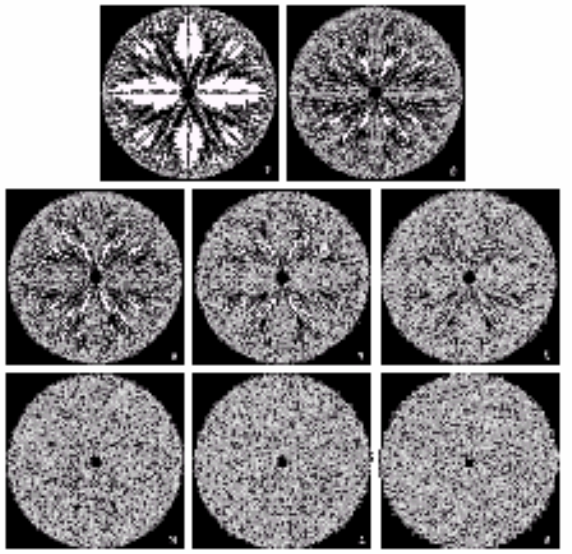
Şekil 2.15. Zıtlık artırma aşamaları



Şekil 2.16. 8 bitlik bir görüntünün bit tabakaları gösterimi



Şekil 2.17. 8-bitlik örnek görüntü



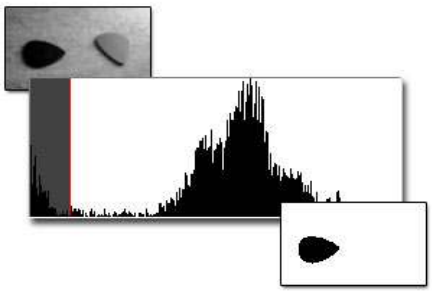
Şekil 2.18. Her bir bitin değişmesiyle ortaya çıkan görüntüler

2.5. Ayrıştırma

Ön işlemeden yani sayısal görüntü işleme teknikleriyle görüntünün daha net ve varsa gürültülerden arınmış hale getirilmesinden sonraki işlem ise görüntüyü, kendisini meydana getiren alt görüntülere parçalama, ayırma yada ayrıştırma işlemidir. Ayrıştırma işlemi görüntü işlemenin üst kısımlarından kabul edilir. Detaylı görüntü ayırma işlemleri, görüntü işlemede en zor işlemlerden sayılır. Bu nedenle genellikle küçük hatalarla birlikte kaba görüntü ayırma işlemleri uygulanır.

Görüntü dosyalarının içindeki nesnelerin analizinde nesneler ve "geri kalanlar"ın ayrılması önemli bir işlemdir. Burada geri kalanlar olarak tasvir ettiklerimiz arka plan olarak adlandırılır. En genel ve yaygın iki ayrıştırma tekniği eşikleme ve kenar bulmadır. Her bir teknikle yapılan ayrıştırma işleminin sonucunun kalitesini daha fazla yükseltmek amaçlanmaktadır.

Burada önemli iki nokta; tüm görüntüler için tek bir geçerli uygulanabilir nesne ayırma tekniği yoktur ve hiçbir nesne ayırma tekniği de tek başına mükemmel sonuç vermez.



Şekil 2.19. Histograma dayalı ayrıştırma

Şekil 2.19 üzerinde görüldüğü gibi histogramın sol üstünde bulunan orjinal görüntü içinden sol taraftaki küçük parçayı ilgilendiğimiz "nesne" , geri kalanları ise "arka plan " olarak tanımlayabilmek için uygun bir ayrıştırma işlemiyle istediğimiz sonuca ulaşabiliriz.

Ayrıca tek bitlik görüntüler üzerinde işlem yapmak 8, 24 yada 32 bitlik görüntüler üzerinde işlem yapmaya nazaran çok daha kolaydır. İyi yapılmış bir ayrıştırma işlemi daha sonra nesneleri ve nesnelerin şeklini tanımada çok kolaylık sağlayacaktır.

Ayrıştırma işleminin en çok kullanılan iki tekniği eşikleme ve kenar çıkarmadır.

Geliştirdiğimiz program içinde nesnelerin kenarlarını çıkarmakta kullandığımız kenar çıkarıma tekniğine bu bölümde değinilmiştir. İlgili bölüm içinde tekrar bu kısma referans verilecektir.

2.5.1. Eşikleme

Eşikleme işlemi, görüntü işlemenin önemli işlemlerinden biridir. Eşikleme renkli yada gri resimleri tek bite indirgeyerek nesneler ve arka plan arasındaki farkı açıkça belirleme işlemidir. Bu işlem, görüntü içinde her pikseli değerine göre beyaz yada siyah olarak belirler. Belirli bir pikselin siyah veya beyaz değer almasına karar verme eşikleme olarak adlandırılır. Uygulanması en basit tekniklerden biridir. θ parametresi

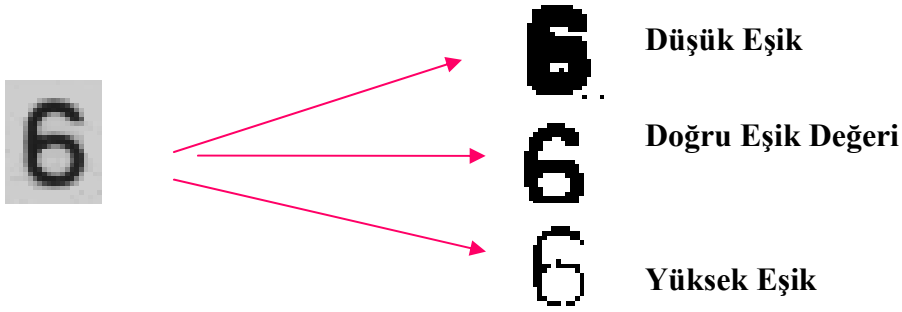
olarak parlaklık eşik değeri belirlenir ve görüntü üzerindeki her bir piksel değeri a[m,n] için aşağıdaki formül uygulanır:

Eğer a[m,n]> 0 ise a[m,n] = nesne = 1
Değil ise a[m,n] = arkaplan = 0

Bu algoritma koyu arkaplan üzerindeki açık renkli nesnelerle çalışıyorsak geçerlidir. Açık arkaplan üzerindeki koyu nesneler için aşağıdaki algoritma doğru sonuç verecektir.

Eğer a[m,n]< 0 ise a[m,n] = nesne = 1
Değil ise a[m,n] = arkaplan = 0

Eşikleme algoritmamız çıktı olarak nesne ya da arka planın ikili değerler olan 1 ve 0'a dönüşmüş halini verecektir. Eşikleme yaparken temel sorumuz eşik değerini nasıl seçeceğimizdir. Eşik değer seçmek için tüm görüntüler için genel tek bir yöntem olmadığı unutulmamalıdır, pek çok farklı alternatifler vardır.



Şekil 2.20. Örnek eşikleme

2.5.2. Kenar çıkarma

Bir resmin kenarları o resim hakkında çok fazla bilgi tutar. Kenarlar; nesnelerin nerede olduklarını, onların şekillerini, büyüklüklerini ve özellikleri hakkında bilgiyi

içerir. Bir resmin yoğunluğunun düşük seviyeden yüksek seviyeye çıktığı yer bir kenardır. Kenar çıkarımı için çeşitli özel efektleri kullanan uygulamalar vardır.

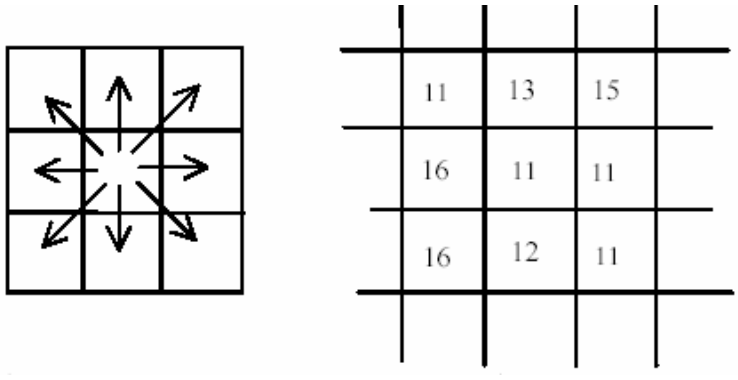
Kenar çıkarım işleyicinin sonuçları kenarları daha da belirginleştirmek için orijinal resme tekrar geri eklenebilir.

Kenar belirleme resim kesimlemenin ilk adımıdır. Resim kesimleme bir resmin oluşumlarını belirlemek için pikselleri alanlara gruplayan resim analizi safhasıdır.

Resim kesimlemenin yaygın bir örneği resim işleme yazılımlarındaki Magic Wand'tır. Bu araç kullanıcıya bir piksel seçme imkanı verir. Bu araç benzer değerlerin piksellerinin etrafını çerçeve ile çizer. Kullanıcı gökyüzünden bir piksel seçebilir ve Magic Wand aracı resimdeki tüm gökyüzünün etrafına bir sınır çizer. Böylece kullanıcı resimdeki diğer nesnelerin rengi değişecek endişesi olmadan gökyüzünün rengini değiştirebilir.

2.5.2.1. Basit kenar çıkarma yöntemi

Basit kenar çıkarma yöntemi piksel farklarının oluşturduğu serinin maximum değerine dikkat eder. Ortadaki pikselin değeri etrafındaki 8 piksel değerlerinden sırası ile çıkarılır. Bulunan değerler arasında en büyük olanı seçilir. Bu ortadaki pikselin yeni değerini verir.



a) İşlem operatoru b) Resim değerleri

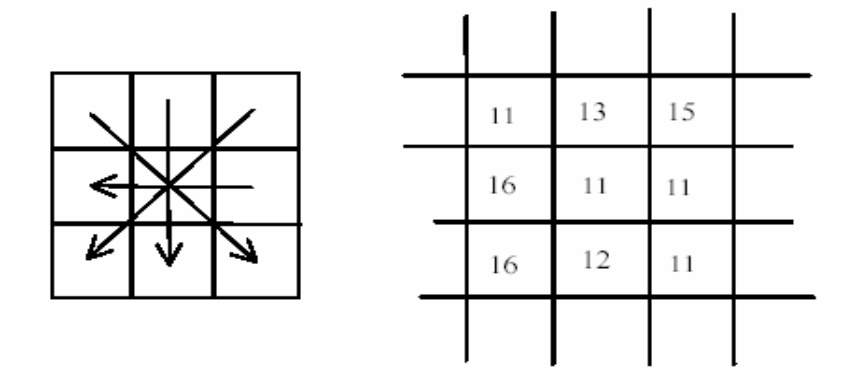
Şekil 2.21. Basit kenar çıkarma yöntemi

Yeni_Piksel=Max{|11-11|,|11-13|,|11-15|,|11-16|,|11-11|,|11-16|,
|11-12|,|11-11|}=5

2.5.2.2 Basit ve hızlı kenar çıkarma yöntemi

Ortadaki pikselin yeni değeri, doğrusal ve köşegen komşu piksellerin birbirlerinden mutlak değerce çıkarılması ile bulunur. Bulunan değerler arasından en büyük olanı seçilir.

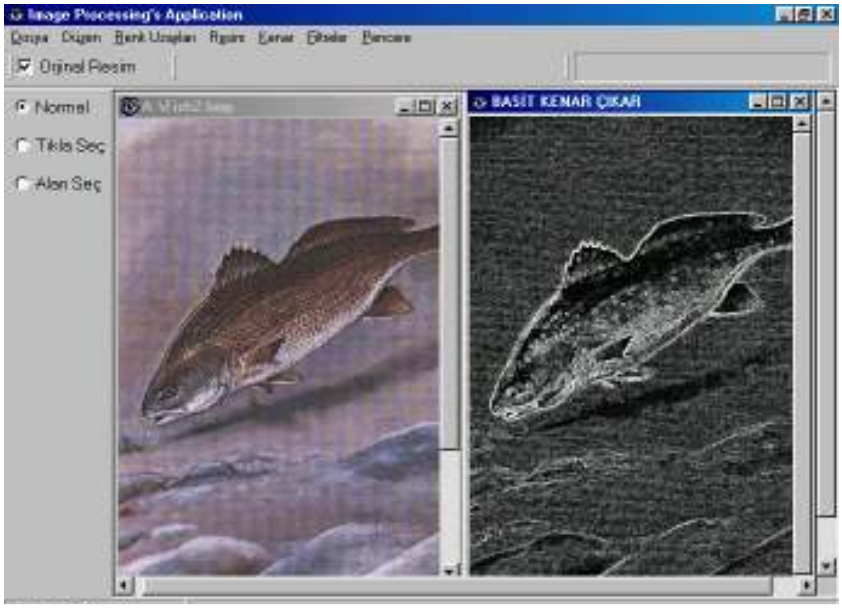
Basit kenar çıkarma yöntemine göre daha hızlıdır. Çünkü her piksel için sadece dört işlem yapması gerekmektedir. Basit kenar çıkarımı ile aynı resim değerleri üzerine uygulandığında aynı sonucu verdiği görülmüştür. Farklı çıkan piksel değerleri yeni oluşturulan görüntü üzerinde önemsenmeyecek bir fark yaratırlar.



a)İşlem operatoru b)Resim değerleri

Şekil 2.22. Basit ve hızlı kenar çıkarma yöntemi

Yeni_Piksel=Maksimum{|11-11|,|13-12|,|15-16|,|11-16|}=5



Şekil 2.23. Basit ve hızlı kenar çıkarma uygulaması

3. NESNE TANIMA SİSTEMLERİ

Nesne tanıma sistemleri kendi içlerinde yada bir veritabanında tanımlı olan modeller sayesinde gerçek dünyadaki bir görüntüden yola çıkarak, nesneleri tanımaya çalışan sistemlerdir. İnsanlar bunu içgüdüsel olarak yapmaya alışmışlardır. Bu görevin algoritmik tanımı çok zordur. Nesne tanıma problemi önceden bilinen nesne modelleri üzerine kurulu bir etiketleme sorunu olarak da yorumlanabilir. Yani bir görüntüdeki ilgilendiğimiz nesnelerin alanına veya alanlarına önceden bilinen özellikler doğrultusunda doğru etiketi verme problemi olarak da yorumlanabilir.

Nesne tanımlama sorunu ayrıştırma sorunu ile doğrudan bağlantılıdır. Kısmi bir tanımlama olmadan ayrıştırmayı yapmak mümkün değildir ve ayrıştırma yapmadan da tanımlama yapmak mümkün değildir.

Nesne tanıma sistemleri genellikle şu özellikleri içerir;

- Modelleri tanımlama
- Karakteristik özellik yakalayıcı
- Hipotez oluşturucu
- Hipotez doğrulayıcı

Eğer tanıma sistemi el yazısı gibi doğrusal olmayan nesneler üzerinde tanıma yapacaksa burada nesnelerin modellerinin içinde olacağı bir veritabanı sistemi gereklidir. Ayrıca nesneler kesin hatlara sahip olmadıklarında bir hipotez oluşturma ve hipotez doğrulama aşamasından da geçmelidir.

Modelleri içeren veritabanı sistem tarafından tanınan bütün cisimlerin, nesnelerin modellerini içerir. Bu veritabanındaki bilgiler tanımlama için kullanılan yaklaşıma bağlıdır. Bu bilgi bir cismin kullanım biçiminden rengine veya geometrik olarak dış yüzeyinin tanımına kadar her şey olabilir. Çoğunlukla modeller soyut değerler içeren birer karakteristik listesidirler. Bir karakteristik ise nesnenin tanımında

veya tanınmasında önemli ve diğer nesnelere göre ayırt edici sayılan herhangi bir özelliğidir. Boyut, renk ve şekil karakteristik için en çok kullanılan özelliklerdir.

Karakteristik özellik tanımlayıcı görüntüler üzerinde işlemler yaparak karakteristik özelliklerin olabileceği bölgeleri saptayarak hipotezlerin oluşturulmasını sağlarlar. Sistem tarafından kullanılan karakteristikler tanımlanmak istenen nesnenin özelliklerine bağlıdır.

Görüntüdeki karakteristikleri kullanan hipotez oluşturucu her olası nesneye bir olasılık atar. Bu durumda işlemleri kolaylaştırmak için veri yapısının da olası olmayan adayları hemen geçersizliğini kanıtlayacak bir yapıda olması önemlidir. Bundan sonra hipotez doğrulayıcı veritabanındaki bilgileri kullanarak hipotezin doğru olup olmadığını kontrol eder. Sistem son olarak deliller ışığında olasılığı en yüksek olan nesneyi doğru nesne olarak kabul eder.

Desenlerin sınıflandırılması bu tür tanıma yapan sistemler için iyi bir örnektir. Fakat yapay zeka sistemlerinin çoğu hipotez oluşturma aşamasını atlayıp doğrudan doğrulama aşamasına geçmektedir. Klasik yaklaşımlardan biri olan korelasyon kullanarak eşleştirme hipotez oluşum aşamasını tamamen atlar.

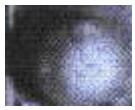
Nesne tanımlama sistemi her aşama için kullanacağı araçları ve teknikleri iyi seçmelidir, metot seçiminde birçok faktör göz önünde bulundurulmalıdır.

3.1. Klasik Nesne Tanıma Yaklaşımı

3.1.1. Şablon kullanarak eşleştirme

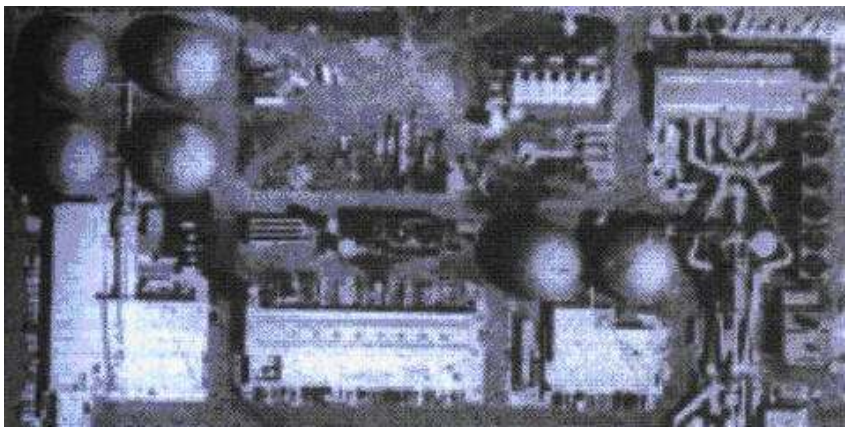
Şablon kullanarak eşleştirme tekniğinin temel amacı verilen görüntü içinde belirli bir nesneye ait örnekleri yine görüntü üzerine bir şablon uygulayarak bulabilmektir. Burada şablon olarak görüntü içinde aradığımız nesnenin bir örneği kullanılır. Aşağıda bu tekniğe ait bir örnek yer almaktadır. Küçük resimde verilen

şablonumuz görüntü içinde bulmak istediğimiz nesne olan gaz tanklarına ait bir görüntüdür.



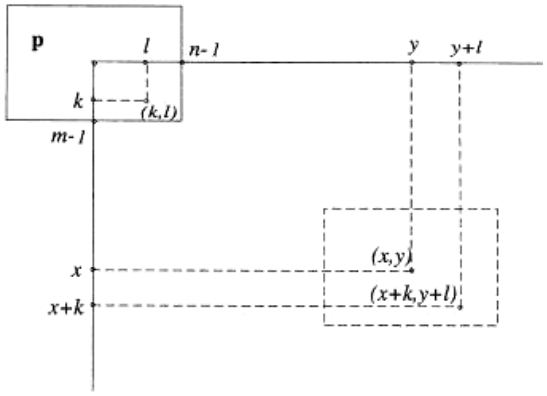
Şekil 3.1. Şablon nesne

Burada amaçlanan; örneği verilen nesnenin verilen görüntü içerisinde bulunmasıdır. Bunun için şablon olarak verilen gaz tankı görüntüsünü şablon maskesi olarak kullanarak aşağıdaki resim içerisinde arama yapılır.



Şekil 3.2. Havadan çekilmiş orijinal resim

İstenen nesnenin yerini belirlemek için gaz tankı maskesi ile görüntü içinde eşleşen piksel grupları beyaza yakın, eşleşmeyenler ise siyaha yakın olacak şekilde işaretlenir. Aşağıdaki grafik şablonun görüntüye nasıl uygulandığını göstermektedir.



Şekil 3.3. Şablonun görüntüye uygulanma grafiği

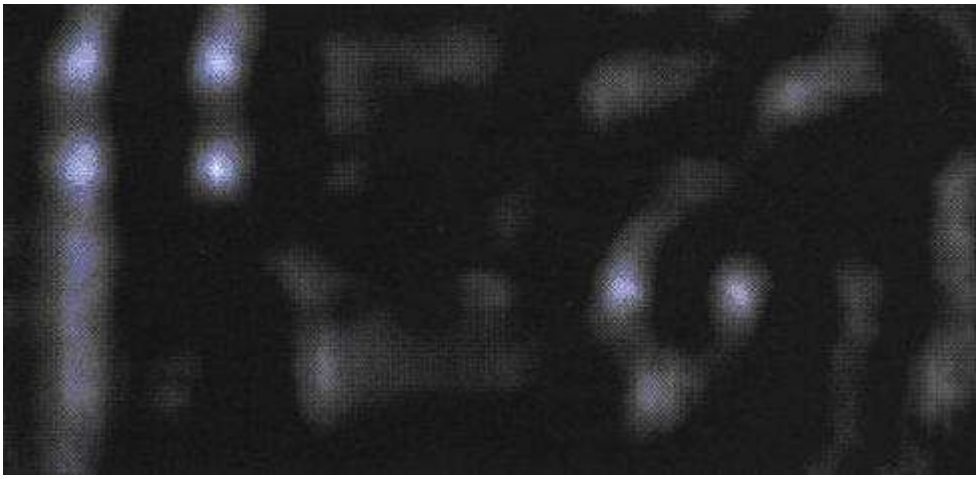
Şablon kullanarak eşleştirme şu formülle özetlenebilir.

$$\mathbf{c} := \mathbf{a} \oplus \mathbf{t}.$$

Bu eşitlikte c çıktı görüntüsü, a kaynak görüntü ve t ise p piksel değerleri ile gösterilen şablonumuzdur. t şu şekilde tanımlanır.

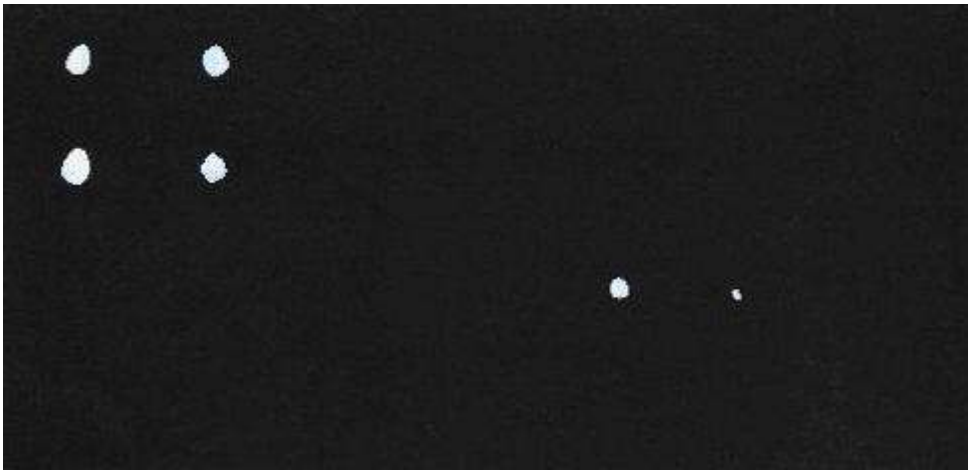
$$\mathbf{t}_{(x,y)}(u,v) = \begin{cases} \mathbf{p}(u-x, v-y & \text{eğer } -(m-1) \le u-x \le m-1 \\ & \text{ve } -(n-1) \le v-y \le n-1 \\ 0 & \text{diğer hallerde} \end{cases}$$

Uygulamanın sonucu aşağıda görülmektedir.



Şekil 3.4. Orijinal resmin şablon uygulanmış sonucu

Yukarıda 6 tankın yerleri beyaz olarak görülüyor. Bu da yöntemin başarılı olduğunu kanıtlar. Fakat görüntü içinde tanklara yakın olarak renklendirilmiş kısımlar da var. Bu kısımlar bulunan tanklara nazaran daha koyu olsalar da yanıltıcı olabilirler. Belirlenen tankları daha da görünür kılmak için önceki bölümde de bahsedilen eşikleme yöntemi kullanmak gerekir. Bu işlem sonucunda istenen sonuca ulaşıldığı görülmektedir.



Şekil 3.5. Eşikleme sonucu elde edilen gaz tankları

Şablon kullanarak eşleştirme ve tanıma yöntemi bu kadar açık sonuç vermesine rağmen çok büyük eksiklikleri vardır. Örneğin şablon içindeki nesne belirli bir açıyla döndürülerek maskeleme yapılırsa sonuç beklenen şekilde olmaz. Ayrıca görüntü ve şablon görüntümüz arasında farklı saatlerde çekilmiş olmalarından dolayı bir ışık farkı varsa sonuç yine hüsran olacaktır. Bu yüzden başta kolay bir teknik gibi görülmesine rağmen şablon kullanarak nesne tanıma sistemleri bu eksikliklerinden dolayı çok az bir grup üzerinde kullanılırken artık tamamen yerini yeni tekniklere bırakmıştır.

3.2. Yeni Nesne Tanıma Yaklaşımları

3.2.1. Özellik çıkarımı

Özellik çıkarımı resimlerden yüksek seviyeli bilginin çıkarılması olarak tanımlanabilir. Bu bilgi şekil, renk gibi resim özellikleri olabilir.

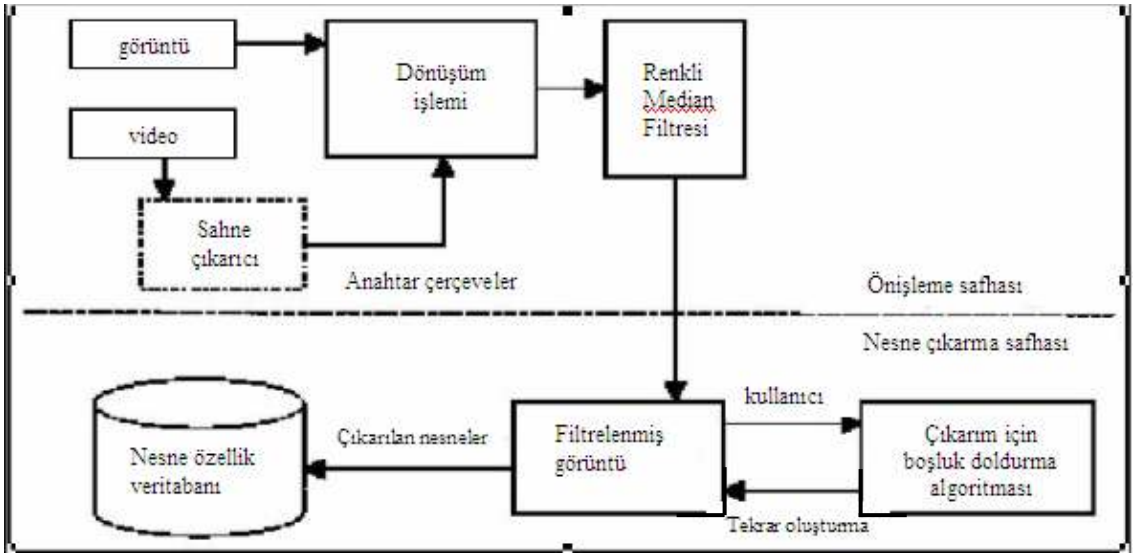
Resim ve video verilerinden saklı nesne temelli bilgiyi çıkarmak için kullanılır. Çıkarım algoritması daha iyi yapmak için filtrelenmiş resim kullanır. Nesne çıkarım metotları üç kategoriye ayrılır.

- Tam Otomatik Çıkarım Metotları: Resim veya video verileri için çıkarım süreci otomatik olarak yapılır. Bütün resim ve video tipleri bu yaklaşım tarzı ile

yönetilemez. Bu yöntem sadece ayrılabilir arka plana sahip video karelerinde ve resimlerde kullanılır.

- Yarı Otomatik Çıkarım Metotları: Kullanıcı nesne çıkarım sürecine yardım eder. Yardımın bir yolu nesne alanını göz önüne getirmek için nesne piksellerinin üzerine tıklayarak bir resmin nesne çıkarımını kolaylaştırmaktır.

- Elle Yapılan Nesne Çıkarımı: Kullanıcı bilgisayar ilişkisi etkileşimden daha fazlasıdır çünkü kullanıcı bütün çıkarım sürecini yönetir. Minimum poligon sınırlama ile bir nesne alanını belirlemek için kullanıcı çizimi kullanılır. Elle yapılan çıkarım çok sıkıcı süreçtir ve çok büyük veri setlerine uygulanamaz. Aşağıdaki şekil Blobworld adlı nesne sorgulama sisteminin özellik çıkarımı aşamalarını göstermektedir.



Şekil 3.6. Yarı otomatik özellik çıkarım sistemi yapısı

3.2.2. Nesnelerin özellikleri

Nesne tanıma sistemlerinin ilk amacı verilen manzara içindeki nesnelerin yerlerini bulmaktır. Bu soruyu cevaplayabilmek için önce nesne dediğimiz "şey"in ne olduğunu tanımlamamız gerekir? Bir şeyi nesne olarak adlandırabilmek hangi özelliklere sahip olması gerekir? Bilgisayarla görme sistemlerinin ulaşmaya çalıştığı

nokta tüm bir sahneyi analiz edebilen ve sahne içindeki nesneleri yorumlayabilen sistemler yaratabilmektir.

Henüz bilgisayarla görme sistemleri nesneleri tamamen ayırabilmekten ve yorumlamaktan uzaklar. Bu durumda böyle bir sistem geliştirebilmek için önce nesne olarak tanımladığımız şeyin ne olduğunu açıkça belirtmemiz gerekir. Nesne olarak tanımladığımız şey bir araba, bir yazı yada yüz olabilir. Görüntü içinde bu nesneleri bulabilmek için kullanılabilecek çeşitli yöntemler vardır. Bu teknikler belirli nesneleri aramak için farklı olabilirler. Çünkü farklı nesneler arasında yapı bakımından farklılıklar olacağı için her belirleme işlemi için ortak bir yöntem önermek imkansızdır. Kullanılan teknikler "nesne"ye özgü teknikler olmalıdır.

Nesne tanıma sistemleri temel olarak nesnelerin;

- Renk
- Doku
- Şekil

özelliklerine göre özelleşirler.

Örneğin; bir araba tanıma sisteminde arabaların renkleri birbirinden çok farklı olacağından arabaların renk özellikleri kullanılamaz. Burada resimler mümkün olduğunca sadeleştikten sonra şekil özelliklerine göre bir algoritma geliştirilebilir. Fakat yüz tanıma sistemlerinde herkesin ten rengi birbirine az çok yakın olacağından renk cetvelinde bir renk aralığı boyunca resim içinde arama yapılabilir. Diğer tüm detaylar bu şekilde atıldıktan sonra gözler ve burun için konuma göre hipotezler oluşturulabilir.

3.2.3. Renk tabanlı tanıma sistemleri

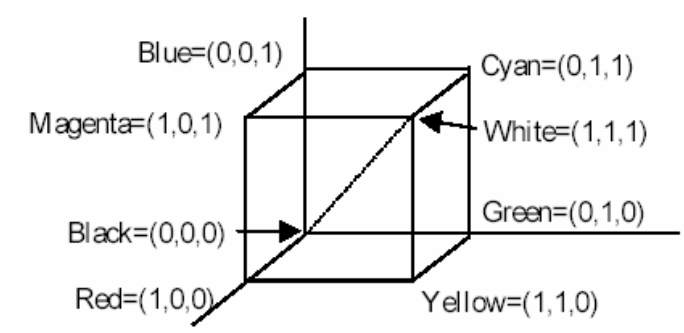
Renk tabanlı tanıma sitemlerinde nesnelerin belirgin renk özelliklerinden yararlanılarak görüntü içindeki istenen nesnelerin bulunması esastır. Renkli görüntüler

RGB renk uzayıyla temsil edilirler. Fakat renk tabanlı tanıma sistemleri genellikle RGB-HSI renk uzayı dönüşümlerini kullanarak işlemlerini HSI üzerinden yaparlar.

3.2.3.1. RGB Renk Uzayı

RGB renk uzayı 3 temel renk olan kırmızı, yeşil ve mavi renklerinden oluşur. Bu renklerin spektrum bileşenleri, sonuçta görünecek rengin oluşmasını gerçekleştirmek için birleşirler.

RGB modeli, her eksenin köşelerinde kırmızı yeşil ve mavi bulunan 3 boyutlu bir küple gösterilirler. Siyah orijindedir. Beyaz küpün köşegeninin sonundadır. Gri siyahtan beyaza doğru takip eden bir doğruyla temsil edilir. Her renk kanalının 8 bitle gösterildiği 24 bit renk grafik sisteminde kırmızı (255,0,0)'dır. Renk küpünde ise (1,0,0) koordinatlarıdır.



Şekil 3.7. RGB renk uzayı gösterimi

RGB modeli bilgisayar grafik sistemlerinin tasarımını kolaylaştırır. Fakat her uygulama için ideal değildir. Kırmızı,yeşil ve mavi renk bileşenleri yüksek düzeyde birbiriyle ilişkilidirler. Bu da bazı görüntü işleme algoritmalarını çalıştırmayı güçleştirir. Birçok işleme teknikleri (Histogram denkleştirilmesi gibi) sadece görüntünün yoğunluk bileşeni üzerinde çalışırlar. Bu işlemler HSI renk modeli kullanarak daha kolay yerine getirilebilir. Birçok kere RGB görüntüyü gri tonlu görüntüye dönüştürmek gerekir.

3.2.3.2. HSI Renk Uzayı

Renk tonu, doygunluk ve yoğunluk rengi tanımlamak için kullanılan 3 özelliktir. Renk modellerine bakarak mantıksal olarak görülür. HSI renk uzayını kullanırken bir renk üretmek için mavi ve yeşilin hangi yüzdede olacağını bilmenize gerek yoktur. Basitçe istenilen renge ulaşana kadar renk tonunu ayarlamak, derinliği değiştirmek için doygunluğu ayarlamak, resmi daha koyu ve daha açık yapmak için yoğunluğu değiştirmek yeterli olacaktır. Bir çok uygulama HSI renk uzayını kullanır. Bilgisayarla görme sistemleri değişik nesnelerin renklerini ayırt etmek için HSI renk uzayını kullanır. Görüntü işleme uygulamaları (histogram işlemleri, yoğunluk dönüşümleri ve kıvrımlar) HSI renk uzayında daha kolay yerine getirilir.

HSI silindirik koordinatlarla modellenmiştir. Renk tonu H, 0 açısıyla gösterilir. 0 dan 360º ye değişebilir.Doygunluk, (S), 0'dan 1'e değişen yarıçapa karşılık gelir, yoğunluk, (I), z ekseni boyunca 0 siyah 1 beyaz olmak üzere değişir.

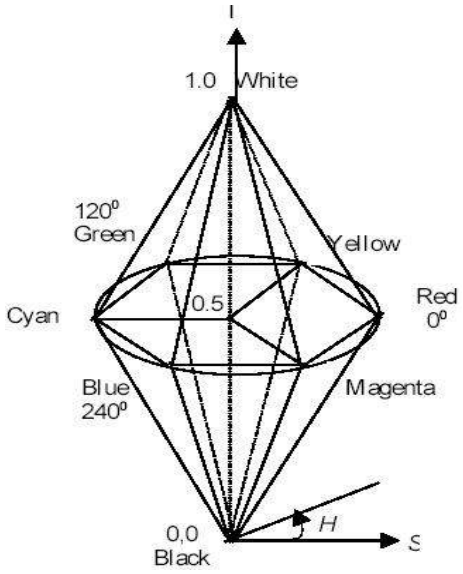
S=0 olduğunda, yoğunluğun gri rengi 1 dir. S=1 olduğunda, renk koni tabanının üst sınırındadır.

H yi ayarlamak 0º de rengi kırmızıdan 120º de yeşile doğru, 240º de maviye doğru değiştirecek ve 360º de tekrar kırmızıya geri dönecektir.

I=0 olduğunda renk siyahtır, bu yüzden H tanımsızdır. S=0 olduğunda renk gridir. H bu durumda da tanımsızdır.

1'e düzeltmeyle bir renk daha koyu veya daha açık hale getirilebilir.

S=1 ve I'yı ayarlamakla, o rengin gölgesi oluşturulur.



Şekil 3.8. HSI renk uzayı gösterimi

Aşağıdaki formül RGB den HSI ya dönüşümü gösterir.

$$I = \frac{1}{3}(R+G+B)$$

$$S = 1 - \frac{1}{R+G+B}[\min(R,G,B)]$$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]$$

3.2.3.3. Renk histogramları

Resim histogramı resmin yoğunluğunu gösteren değerli bir şekildir. Histogram resmin yoğunluğu ve zıtlığı hakkında bilgi verir. Resim histogramı resmin piksel yoğunluklarının basit çubuk grafiğidir. Piksel yoğunlukları X ekseni boyunca, olayların numaraları ise Y ekseni boyunca gösterilir.

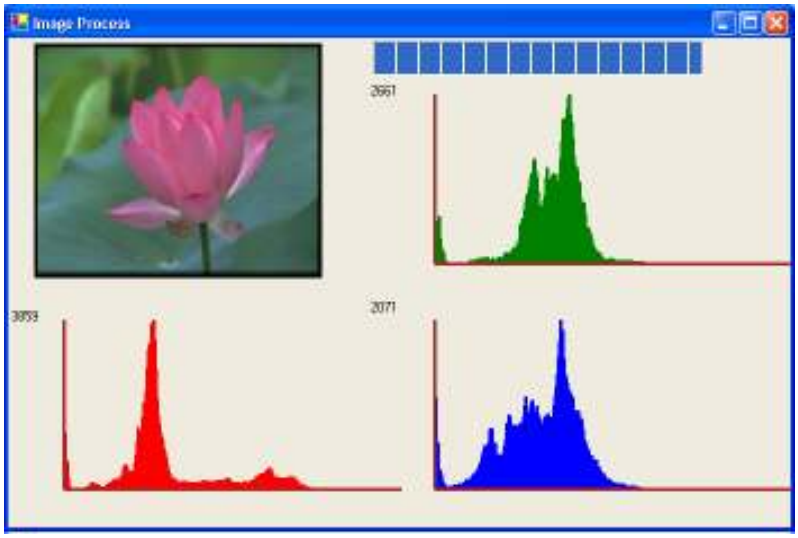
Bir pikselin RGB renk uzayına göre R,G,B üç değeri vardır. Kırmızı histogramı için kırmızı değeri kontrol edilir. Tüm piksellerin kırmızı değerine bakılır. Bunlar bir dizide tutulur. Dizinin indeksi pikselin kırmızı değeridir, belirtilen indeksli dizi

elemanında ise aynı değere sahip olan piksellerin sayısı tutulur. R değeri 8 bit için 256 farklı değer alabilir. Dizinin boyutu ise 255'dir. Daha sonra tüm kırmızı değere sahip piksellerin sayısı hesaplanır. Yani tüm dizi elemanları toplanır.

Bu anlatılanlar hem R, hem G hem de B için ayrı hesaplanır. Kırmızı değerleri tutan, yeşil değerleri tutan, mavi değerleri tutan olmak üzere üç dizi oluşur. Histogram çizilirken kırmızı histogramı için aynı kırmızı değere sahip piksel sayısının en büyüğü (kırmızı dizisinin en büyük elemanı) bulunur.Bu kırmızı histogramının y ekseni üzerinde maximum değeri verir ve ekrana yazılır.

X ekseni 0'dan 255'e kadar değer alır. Histogram çizme fonksiyonu kırmızı dizisinin elemanlarına göre dikey doğrultuda çizgiler çizmeye sağlar. Böylece kırmızı histogramı çizilmiş olur.

Aynı zamanda yeşil ve mavi histogramları da yukarıda anlatılanlar gibi çizilir. Örnek bir histogram çıkarımı aşağıdadır.



Şekil 3.9. RGB renk histogramı

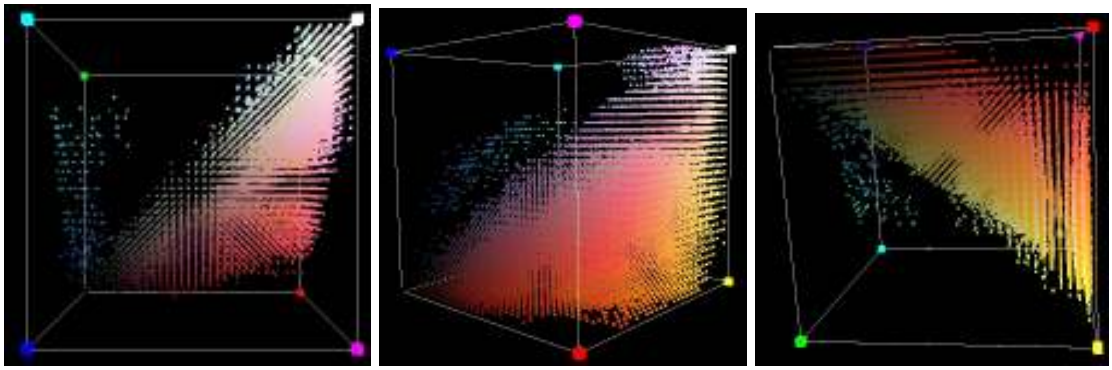
3.2.3.4. Yüz tanıma sistemleri

Yüz tanıma sistemleri herkesin deri rengi tonu belirli bir aralıkta bulunacağından renk tabanlı tanıma yapan sistemlerdir.

Örnek bir yüz tanıma sisteminin nasıl işleyeceği aşağıda anlatılmıştır.

Şekil 3.10. Yüz tanıma için kullanılacak görüntüler

İlk olarak yüz renginin renk tonu aralığını seçebilmek için RGB yada HSI histogramları oluşturulur. Aşağıdaki örnekte bir 3 boyutlu RGB histogramı ve ten rengi aralık değerleri oluşturulmuştur.



Şekil 3.11. Ten rengi aralığını gösteren 3 boyutlu histogram

Görüntü içindeki piksel değerleri teker teker kontrol edilerek histogramda ten rengi aralığında olanlar -örnekte mavi ile- işaretlenir. Eğer sonuç görüntüsü bir gruplama algoritmasıyla birlikte kullanılırsa yüz olmayan kısımların görüntü içinden temizlenmesi de sağlanmış olur.

Aşağıda böyle bir sistemin olası sonuçları görülmektedir.

Şekil 3.12. Ten rengi aralığında seçilen renklerin belirlendiği görüntüler

3.2.4. Doku(Örüntü) Tabanlı Tanıma sistemleri

Örüntü tanıma sistemleri genellikle iki kısımdan oluşur; birinci kısım, örüntülerin tanınması için gerekli özelliklerin veya ölçülecek büyüklüklerin seçilmesi işlemini gerçekleştirir. Bu işleme, özellik çıkarma ve bu işlemi gerçekleştiren sisteme, özellik çıkarıcı adı verilmektedir. Örüntüleri belirlemede kullanılan her bir özelliğin veya ölçülecek büyüklüğün, ölçüm sonucunu veren gerçel sayıya özellik adı verilmekte ve bileşenleri bu özelliklerden oluşmuş vektöre, özelliklerin alındığı örüntüye ilişkin öznitelik vektörü denilmektedir. Pratikte uygun özellik seçimi, bir örüntü tanıma probleminin en güç ve en önemli bölümünü oluşturur. Eğer seçilen özellikler örüntülerin bütün özelliklerini tam olarak taşımıyorsa, tanıma ve sınıflandırma işlemlerinde yapılacak hatalar büyük olacaktır.

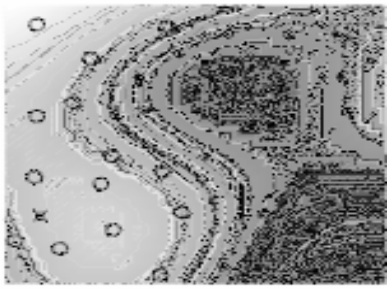
Örüntü tanımanın ikinci kısmı ise, elde edilen öznitelik vektörlerinden faydalanarak örüntüleri sınıflandırılmasıdır. Örüntü sınıflandırma işlemi, öznitelik vektörlerinden oluşmuş uzayın her bölümü bir örüntü sınıfına karşı düşecek şekilde, birbirlerinden ayrık bölümlere ayrılması ve her bölüme ilişkin bir doğrusal ayırım fonksiyonu tanımlanması temeline dayanır.

Şekil 3.13. Örüntü tanıma sistemi yapısı

Örüntü tanıma sistemlerinin en büyük uygulama alanı endüstriyel robot gözlerle birlikte hatalı ürün tespitidir. Özellikle tekstil endüstrisinde oldukça ihtiyaç duyulan bu sistemler son yıllarda kullanılmaya başlanmıştır. Dokunun herhangi bir yerinde çizgi yada yapı bozukluğu bu sistemler tarafından yakalanır ve o kısımda bir problem olduğu ilgili kişiye haber verilir. Bir başka yeni uygulanmaya başlanan alan da araba lastiklerinin üzerindeki şekil bozukluklarının tespitidir.

Bu tür büyük sanayi işletmelerinde hata kontrolü çok zaman alan ve insan tarafından yapıldığı için hatalara açık bir işlemdir. Doku tanıma sistemleri ile bu işlem daha kısa süreli ve dolayısıyla iş gücünü ortadan kaldıracağı için uzun vadede karlı bir yöntemdir. Bu sistemlerde en önemli kısımlardan biri de doğru bir ışıklandırma yapılması ve görüntülerin net bir şekilde işlenebilecek şekilde alınmasıdır. İlk bölümde kısaca bu konulara değinilmişti. Kaliteli görüntüler ve iyi bir örüntü(doku) tanıma sistemi ile otomasyonda hata oranı sıfıra çok yaklaşabilir.

Şekil 3.14. Asya'ya özgü bir kumaş dokusu örneği



Şekil 3.15. Doku tanıma sisteminde hata yakalama

Şekil 3.15'te doku içerisinde uyumu bozan kısımların (sol-alt köşedeki daire ve sağ-üst köşedeki çember) hata tespit sistemi tarafından yakalandığı görülmektedir.

3.2.5. Şekil Tabanlı Tanıma sistemleri

Nesnelerin kenar, köşe, boyut gibi özelliklerini kullanarak tanıma yapan sistemlerdir. Trafik işareti tanıma sistemleri, imza tanıma sistemleri bu tür yaklaşım kullanırlar. Resimlerde renk özelliği kullanılmayacaksa bu özelliği tamamen ortadan kaldırmak hem süre hem performans açısından daha olumlu sonuçlar vereceğinden bu tür sistemlerde genellikle renk bilgisinden kurtulmak için görüntüler gri seviyesine yada siyah-beyaz resimlere dönüştürülür. Sonraki bölümde bir uygulaması yapılmış olan sistem de bu aşamaları kullanmış bir geometrik şekil bilgisini kullanarak tanıma yapan bir sistemdir.

Örnek bir yol ve trafik işareti tanıma sistemini ayrıntılı bir şekilde inceleyecek olursak;

Bütün işaretlerin ortak özellikleri:

- Şekil kırmızıdır.
- Her işarette 3 tane alt şekil vardır.
- İlk olarak karşılaşılan büyük çevrel çemberdir.
- Ardından gelen iki şekilden biri küçük çemberdir
- Diğeri ise model veritabanında bulunan geometrik bir şekildir.

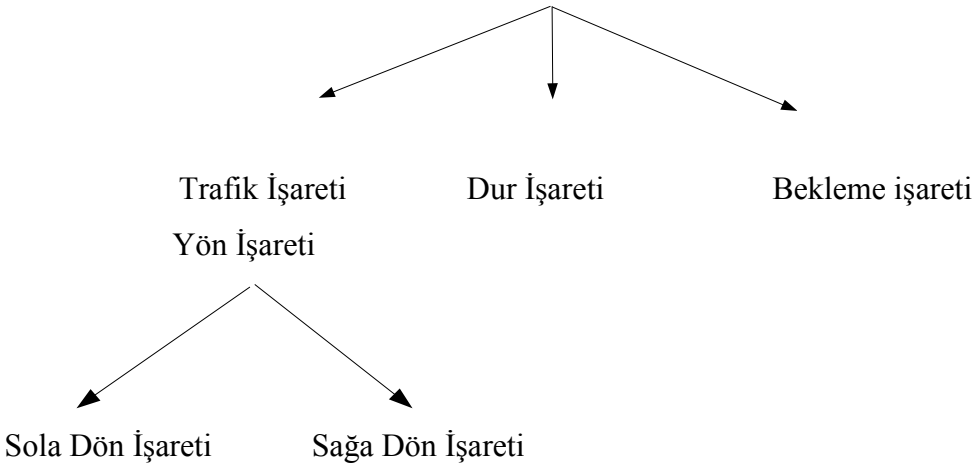
Her tanıma sistemi için bu kadar kesin ve net özellikler bulunamayabilir. Bu özellikler daha önce de belirtildiği gibi tanıma sistemine özgü olmalıdır. Aynı görüntüler üzerinde bile farklı karakteristik özellikler baz alındığında farklı sonuçlar alınabilir. Fakat görüntüler ile ilgili mutlaka karakteristiklerin buna benzer şekilde çıkarılması gereklidir.

Üzerinde algılayıcılar ve bir kamera bulunan basit bir robota bağlı sistemin, kameradan alınan görüntüler üzerinde işlem yaparak yol ve trafik işaretlerini tanımaya çalışması aşamaları şöyle devam eder.

Kısıtlamalar:

- İlk olarak karşılaşılan şekil yani çevrel çember şekillerin en büyüğüdür.
- En Küçük olan şekil referans için kullanılan çemberdir
- Orta boyuttaki şekil mana taşıyan geometrik şekildir.

Bu sistemdeki trafik işaretleri aşağıdaki hiyerarşiye uymaktadırlar:



Her adımdan önce bir hipotez yaratılır, daha sonra bu hipotez doğrulanır, bundan sonra bir sonraki adım için olasılıklar hesaplanır ve sonra bunlar doğrulanır, bu adımlar esnasında herhangi bir yerde işaret tespit etme olasılığının sıfır olduğu tespit edilirse o zaman işlem sürdürülmez görünürde ya işaret yoktur yada işaret tanımlanamayacak kadar bozuktur.

Her nesne tanıma sisteminde bu örnekteki aşamalara benzer aşamalar bulunur.

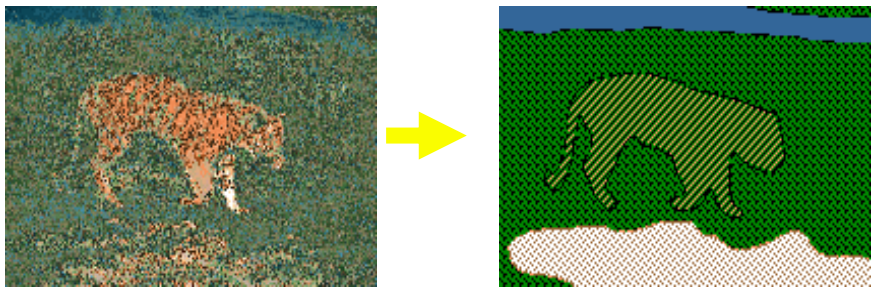
Şekil 3.16. Kameradan alınan görüntü

Şekil 3.17. Belirlenen özellikler doğrultusunda tespit edilen işaret

3.2.5.1. Şekil tabanlı nesne tanıma ile içerik tabanlı nesne sorgulama arasındaki farklar

Berkeley Üniversitesi'nde "Computer Vision Group" tarafından geliştirilmeye başlanan bir içerik tabanlı nesne sorgulama sisteminin ayrıntılarına aşağıda yer verilmiştir. Bu sistemde tam otomatik özellik çıkarımı yapabilen bir sistem hedeflenmiştir. Resimlerin renk özelliklerini yada renk histogramlarını kullanarak sağlıklı bir sonuç elde edilemediği görülmüş ve renk özellikleri kaybedilmeden yalnızca aralarında gruplama yapılarak yakın piksellerdeki renklerin tek renge dönüştürülmesi

sonucu renk azaltılması yoluna gidilmiştir. Renkli resimler için içerik tabanlı nesne sorgulama sistemlerinde sıkça kullanılan bir teknik olan gruplama bir çeşit 3 boyutlu eşiklemedir. Böylece hassas ve ayrıntılı kenar çıkarımı yapan algoritmaların (örneğin Sobel algoritması) gereksiz ayrıntılardan kurtularak sağlıklı çıkarımlar yapılması sağlanmıştır.



a) Orijinal resim b)Renkleri gruplanmış resim

Şekil 3.18. Renk gruplama

Bu tür gruplama her ne kadar kolay gibi görünse de aslında oldukça karmaşık bir işlemdir. Başlı başına oldukça zaman ve emek isteyen böyle bir dönüşüm genellikle RGB görüntüler HSI renk uzayına çevrildikten sonra HSI histogram üzerinde birbirine yakın piksellerin gruplanması şeklinde yapılır. Fakat nesne tanıma sistemlerinin genel yapısında olduğu gibi burada da çok net bir kesinliğe ulaşabilmek için ele alınan piksellerin hangi renk grubuna ait olduğu yönünde bir hipotez oluşturma ve doğrulama aşaması gereklidir.

Her ne kadar şekil 3.18.de oldukça güzel bir çıkarım yapıldığı görülse de bu çıkarımın her görüntü için tam otomatik olarak yapılması bu kadar iyi olmayabilir. Bir başka istemeden alınana bir gruplama aşamasının adımları görülmektedir. Burada bu işlemin aslında tek bir adımdan ibaret olmadığı, pek çok deneme-yanılma yapıldıktan sonra uygun algoritma seçilerek böyle bir gruplamanın mümkün olduğu açıkça görülmektedir.

Örnek olarak yapılan ve bir sonraki bölümde anlatılacak olan nesne tanıma uygulamasında gruplama kısmını ihmal ederek, kesin sonuç alabilmek için azaltılmış renkli resimler üzerinde özellik çıkarımı yapan bir uygulama oluşturulmuştur.

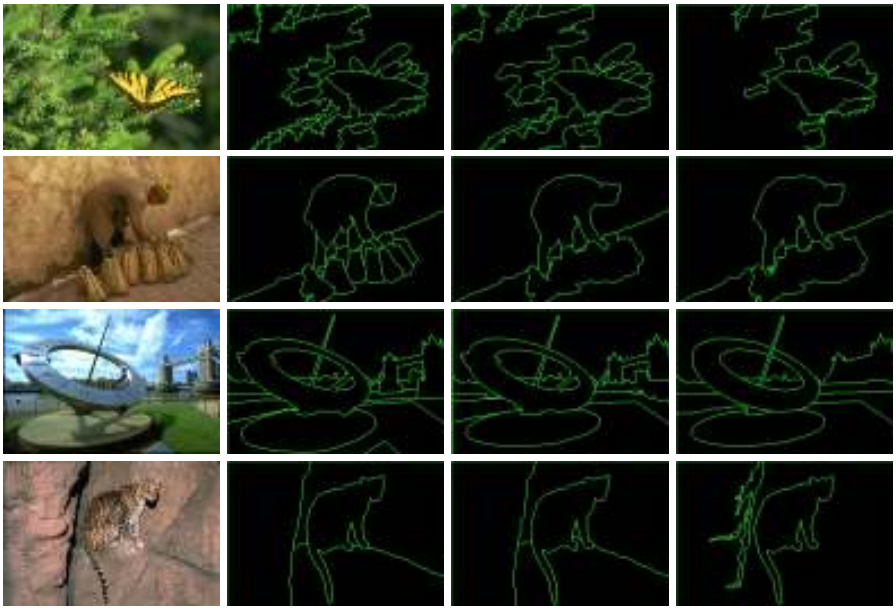
a) Renk gruplama yapılmış resim | b) Orijinal resim ile soldaki resmin kesişimi | c) Orijinal görüntünün gruplanmış hali

Şekil 3.19. Renk özelliğine göre gruplama

Geliştirilen projede daha sonra şekil tabanlı bir çıkarım yapılarak (kenar çıkarımı) nesnelerin yerleri belirlenmeye çalışılmıştır. Aynı projenin devamında diğer sistemlere göre oldukça başarılı bir sonuç elde edilmiştir. Yerleri bu şekilde belirlenen nesneler üzerinden bit düzeyinde bir inceleme daha yapılarak, piksel komşuluklarından faydalanılarak, belirli noktalar saptanmıştır. Bunlar ilgi noktaları olarak adlandırılırlar. Örnek uygulamada daha basit haliyle bu adımlardan geçilmiştir. Model veritabanında da bulunan görüntüler de aynı aşamalardan geçerek ilgi noktaları arasındaki benzerliklere göre aralarında bir eşleşme olup olmadığına bakılır.

İçerik tabanlı nesne sorgulama sistemlerinin, nesne tanıma sistemleriyle ayrıldığı noktalardan biri burasıdır. Çünkü içerik tabanlı nesne sorgulama sistemleri verilen nesnenin benzerlerini görüntü veritabanında arar. Yani tek bir sorgu nesnesini bir yığının içinde arar. Eşleşenleri hipotez oluşturma ve doğrulama kısımlarını kullanmadan sonuç olarak döndürür. Nesne tanıma sistemleri ise tek bir görüntü içinde özellikleri belirli bir nesneyi çıkarmakla uğraşır.

Bir örnekle açıklayacak olursak; elimizde örnek bir iğne olduğunu varsayalım. İçerik tabanlı nesne sorgulama sistemleri pek çok kutu içerisine bakarak hangisinin içinde elimizdeki iğneden olduğuyla ilgilenir. Nesne tanıma sistemleri ise verilen kutu içerisinde iğne olabileceğini varsaydığı "şey"lerle elimizde bulunan iğne ile karşılaştırıp sağlam bir kanıt bulmaya çalışır. Kutuların her birinin içinde mutlaka en az bir iğne olduğunu varsayarsak bulduğu nesne(leri) "iğne" olarak etiketleyip sonuca ulaşır.



Şekil 3.20. Renk gruplamadan sonra kenar çıkarımı yapılan görüntüler

3.3. Topolojik İlişkiler ve Semantik

Topolojik ilişkilerin formel modellerinin geliştirilmesi uzaysal anlamlandırma, coğrafi bilgi sistemleri (CBS), görüntü yorumlama ve bilgisayarla görme konuları için büyük önem taşır. Bu konular son yıllarda araştırmacılarının da üzerinde çokça durdukları konulardır. Görüntü içinde nesnelerin sınırlarını bulan algoritmaların ve yerlerini tespit edebilen tekniklerin geliştirilmesiyle, uzaysal ilişkilerin modellenmesi alanında da önemli gelişmeler kaydedilmiştir. Fakat sınırları tam olarak belirtilmemiş nesneler üzerinde bu ilişkilerin belirlenmesi oldukça zordur. Topolojik ilişkilerin matematik modellerinin çıkarılması ve bu modellerin semantik olarak ifadeleri konusunda da çalışmalar yapılmaya devam etmektedir.

3.3.1. Topolojik İlişkilerin Modellenmesi

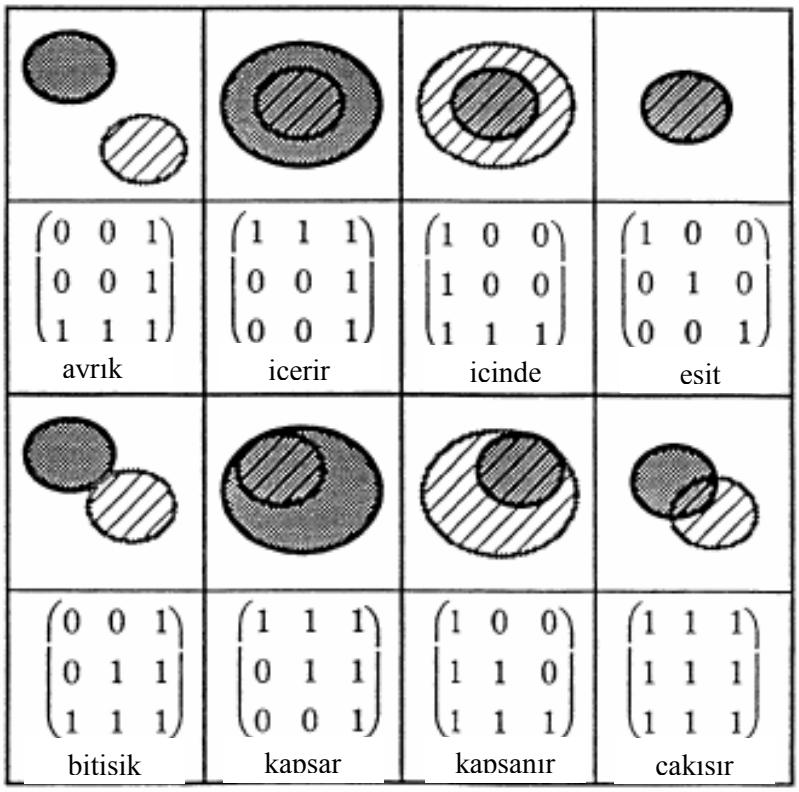
Nesneler arasındaki topolojik ilişkiler döndürme, ölçeklendirme gibi dönüşümlerle kaybolmayacak olan uzaysal ilişkilerdir. İkili sistem topolojik ilişkileri modellemek için "9-kesişim modeli" adlı bir model geliştirilmiştir. Bu modelde A nesnesi 3 bölüme ayrılır. [Clementini E.]

- İç kısım (A° ile gösterilir)
- Sınır (A^ ile gösterilir)
- Dış kısım (A^{-} ile gösterilir)

İki belirli A ve B nesnesi için topolojik ilişkiler iki nesnenin iç kısımlarının, sınırlarının ve dış kısımlarının kesişmelerine göre belirlenir. Bu iki nesnenin 6 parçasına göre 9 kesişim belirlenebilir. Bu 9 kesişim aşağıdaki matrisle ifade edilebilir.

$$\begin{array}{c} \\ A^{\circ} \\ A^{\wedge} \\ A^{-} \end{array} \begin{array}{c} \begin{array}{ccc} B^{\circ} & B^{\wedge} & B^{-} \end{array} \\ \begin{pmatrix} A^{\circ}\cap B^{\circ} & A^{\circ}\cap B^{\wedge} & A^{\circ}\cap B^{-} \\ A^{\wedge}\cap B^{\circ} & A^{\wedge}\cap B^{\wedge} & A^{\wedge}\cap B^{-} \\ A^{-}\cap B^{\circ} & A^{-}\cap B^{\wedge} & A^{-}\cap B^{-} \end{pmatrix} \end{array}$$

Yukarıdaki matrisle ifade edilen 9 kesişim modelini şekil üzerinde özel adlarıyla görelim.



Şekil 3.21. 9-kesişim modeliyle belirlenmiş 8 temel topolojik ilişki

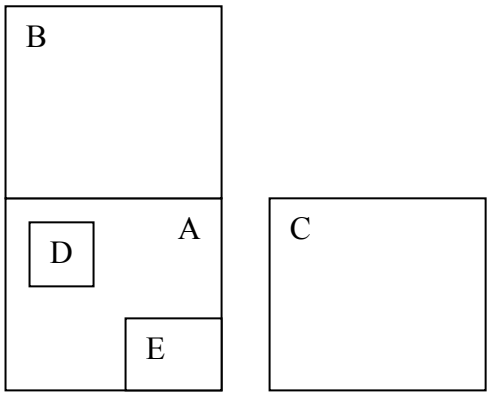
3.3.2. Mantıksal ve semantik gösterim

Yapay zeka üzerinde çalışan bilim adamlarının üzerinde durduğu konulardan biri de mantık tabanlı gösterimlerdir. Uzaysal anlamlandırma yapan YZ uygulamalarında kullanılan mantık gösterimleri gerçek dünyanın modellenmesi açısından başarılı sonuçlar vermektedirler. Yüklem mantığının ve çıkarım kurallarının esas alındığı bu gösterim sistemleri topolojik ilişki modelleri ile birleştiğinde mantıksal görüntü ve nesne analizi yapabilen sistemler oluşturulabilir.

Aşağıda Randell tarafından oluşturulan, topolojik ilişkiler bazında bir mantıksal gerçekler ve çıkarımlar ifadesi verilmiştir.

(1) sol(A,C), altında(A,B), içerir(A,D), kapsar(A,E), bitişik(A,B), ayrık(A,C), ayrık(B, C)

Yukarıdaki mantık önermesi grubu aşağıdaki gibi bir nesne grubunu modelleyebilir.



Şekil 3.22. (1) modeli için girdi şekiller

(2) sol(X, Y) AND sol(Y, Z) => sol(X, Z)
(3) içerir (X, Y) AND ayrık (X, Z) => ayrık (Y, Z)

Yüklem mantığının semantik olarak ifade edilebilmesi görüntüler içindeki nesnelerin birbirlerine göre uzaysal konumlarını yorumlama ve görüntüleri yazıya

dönüştürme sistemlerine olanak tanıyacaktır. Bu sistemler engelli eğitimi için kullanılabilirler.

Aşağıda Türkçe dili için bir yüklem mantığı ifadesi ve semantik ifade karşılıkları verilmiştir.

“içinde (ev,orman), bitişik(ev,araba), ayrık(ağaç,ev)”

şeklinde oluşturulan bir ifadenin semantik karşılığı;

“(Görüntü içerisinde) Orman içinde bir ev, evin bitişiğinde bir araba, evden uzakta bir ağaç (vardır).”

şeklinde olacaktır. Fakat bu işlemin uygulaması Türk dilinin yapısı, yüklem mantığı ve doğal dil işleme konularında yoğun bir bilgi ve alt yapı gerektirdiğinden bu sistemi oluşturmak oldukça zor ve zahmetli bir işlemdir. Burada topolojik sistemler ve semantik ifadeleri yalnız teorik olarak incelenmiştir.

4. UYGULAMA

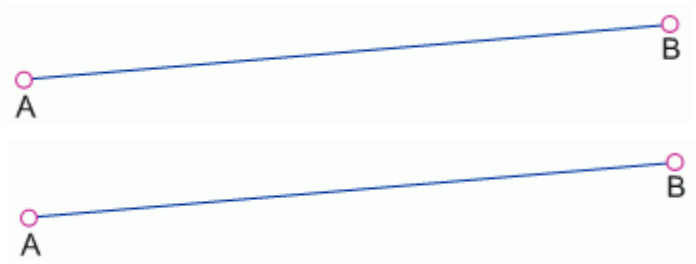
4.1. Ön Çalışmalar

Program içinde nesne çıkarımı yapan bölümde kullanılacak olan yöntem ızgara nesnelerin vektöre dönüştürülmesi işlemidir. Bu konuda ayrıntılı bir çalışma yapılmış ve programda kullanılması gerektiğinden vektörel grafik formatı olan SVG formatı incelenmiştir. Görüntü işleme ve nesne belirleme olarak yazılan iki ayrı program ileriki çalışmalarla birleştirilmeye çalışılacaktır.

4.1.1. Izgara görüntülerin vektöre dönüşümü

Izgara görüntüler, harita, yazılı doküman, fotoğraf gibi grafiksel bilgilerin tarayıcılarla taranması ile ya da dijital formatta resim çekebilen kameralarla elde edilen ve ızgara dosya (bmp, gif vs.)formatında bilgisayara aktarılan görüntülerdir. Fakat vektörel grafikler yazılımlar sayesinde elde edilir. Bir yazılım kullanmadan tarayıcımızdan aktardığımız resimleri vektör grafik olarak kaydedemeyiz.

Tüm görüntünün piksellerden oluştuğu normal görüntü dosyaları bitmap formatında kaydedilir. Piksel değerlerinin açık olarak görülebileceği format bmp yada tiff'tir.Görüntüler, farklı sıkıştırma yöntemleri ile boyut olarak daha küçük ama kalite olarak çok yakın ".jpg",".gif",".jpeg" gibi dosya formatlarına dönüştürülebilir. Fakat vektöre dönüştürmenin sonucu ortaya çıkan resim; ".svg", ".eps", ".swf" gibi özel çizim programlarınca kullanılan bir formatta saklanır.



Şekil 4.1 Örnek çizimler

Yukarıdaki ilk şekil A noktasından B noktasına çizilmiş bir ızgara çizgiye aittir. Hemen altındaki şekil ise aynı şeklin vektörel formatta çizilmiş halidir. Bu çizgiler arasındaki benzerlik ızgara ile vektör çizimler arasında gözle görülür bir fark olmadığı kanısını uyandırabilir. Fakat bu, kullandığımız çizim programıyla da ilgilidir. Yüksek çözünürlükte çizilen çizgiler her zaman düz bir görüntü oluşturacaklardır. Izgara ve vektör görüntüler arasındaki fark çizgiyi yeterince yakınlaştırdığımızda açıkça kendini gösterecektir.

a) Izgara çizim b)Vektörel çizim

Şekil 4.2. Izgara ve vektörel çizim arasındaki fark

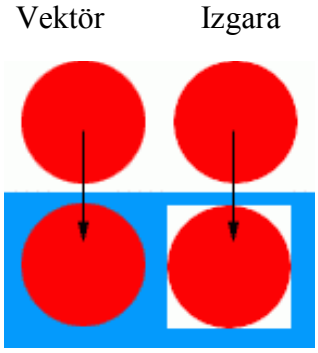
4.1.2. Vektörlerle Çalışmanın Avantajları

Çözünürlükten bağımsız olan, ölçeklenebilir vektör grafikler genellikle AutoCad yada Autodesk gibi yüksek geometrik işleme ve hassasiyet gerektiren mühendislik uygulamalarında çizim yapmak amacıyla kullanılırlar. Ayrıca çizgi film, karikatür benzeri görüntülerde de kullanılıyor olsa da birçok farlı amaca hitap etmektedirler. Vektörlerin çizimlerde kullanılmasının nedeni çizgilerin vektörel görüntülerde ızgara görüntülere oranla daha belirgin ve net olmasındandır.

Ayrıca ızgara görüntülerde, görüntü içindeki her bir pikselin değeri tutulduğundan kaplayacakları alan resmin alanının –her piksel için üç byte kullanılırsa- 3 ile çarpımı kadar olacaktır. Vektörel görüntüler ise sadece belirli pikselleri tutuğu için boyutları ızgara görüntülere oranla çok küçük olacaktır.

Vektörlerin bir başka üstünlüğü; ekran çözünürlüğüne göre yeniden meydana getirildikleri için ekranda iyi gözükmenin dışında kağıda çıktı alırken de başarılı sonuçlar sağlamaktadırlar. Bu tip grafikler metinleri de, özellik olarak barındırdığından resim içindeki yazılar da kaliteli olarak görüntülenir. Ayrıca çizilen şekillerin arka planları olmak zorunda değildir. Vektör grafikler dörtgenlerle sınırlandırılmamıştır. Başka nesnelerin üzerine yerleştirebilen bu grafiklerin boş bölümlerinde diğer nesneler görüntülenebilir.

Izgara biçimde kaydedilen bir dairenin dörtgen bir çerçevesi mutlaka olacaktır. Bu çerçevenin boş alanları arka plana uymadığında istenmeyen görüntüler ortaya çıkabilir. Ama vektör grafiklerde böyle bir sorun yaşanmaz.



Şekil 4.3. Vektör ve Izgara gösterimleri

4.1.3. SVG (Ölçeklenebilir Vektör Grafikleri) Formatı

Vektör görüntüler yalnızca nokta bilgisi tuttuğu ve resmin bütününe ilişkin piksel bilgileri tutmadıkları için klasik resim gösterme programlarıyla görüntülenemezler. Ancak piyasada çokça bulunan ve bu formatı tanıyan ticari görüntüleme programlarıyla görüntülenebilirler. Vektörel görüntüleri farklı formatlarda da kaydedebiliriz. Fakat kullanım kolaylığından dolayı biz program içinde SVG dosya formatını kullandık.

SVG henüz yaygınlaşmamış olan ama üzerinde yoğun çalışmalar yapılan bir vektör grafik formatıdır. Bu formatın yaygınlaşmasına yardım edecek en büyük özelliği

XML teknolojisinden faydalanmasıdır. SVG formatı GIF ve JPG dosyalarına İnternet'te alternatif olması için geliştirilen başlıca formatlardan biridir.

Bu formatla çok daha derin renk özellikleri elde etme, mükemmel piksel konumlandırma, daha başarılı çıktılar sağlama, sayısız yazı tipini barındırıp hatta içindeki metinlerin aranabilmesi mümkündür.

4.2. Analiz

Uygulama programlarının analiz aşamalarında girdilerin ve çıktı olarak istenenlerin tam olarak belirlenebilmesi gereklidir. Bu ikisi net olarak ortaya konduktan sonra program tasarımı ve kodlanması çok kolaylaşacaktır. Burada amaçlanan; sistemin temel geometrik şekiller üzerinde ayırım ve tanımlama yapabilmesidir.

Girdi olarak kullanacağımız görüntüler Microsoft Paint programında oluşturulacak olan ve geometrik şekiller içerecek olan renkli görüntülerdir.

Çıktı olarak programdan beklenenler; şekillerin köşe noktalarının tespit edilerek model veritabanı kısmında tanımlı olan şekillerle eşleştirilmesi ve sonucunda tanımlı şekillerle benzerlik göstermiş ise hangi şekil olduğunun belirtilmesi, aksi takdirde tanımlama yapılamadığının belirtilmesidir.

4.3. Program Tasarımı

4.3.1. Örnek görüntülerin özellikleri

Program için örnek olacak görüntüler Microsoft Paint programında oluşturuldu. Nesne ve görüntü tanıma sistemlerinin en önemli ve zor kısımları özellik çıkarma kısmı olduğu için yaptığımız programda en temel nesne özellikleri olan kenar ve köşe özelliklerini kullandık. Programa girdi olarak kullandığımız görüntüler düz arka plan üzerinde içi dolu renkli geometrik şekiller olarak oluşturuldu.

Düz arka plan yerine karışık arka plan kullanmış olsaydık tezin ilk bölümlerinde kısaca anlattığım görüntü iyileştirme tekniklerini kullanmak zorunda kalacaktık. Bu konuya seminer çalışmamda da yer verdiğim için bu teknikler ve gürültü giderme algoritmaları bu tezin konusu dışında bırakılmıştır. Sadece en temel görüntü işleme adımlarından olan verilen RGB renkli görüntülerin tek bitlik yani siyah-beyaz görüntülere dönüştürülmesini programın ilk adımı olarak kullandık. Bu konu programın kodlanması bölümünde ayrıntılı olarak anlatılmıştır.

4.3.2. Adımlar

Programın temel adımları 3 bölümden oluşmaktadır. İlk adım görüntü işleme teknikleri kısmında anlatılan resmin işlenerek, tanınmanın kolaylaştırılması için yapılması gereken ön işlemlerdir. Bu işlemler, renkli görüntülerin eşikleme yapılarak iki renkli görüntülere dönüştürülmesi ve kenar bulma algoritması kullanılarak şekillerin kenarlarının bulunması aşamalarıdır. Daha sonra ızgara verilerle işlem yapmanın zorluklarından kurtulmak için şekilleri vektörlere çevirmemiz gerekir. Son aşamada ise kenar ve köşe sayıları tespit edilerek tanımlı olan şekillerin hangi şekiller olduğu belirlenecektir.

4.4. Programın Kodlanması ve Kod Örnekleri

Hedeflenen amaca ulaşabilmek iki ayrı program yazılmıştır. İlk program temel görüntü işleme uygulamalarını içerir. İkinci program ise tanıma işlemini gerçeklemek üzere yazılmıştır.Izgara görüntüleri vektörlere çevirmek için ise kendi kütüphanemizi oluşturmamız gerekeceğinden yardımcı bir vektör dönüşüm programı kullanılmıştır. Vektör dönüşümü için kullandığımız program Vextractor 2.60 adlı profesyonel bir dönüşüm programıdır.

İki program da Borland C++Builder ile yazılmıştır.

Programın tek eksik yanı ızgaradan vektöre dönüşümü başka bir program aracılığıyla yapıyor oluşudur.

İlk programda kullanılan görüntü işleme fonksiyonları ImageProc.h adlı kütüphanede tanımlanmıştır.

ImageProc.h

```
#ifndef ImageProcH
#define ImageProcH

#include <StdCtrls.hpp>
#include <vector>
using namespace std;
enum TFunction {_brightness=0,_contrast,_blur,_bwImage,_side};
namespace ImageProc{
class TProcessedImage:public TObject{
    Graphics::TBitmap *FBMP;  // konu bitmap
    public:
    __fastcall TProcessedImage();
    __fastcall TProcessedImage(Graphics::TBitmap* aBmp);
    __fastcall TProcessedImage(AnsiString FileName);
    void __fastcall Create(Graphics::TBitmap* aBmp);
    Graphics::TBitmap* getBitmap();          // oku
    void setBitmap(Graphics::TBitmap *aBmp); // ata
    void clearBitmap(); // temizle
    TProcessedImage*      grayScale();              //gri donusum
    TProcessedImage*      brightness(__int8 aLight); //parlaklik
    TProcessedImage*      contrast(float aContrast); // karsitlik
    TProcessedImage*      invert();                 // ters
    TProcessedImage*      bwImage(__int32 aTreshold); // iki renk
    TProcessedImage*      side(__int32 aTreshold);   // kenar bulma
    vector<unsigned __int8> histoGrab();// parlaklık dizisi    -
    TProcessedImage*      crop(int x,int y,int width,int height); // kırp
```

Gri dönüşüm fonksiyonu aşağıdaki şekilde tanımlanmıştır.

```
TProcessedImage* TProcessedImage::grayScale()//gri donusum
 {
          Graphics::TBitmap *Temp=new Graphics::TBitmap();
          TProcessedImage   *Result;
          Temp->Width=FBMP->Width;
          Temp->Height=FBMP->Height;
          __int32 r,g,b,color,newValue;
          for (int i=0;i<FBMP->Width;i++)
          {
               for (int j=0;j<FBMP->Height;j++)
               {
                    color=FBMP->Canvas->Pixels[i][j]; // renk
                    r=(color >> 16) & 0xFF;
                    g=(color >> 8)  & 0xFF;
                    b=color &  0xFF;
               newValue=(__int32) ((float).56*g + (float).33*r + (float).11*b);
               newValue=(newValue << 16 | newValue << 8 | newValue);
               Temp->Canvas->Pixels[i][j]=(TColor)newValue;
               }
          }
          Result=new TProcessedImage(Temp);
          return Result;
     }
```

Aydınlık fonksiyonu;

```
TProcessedImage*  TProcessedImage::brightness(__int8 aLight)
{
 #define NORMAL_COLOR( _c ) if (_c <0) _c=0; \
                            if (_c >255) _c=255;
     Graphics::TBitmap *Temp=new Graphics::TBitmap();
```

```
        TProcessedImage   *Result;
        Temp->Width=FBMP->Width;
        Temp->Height=FBMP->Height;
       __int32 r,g,b,color;
        for (int i=0;i<FBMP->Width;i++)
        {
              for (int j=0;j<FBMP->Height;j++)
              {
                    color=FBMP->Canvas->Pixels[i][j]; // renk
                    r=(((color >> 16) & 0xFF)+aLight);
                    g=(((color >> 8) & 0xFF) +aLight) ;
                    b=((color &  0xFF) +aLight) ;
                    NORMAL_COLOR(r)
                    NORMAL_COLOR(g)
                    NORMAL_COLOR(b)

              Temp->Canvas->Pixels[i][j]=(TColor) (r << 16 |g <<8 | b);
              }
        }
        Result=new TProcessedImage(Temp);
        return Result;
    }
      Ters dönüşüm fonksiyonu;

TProcessedImage* TProcessedImage::invert()
    {
        Graphics::TBitmap *Temp=new Graphics::TBitmap();
        TProcessedImage   *Result;
        Temp->Width=FBMP->Width;
        Temp->Height=FBMP->Height;
       __int32 r,g,b,color;
        for (int i=0;i<FBMP->Width;i++)
```

```
        {
            for (int j=0;j<FBMP->Height;j++)
            {
                color=FBMP->Canvas->Pixels[i][j]; // renk
                r=(0xFF-((color >> 16) & 0xFF))& 0xFF;
                g=(0xFF-((color >> 8) & 0xFF)) & 0xFF;
                b=(0xFF-(color &  0xFF)) & 0xFF;
            Temp->Canvas->Pixels[i][j]=(TColor) (r << 16 |g <<8 | b);
            }
        }
        Result=new TProcessedImage(Temp);
        return Result;
    }
```

Eşikleme fonksiyonu;

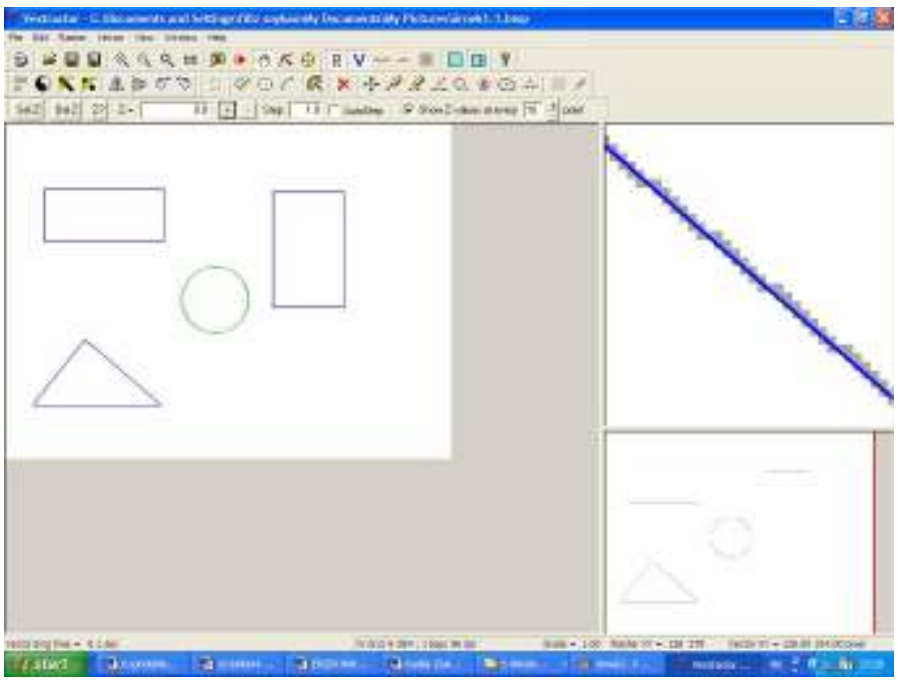
```
TProcessedImage*        TProcessedImage::bwImage(__int32 aTreshold)
    {
     Graphics::TBitmap *Temp=new Graphics::TBitmap();
     Graphics::TBitmap *objBitmap=this->grayScale()->getBitmap();
        TProcessedImage   *Result;
        Temp->Width=objBitmap->Width;
        Temp->Height=objBitmap->Height;
        __int32 r,g,b,color;
        for (int i=0;i<objBitmap->Width;i++)
        {
            for (int j=0;j<objBitmap->Height;j++)
            {
                color=objBitmap->Canvas->Pixels[i][j]; // renk
                if (color<aTreshold){
                    Temp->Canvas->Pixels[i][j]=(TColor) 0;
                }else
                {
```

```
                    Temp->Canvas->Pixels[i][j]=(TColor)0xFFFFFF;
                }} }
        Result=new TProcessedImage(Temp);
        return Result;      }

      Kenar çıkarma fonksiyonu;

TProcessedImage*   TProcessedImage::side(__int32 aTreshold)
    {
        // siyah beyaz donusum
        TProcessedImage *Result= bwImage(aTreshold);
        Graphics::TBitmap *Bmp=Result->getBitmap();
        delete Result; // sil
        Graphics::TBitmap *TempBitmap=new Graphics::TBitmap();
        TempBitmap->Width=Bmp->Width;
        TempBitmap->Height=Bmp->Height;
        // Her pixel[i,j] için
        for (int i=0;i<Bmp->Width-1;i++)
        {
            for (int j=0;j<Bmp->Height-1;j++)
            {
        // kenar sorgusu (sağıyla ve altıyla karşılaştır).
        if (
            (Bmp->Canvas->Pixels[i][j]!=Bmp->Canvas->Pixels[i+1][j])
            ||(Bmp->Canvas->Pixels[i][j]!=Bmp->Canvas->Pixels[i][j+1])      )
        {
            TempBitmap->Canvas->Pixels[i][j]=(TColor)0x0;
        }else{
            TempBitmap->Canvas->Pixels[i][j]=(TColor)0xFFFFFF;
        }
            } // for
        } // for
```

```
        Result=new TProcessedImage(TempBitmap);
        return Result;
}
void TProcessedImage::setBitmap(Graphics::TBitmap *aBmp)
{
        delete FBMP;
        FBMP=new Graphics::TBitmap(*aBmp);
}
```

Yukarıdaki kodlar incelendiğinde her bir çıkarım işleminin ayrı fonksiyonlar halinde düzenlenmesi, kodun derli toplu olmasını ve birbirleri içinden çağrılabilmelerini sağlar. Kenar çıkarma fonksiyonunda olduğu gibi önce siyaha beyaza dönüşüm fonksiyonu çağrılarak Kaydırma çubuğu ile belirlenen eşik değerine göre eşikleme yapılması bu iki adımı birleştirmiştir.

Bu aşamadan sonra Vextractor adlı vektöre dönüşüm programını kullanarak şekillerimi vektör olarak ifade ediyoruz. Vextractor programının çevirim yaparken alınan bir görüntüsü aşağıdadır.



Şekil 4.4. Vextractor programı form görüntüsü

Şeklin sol tarafında ilk programla elde ettiğimiz kenarları çıkarılmış şekillerimiz vardır. Sağ üstte ise Vextractor programının yaptığı dönüşüm ve normalizasyon işlemi açıkça görülmektedir.

İkinci programda kullanılan özellikler SVG dosya formatının çözümlenmesi sonucu vektörlerin özelliklerinin belirlenmesi, şekillerin köşe noktalarının bulunması, tanımlı şekillerin belirlenmesi şeklindedir. Bu kısma ait kod parçaları aşağıdadır.

```
void __fastcall TForm1::VektorA1Click(TObject *Sender)
{
  try
{          if(shapes.size()>0){
            shapes.clear();
          }
            Image1->Canvas->FillRect(Rect(0,0,Width,Height));
            dlgOpen->Filter="vector (*.svg)|*.svg|Tüm dosyalar (*.*)|*.*";
            dlgOpen->Execute();
            vectorFileName=dlgOpen->FileName;
            this->Caption="Goruntu Tanimlayici " + dlgOpen->FileName;
            memFile->Lines->LoadFromFile(vectorFileName);
            //ShowMessage(memFile->Text);
            // tüm texti oku
             AnsiString txt=clearAllBlankSpace(memFile->Text);
            // ShowMessage(header);
            // basligi oku
             AnsiString header=getRootSvgFile(txt);
             RECT r=getViewBox(header);
              Panel1->Width=r.right;
              Panel1->Height=r.bottom;
              Image1->Width=Panel1->Width;
              Image1->Height=Panel1->Height;
  this->Width=Panel1->Width+25;
```

```
        this->Height=Panel1->Height+ 54;
         //eleman sayısı bul
         //ShowMessage(getPointSetCount(txt));
         // Noktalar kümesini hazirla
          getPolySet(txt);
           // tümünü çiz
          drawAll();
       }catch(Exception &e){
            ShowMessage("Hata");     }
}
//---------------------------------------------------------------------------
//Bir katar vektoru dondurur
vector<AnsiString> TForm1::Split(AnsiString str, String splitter){
         vector<AnsiString> result;
         AnsiString sub;
          // içinde olduğu sürece
          while(str.Pos(splitter)>0){
                    sub=str.SubString(0,str.Pos(splitter)-1);
                    if (sub.Length()!=0){
                     result.push_back(sub);
                    }
            str=str.SubString(sub.Length() + splitter.Length()+1,str.Length());
              }
              // son parca hep olacak
              result.push_back(str);
              return result;
}
// bosluk temizler
     AnsiString TForm1::clearAllBlankSpace(AnsiString str){
     AnsiString result="";
     for(int i=1;i<=str.Length();i++){
           if(str[i]=='\n'){
```

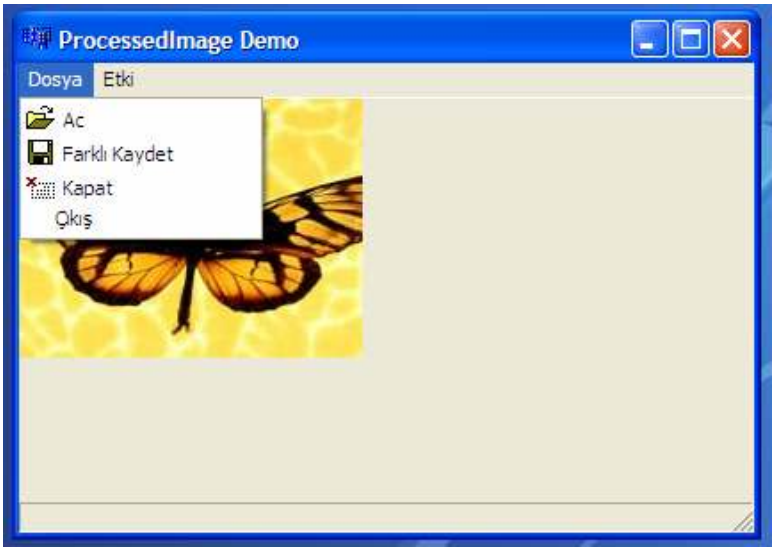
```
            result+="";
           }else if(str[i]=='\t'){
            result+="";
           }else{
            result+= str[i];
           }    }
       return result;
}
// koku dondurur
AnsiString TForm1::getRootSvgFile(AnsiString str){
       AnsiString tail=Split(str,"<svg")[1];
       AnsiString body=Split(tail,">")[0];
       return body;
}
// bir dortgen dondurur
RECT TForm1::getViewBox(AnsiString header){
       AnsiString tail=Split(header,"viewBox=\"")[1];
       AnsiString view=Split(tail,"\"")[0];
       AnsiString temp="";
       for(int i=1;i<=view.Length();i++){
            (view[i]=='.')?temp+=',':temp+=view[i];
       }
       view=temp;
       //ShowMessage(view);
       vector<AnsiString> data=Split(view," ");
       // pozisyon al
        RECT r;
        try{
         r= Rect((int)data[0].Trim().ToDouble(),
                  (int)data[1].Trim().ToDouble(),
                  (int)data[2].Trim().ToDouble(),
                  (int)data[3].Trim().ToDouble());
```

```
        }catch(Exception &ex){
      }
      return  r;
}

// nokta kümesi sayısı
int TForm1::getPointSetCount(AnsiString allText){
       // öndeki parca ise yaramaz
      int pointCount=Split(allText,"points=\"").size()-1;
      int pathCount=Split(allText,"<path").size()-1;
      return pointCount + pathCount;
}
      // tüm noktalar dizisinin kümesini döndür
void TForm1::getPolySet(AnsiString allText){
       vector<AnsiString> tails=Split(allText,"points=\"");
      //:sil
       //ShowMessage("tüm text " + allText);
     for(int i=1;i<tails.size();i++){
            // her bir kuyrugu parcala
            AnsiString body=Split(tails[i],"\"/>")[0];
            AnsiString temp="";
            for(int i=1;i<=body.Length();i++){
             // ayrac degistir gosterim
             (body[i]==',')?temp+=';':temp+=body[i];
            }
            body=temp;
            temp="";
             for(int i=1;i<=body.Length();i++){

             // numerik gosterim
             (body[i]=='.')?temp+=',':temp+=body[i];
             }
```

```
            body=temp;
            //ShowMessage("body= " + body);
            vector<AnsiString> pointStrings=Split(body," ");
            for(int i=0;i<pointStrings.size();i++){
             if(pointStrings[i].Length()!=0){
             shapes.push_back(new ShapeObject(body));
                 }   }  }
        }
    void TForm1::drawAll(){
        for(int i=0;i<shapes.size();i++){
        shapes[i]->drawTo(Image1);
        }}
```

4.5. Form Görüntüleri ve Örnekler

İlk programın içindeki uygulamaları görebilmek için çok renkli bir görüntü üzerinde deneme yapılmıştır. Bu görüntü içinde şekil olmadığı için nesne tanıma ve vektörleştirme aşamaları için değil sadece yapılan programın yeteneklerinin tam olarak görülebilmesi için seçilmiştir.

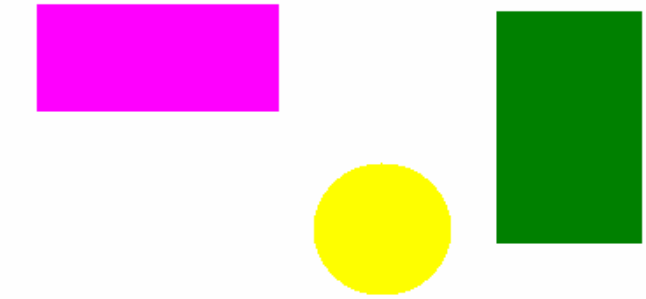


Şekil 4.5. ImageProcess v1.1 programının örnek form görüntüsü

Aşağıdaki şekilde de görüldüğü gibi kenar bulma algoritması için hassasiyet ayarı kullanıcının seçebileceği şekilde oluşturulmuştur.

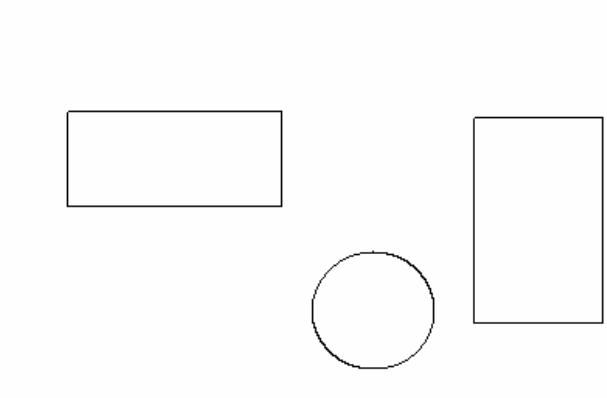
Şekil 4.6. Eşik değere göre kenar çıkarımı fonksiyonu

Örnek şekil dosyalarımızı ise şu şekilde oluşturduk.



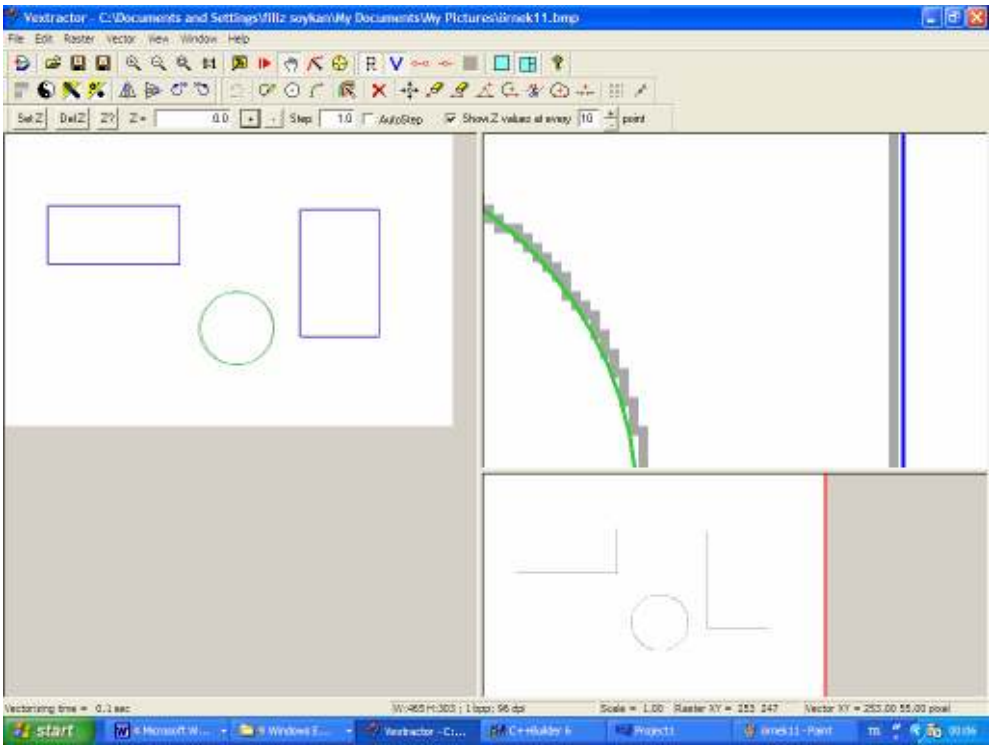
Şekil 2.7 Örnek şekil dosyası

Örnek şekil dosyamızı ImageProcess v1.1.adlı ilk programımıza girdi olarak verdiğimizde renk özellikleri kaybedilerek siyah-beyaz bir görüntüye dönüşmüş ve kenar çıkarma fonksiyonu uygulanarak içi boş şekiller olarak alıyoruz.



Şekil 2.8 ImageProcessing aşamasından sonraki görüntü

Sonraki aşama vektöre dönüşüm aşamasıdır.

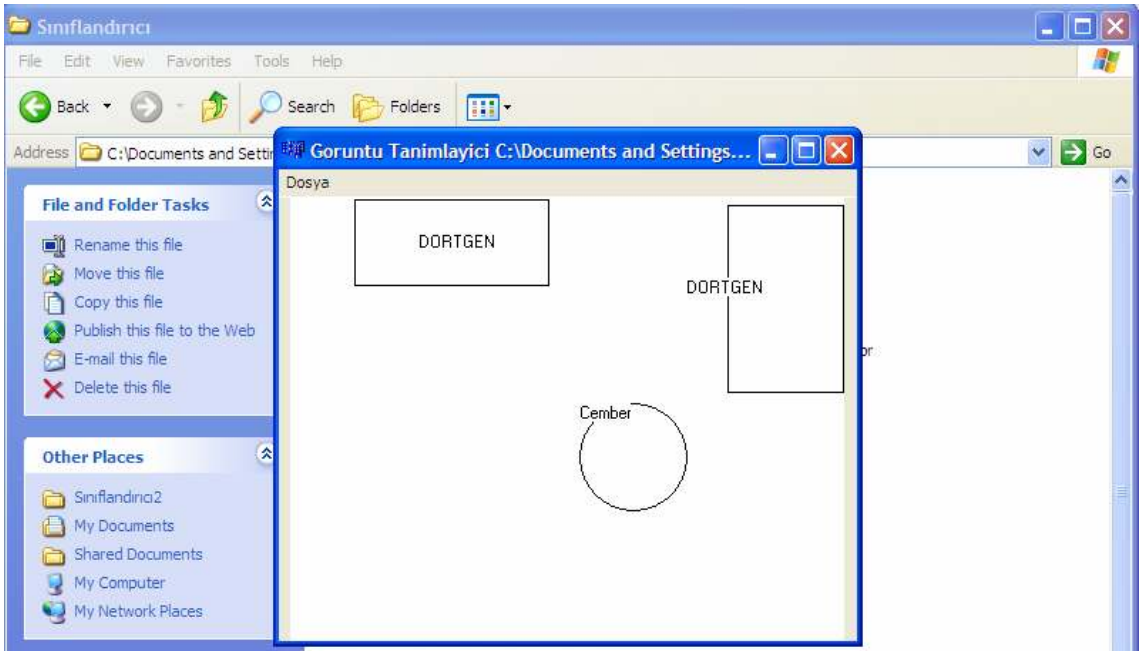


Şekil 2.9. Şekillerin vektöre dönüşüm aşaması

Şekiller vektörleştirildikten sonra SVG dosyası olarak kaydedilir. İkinci programın girdisi olacak görüntü, SVG formatında olmalıdır.Çünkü program SVG formatını tanıyacak şekilde yazılmıştır. ImageRecognizer adlı program görüntüyü analiz ederek içindeki şekilleri bulmaya çalışır.

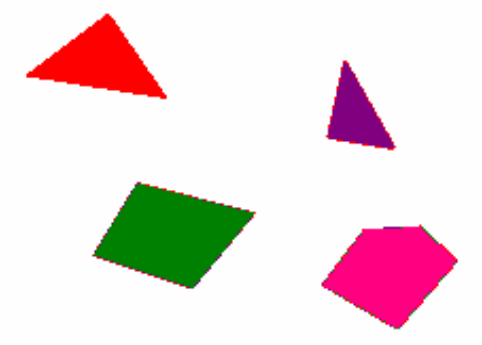
Fakat Vextractor programının dolayısıyla da ImageRecognizer'in hassasiyeti fazla olduğu için Paint programında yaratılan düşük çözünürlüklü şekiller üzerinde her zaman istenen sonucu veremeyebiliyor. Yani Vextractor ile aynı doğrultu üzerinde fazladan bir kenar ortaya çıktığında programı yanıltıcı sonuçlar alabiliyoruz. Program, içinde tanımlı olan üçgen, dörtgen ve çember bulma esasına göre çalıştığı için (örneğin bir dörtgen için bir kenarının tam düz olarak algılanamaması nedeniyle bir köşe daha varmış gibi vektörizasyon yapılması, ImageRecognizer'in bu şekli bir dörtgen değil tanımlı olmayan beşgen olarak görmesi) bunların dışında bir şekli "tanımlanamadı" olarak görmesini sağlar.

Bu istenmeyen durum için bir çözüm, Vextractor programında piksel hasiyeti artırılarak çizgiler üzerindeki kırılmaların engellenebilmesidir. Fakat düşük hassasiyet de çemberler üzerinde yanlış tanımlamalara yol açabilmektedir.Dolayısıyla bu sorunu da ortadan kaldırmak için üçgen ve kare yada dikdörtgen olarak dışında çizilen dörtgenlerin ayrı bir hassasiyet değeri ile vektörleştirilmesi, çember ve x ve y düzlemleriyle paralel kenarlara sahip olan dörtgenlerin ayrı bir hassasiyet değeri ile vektörize edilmesidir. İlk oluşturduğumuz örnek görüntü dosyası 1.4 piksel hassasiyetine göre vektörize edilmiş ve iyi bir tanımlama işlemi gerçekleştirilmiştir.

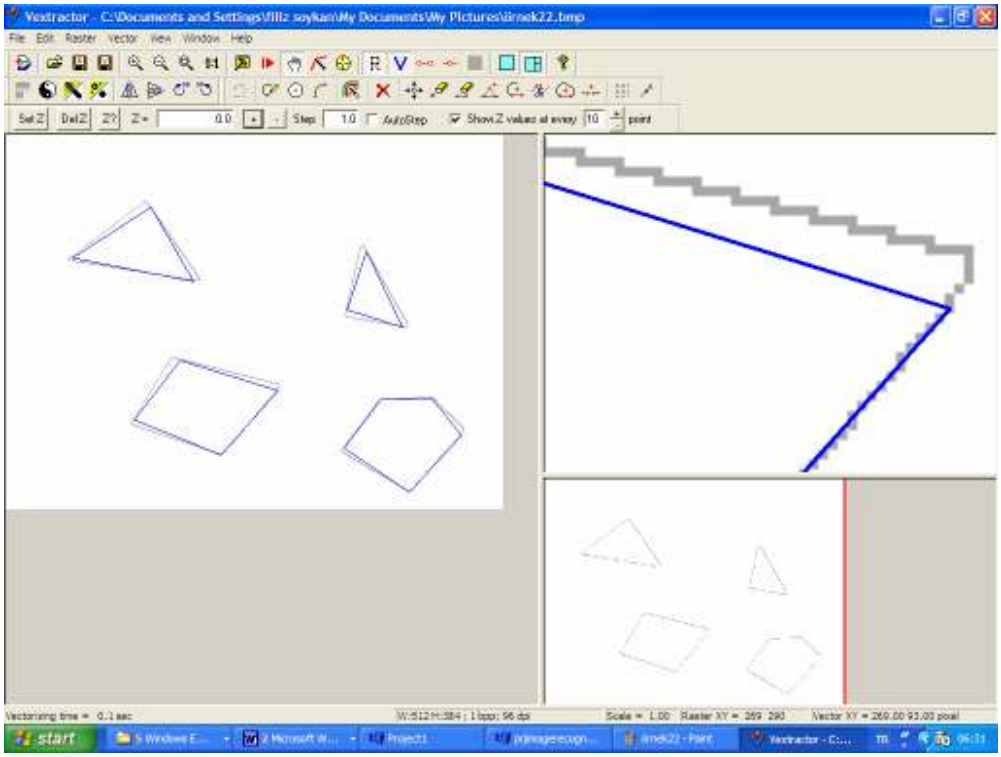


Şekil 2.10. Görüntü tanımlama programı sonucu

Yukarıda bahsedilen sorunun çözümüne ilişkin örnek bir uygulama daha yapılmıştır.



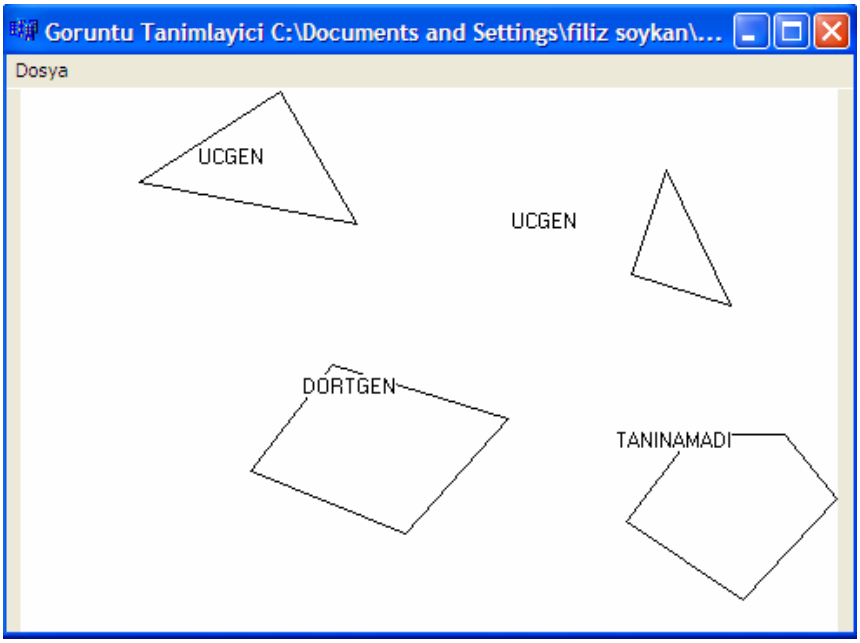
Şekil 2.11. Örnek görüntü dosyası



Şekil 2.12. Örnek vektöre dönüşüm görüntüsü

Bu örneğimizde piksel hassasiyeti olarak 8.0 piksel belirlediğimizde sağ üstte görülen ve köşe yapısını bozan pikseller ihmal edilerek sağlıklı sonuca ulaşılmıştır. Bu vektörümüzü kaydederek ImageRecognizer dosyamıza girdi vektörü olarak sokalım. Sonraki şekilde de görüldüğü üzere tanıma işlemi başarıyla gerçekleşmiştir.

Program içinde tanımlı olan çember, üçgen ve dörtgen şekillerinin sayısı artırılabilir. Daha önce de söylendiği gibi beşgen,altıgen, sekizgen gibi köşe sayılarına göre farklılık gösteren şekiller, programlar modüler olarak yazıldığından rahatlıkla eklenilebilir. Burada amaçlanan temel geometrik şekiller olduğu için bu tanımların yeterli olduğunu düşünüyorum.



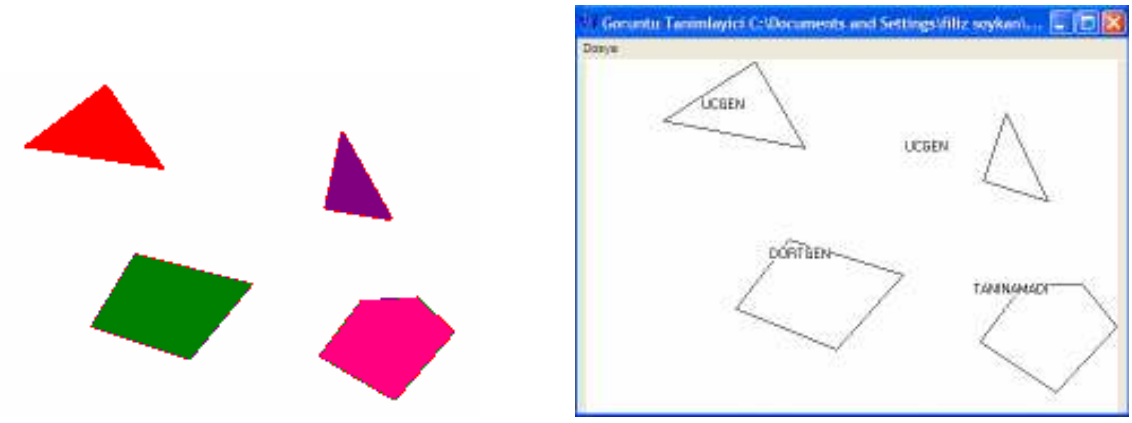
Şekil 2.13. Görüntü tanımlayıcı sonuç ekranı

5. SONUÇLAR

Görüntü içindeki nesneleri tanıma konusu üst düzey bir konu olup son yıllarda robotik alanındaki gelişmelerle birlikte akıllı tanıma ve yapay görme sistemlerinin bu konuya olan ilgisini artırmıştır. Nesne tanıma konusu tez içinde ilk aşamalarından itibaren ele alınmış, alt düzey işlemler olan görüntü işleme tekniklerine de yer verilmiştir. Görüntülerin elde edilmelerinden başlayarak, işlenmeleri, içeriklerinin analizi konuları sıralı biçimde işlenmiştir.

Nesne tanıma sistemleri son yılarda konuya artan ilgiyle birlikte kendine pek çok uygulama alanı bulmuştur. Bu tezde farklı alanlarda, nesnelerin farklı özelliklerine göre tanıma yapan sistemlerden örnekler verilmiştir.

Temel görüntü işleme ve ayrıştırma tekniklerini gerçekleyen bir uygulama geliştirilmiştir. Görüntü işleme programıyla verilen, renkli şekiller içeren, görüntülerin içindeki şekiller belirlenerek kenar özellikleri çıkarılmış ve vektöre dönüşümleri gerçeklenmiştir. Vektör yapısı ve SVG formatı üzerinde çalışılarak görüntüdeki geometrik şekilleri bulan ikinci bir program daha geliştirilmiştir. Aşağıdaki örnekte ilk alınan görüntü ve sistemden son çıkan görüntü gösterilmektedir.



a) ilk görüntü b)sistemden çıkan son görüntü

Şekil 5.1. Uygulama sonucu

Ayrıca tez içinde yapay zekanın en büyük ilgi alanlarından olan manzara yorumlama ve görüntülerin yazıya dönüştürmesi konularının ilk safhası olan görüntü içindeki nesnelerin birbirlerine göre konumlarının mantıksal ve semantik olarak incelenmesi teorik olarak yapılmıştır.

İleri konular olarak tezin devamında, çıkarılan nesnelerin birbirlerine göre durumları incelenerek ve topolojinin semantik modellenmesi konusunda daha yoğun bir çalışma yapılarak izafi bir görüntü yorumlama sistemi geliştirilebilir. Uygulamaya geçebilmek için yapay zeka sistemlerinin, doğal dil işlemenin ve modelleme sistemlerinin üzerine yoğun bir çalışma yapılması gerektiği için bu tezde nesne tanıma uygulaması gerçeklenerek, topolojik ilişkilerin modellenmesi konusuna giriş yapılmıştır. Tezin devamında görüntü yorumlama uygulamaları için gerekli konularda araştırma yapılabilir. Görüntü yorumlama sistemleri ses sistemleri ile birleşirse görme engelli kişiler için kullanılabilecek, görüntüleri yazıya ve sonra sese dönüştüren bir sistem oluşturulabilir.

Halen büyük bir hızla devam etmekte olan robotik ve bilgisayarla görme alanlarındaki gelişmeler nesne tanıma konusunun çok yıllar gündemde olacağının belirtilerini vermektedir.

KAYNAKLAR

Belongie S., Malik J., and Puzicha J., 2000, "Shape Context: A new descriptor for shape matching and object recognition"

Berg A.C., Berg T.L.,2005, "Shape Matching and Object Recognition using Low Distortion Correspondence"

Breuel, T.M., 1990, "Indexing for Visual Recognition from a Large Model Base" A.I. Memo 1108, A.I. Lab., Mass. Inst. Technol.

Carson C., Thomas M., Belongie S., Hellerstein J. M., Malik J., 1999, "Blobworld: A System for Region-based Image Indexing and Retrieval" Visual Information Systems,

Chen, C.H., Kak, A.C., 1989, "A robot vision system for recognizing 3-D objects in low-order polynomial time." IEEE Trans. Syst. Man Cybernetics

Clementini E, Di Felice P., 1996, "A model for representing topological relationships among complex geometric features in spatial databases", Information Sciences

Connell, J.H., Brady, M., 1987, "Generating and generalizing models of visual objects." Artificial Intell.

Gordon, G.G., 1992, "Face recognition based on depth and curvature features." In Proc. IEEE Conf. Comput. Vis. Patt. Recogn.,

Gottschalk, P.G., Turney, J.L., Mudge, T.N., 1989, "Efficient recognition of partially visible objects using a logarithmic complexity matching technique." Int. J. Robotics

Grimson, W.E.L., 1990, "Object Recognition by Computer: The Role of Geometric Constraints" MIT Press.

Huttenlocher, D.P., Ullman, S., 1990,"Recognizing solid objects by alignment with an image."

Jacobs, D.W., 1989, "Grouping for Recognition" A.I. Memo No. 1177, A.I. Lab., Mass. Inst. Technol.

Joachim M.B., Jitendra M., Pietro P., 2002 ,"Image recognition : Visual grouping, Recognition and learning"

Johansson B., 2000, "A Survey on: Contents Based Search in Image Databases", Computer Vision Laboratory, Department of Electrical Engineering Linkoping University

Lowe, D.G., 1985, Perceptual Organization and Visual Recognition. Boston: Kluwer.

Mahoney, J.,1987, “Image Chunking: Defining Spatial Building Blocks for Scene Analysis”

Moses, Y., Ullman, S., 1991, “Limitations of Non Model-Based Recognition Schemes” A.I. Memo 1301, A.I. Lab, Mass. Inst. Technol.

Mundy, J.L., Zisserman, A., 1992, “Geometric Invariance in Computer Vision” Cambridge: MIT Press.

Sato K., Ikeuchi K. and Kanade T, 1992, “Model based recognition of specular objects using sensor models.” CVGIP: Image Understanding

Strat T.M., Fischler M.A., 1991, “Context-based vision: Recognizing objects using information from both 2-D and 3-D imagery.”

Price K. E., 2001, “Relaxation Matching Tecniques - A Comparison”

Rensink, R.A., 2000, “Differential Grouping of Features”

Randal C. Nelson A., 2003, ``From Visual Homing to Object Recognition"

Rubner Y., Tomasi C., Guibas L.J., 2003, “The Earth Mover’s distance as a metric for image retrieval”

Torralba, A., Oliva, A., Freeman, W. T., 2003,“Object recognition by scene alignment”

Ullman S., 1989, “Aligning pictorial descriptions: An approach to object recognition.”

Walker Renninger L., Malik J. , 2004, "When is scene identification just texture recognition?"

Wayner, P.C. , 1991, “Efficiently using invariant theory for model-based matching.”, In Proc. IEEE Conf. Comput. Vis. Patt. Recogn.

Weiss, I. , 1993, “Geometric invariants and object recognition.”

Wong, E.K, 1992, “Model matching in robot vision by subgraph isomorphism.”

Woodham R.J., 1987, “Stable representation of shape.” In Computational Processes in Human Vision

ÖZGEÇMİŞ

16.09.1979 tarihinde İstanbul'da doğdum. İlkokulu Avcılar İlkokulu'nda, ortaokulu Basınköy Ortaokulu'nda bitirdim. Lise eğitimimi İstanbul-Ataköy Hasan Polatkan Yabancı Dil ağırlıklı lisesinde tamamladıktan sonra 1997 yılında Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandım. 2001 yılında mezun oldum. 2001 yılı aralık ayında araştırma görevlisi olarak Trakya Üniversitesi Bilgisayar Mühendisliği bölümünde çalışmaya başladım. Halen aynı görevde çalışmaktayım.

EK-A

Artificial intelligence	Yapay zeka	
Segmentation	Ayrıştırma	
Image	Görüntü	
Image processing	Görüntü işleme	
Computer vision	Bilgisayarla görme	
Thresholding	Eşikleme	
Template matching	Şablon kullanarak eşleştirme	
Content based image query	İçerik tabanlı nesne sorgusu	
Object recognition	Nesne tanıma	
Hue	Renk	H
Saturation	Doygunluk	S
Intensity	Yoğunluk	I
Scalable vector graphics	Ölçeklenebilir vektör grafikleri	SVG
Raster images	Izgara görüntüler	
Noise	Gürültü	
Pattern reconition	Doku tanıma	
Digital images	Sayısal görüntüler	
Topological relations	Topolojik ilişkiler	
Disjoint	Ayrık	
Contains	İçerir	
Inside	İçinde	
Equal	Eşit	
Meet	Bitişik	
Covers	Kapsar	
Covered by	Kapsanır	
Overlap	Çakışır	