Görüntü İşleme ve Şekil Tespit Algoritmaları

B.A. DURSUN[1] ve H.F. KESKİN[2]

[1] Selçuk Üniversitesi, Konya/Türkiye, 213302116@selcuk.edu.tr
[2]Selçuk Üniversitesi, Konya/Türkiye, 213302070@selcuk.edu.tr

Özet **- Görüntü işleme, bilgisayarla görmede kullanılan bir özellik çıkarma tekniğidir. Tekniğin amacı, bir oylama prosedürü ile belirli şekil sınıfındaki nesneler üzerinde işlem yapmaktır. Bu makalemizde Paul Hough dönüşümüyle daireler, elipsler ve diğer kapalı şekillerin tespitini açıklayacağız.**

Anahtar Kelimeler **– Paul Hough Transform, Image Procces, Shape detection, Circle detection**

I. Hough Dönüşümü Tarihçesi

Hough dönüşümü, görüntü analizi, bilgisayarla görme ve dijital görüntü işlemede kullanılan bir özellik çıkarma tekniğidir Tekniğin amacı, bir oylama prosedürü ile belirli bir şekil sınıfındaki nesnelerin kusurlu örneklerini bulmaktır.

Başlangıçta Hough tarafından 1959 yılında kabarcık odası fotoğraflarının makine analizi için icat edildi. Bugün evrensel olarak kullanıldığı şekliyle Hough dönüşümü, 1972'de Richard Duda ve Peter Hart tarafından icat edildi ve bunu Paul Hough 'un ilgili 1962 patentinden sonra "genelleştirilmiş Hough dönüşümü" olarak adlandırdı. Dönüşüm bilgisayarlı görme topluluğunda Dana H. Ballard tarafından 1981 tarihli "Hough dönüşümünün şekilleri algılamak için genelleştirilmesi" başlıklı bir dergi makalesiyle popüler hale getirildi.

II. Hough Dönüşümü kronik sorunları ve bulunan çözüm

Dijital görüntülerin otomatik analizinde, genellikle düz çizgiler, daireler veya elipsler gibi basit şekillerin algılanmasıyla ilgili bir alt problem ortaya çıkar. Birçok durumda, görüntü alanında istenen eğri üzerinde bulunan görüntü noktalarını veya görüntü piksellerini elde etmek için bir ön işleme aşaması olarak bir kenar detektörü (CannyEdge Fonksiyonu) kullanılabilir. Ancak hem görüntü verisindeki hem de kenar algılayıcıdaki kusurlar nedeniyle, istenen eğriler üzerinde eksik noktalar veya pikseller olabileceği gibi, ideal çizgi/daire/elips ile gürültülü kenar noktaları arasında uzamsal sapmalar olabilir. Kenar dedektörü Bu nedenlerden dolayı, çıkarılan Kenar özelliklerini uygun bir dizi doğru, daire veya elips olarak gruplamak genellikle önemsizdir. Hough dönüşümünün amacı, bir dizi parametreli görüntü nesnesi üzerinde açık bir oylama prosedürü uygulayarak kenar noktalarının nesne adaylarına gruplandırılmasını mümkün kılarak bu sorunu çözmektir.

III. Numpy Kütüphanesi

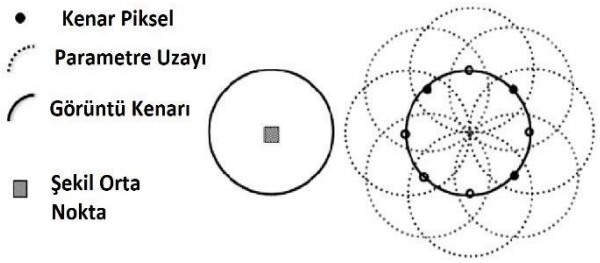
Python programlama dili için çok boyutlu dizileri destekleyen ve bu dizilerde işlem yapmamızı sağlayan güçlü bir kütüphanedir. NumPy'nin atası Numeric, ilk olarak Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla oluşturuldu. 2005 yılında Travis Oliphant, Numarray'in özelliklerini kapsamlı değişikliklerle Numeric'e dahil ederek NumPy'yi yarattı. NumPy OpenCV ile paralel çalışmaktadır. Ve OpenCV kütüphanesini kullanırken NumPy'ı da kullanmalıyız.

IV. OpenCV Kütüphanesi

OpenCV ilk olarak INTEL tarafından Haziran 2000 yılında yayınlanan ardından da günümüze kadar geliştirilen gerçek zamanlı bilgisayarlı görü teknolojisini barındıran açık kaynak kodlu bir programlama kütüphanesidir. Çoğunluk özelliğini C++ üzerinde barındırır. Ardından java, Python ve C#'a da eklenmiştir 2011'den itibaren GPU-CUDA üzerinden işlemler yapabilme özelliğine sahiptir. Derin öğrenme ile kullanılabilir.

V. Hough Çemberi Kısaca

Kusurlu görüntülerdeki daireleri tespit etmek için dijital görüntü işlemede kullanılan temel bir tekniktir. Daire adayları parametre uzayında oylama yapılarak sonuca varır. Oylama sonucunda istenilen değerlere uygun çemberler tespit edilir.



VI. Teori

Elimizde (K,L) merkezli çemberin yarıçapı r olmak üzere ve $t: 0 \rightarrow 2\pi$

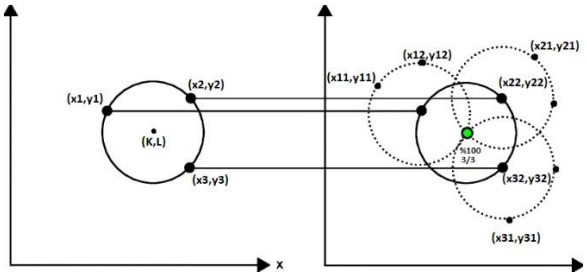
$$x = K + r \times \cos(t)$$
$$y = L + r \times \sin(t)$$

t açısı 360 dereceyi tamamladığında merkezi K ve L; kenarlarının noktaları ise X ve Y olan bir çember vardır. Bu sayısal değerler elimizdeki datadır. Yani dışarıdan gelen dijital bir görüntüdür. Elde edilen x ve y değerleri bir {x,y} biçiminde kaydedilir. Örnek; {x1,y1} , {x2,y2} , {x3,y3} (Esasında CannyEdge fonksiyonu ile kenar pikseller tespit edilir.)

Ardından (x?,y?) merkezli çemberin yarıçapı r olmak üzere ve $f: 0 \rightarrow 2\pi$

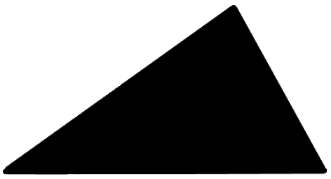
$$x?? = x? + r \times \cos(f)$$
$$y?? = y? + r \times \sin(f)$$

Olacak şekilde f açısı 360 dereceyi tamamladığında merkezi x? ve y?; kenarlarının noktaları ise x?? Ve y?? olan yeni çemberler çizilir. Aşağıdaki örnek görüntüde kenar üzerinden çizilen yeni çemberler tespit etmek istediğimiz (K,L) merkezli çemberin ortasından geçmektedir. Bu durumda oranlama yapılır. Örneğin 100 yeni çember çizilmiş ise ve 100 de orta noktadan geçiyor ise; doğruluk oranı %100'dür.

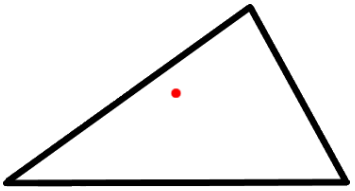


VII. Geometrik şekillerin tespiti

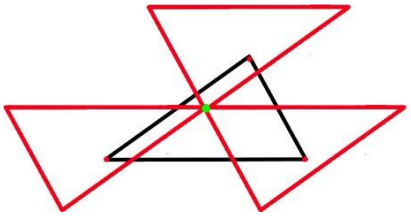
Elimizdeki dijital görüntüye (şekil a) CannyEdge (FindContours) Fonksiyonu uyguladıktan sonra (şekil b'de olduğu gibi) kenarları tespit edilir ve bir diziye kaydedilir. FindContours'dan elde ettiğimiz veriyi MinAreaRect fonksiyonuyla işlediğimizde elimizde şeklin yüksekliği, genişliği, orijine göre açısı ve orta noktası bulunur. Ve Şekil b'deki gibi çıktı elde edilir. Şekil b'de kırmızı nokta şeklin orta noktasıdır. Ardından da ana şeklin kenarları orta nokta olacak şekilde aradığımız şekil çizilir. Ve görüntü C elde edilir. Burada çizilen yeni şeklin kenarları ana şeklin orta noktasından geçiyor ise doğruluk oranı yüksek demektir. Düşük oranda geçiyor ise aranan şekil ile elimizdeki şekil uyumsuz demektir. Hemen hemen çoğu kapalı şekil için şekil tespit algoritması aynı yöntem ile çalışır.



Şekil a



Şekil b



Şekil c

VIII. OpenCV Fonksiyonları

A. RGB resmi HSV'ye çevirmek

R, G, B ∈ (0,255)

R' = R/255 G' = G/255 B' = B/255

R', G', B' ∈ (0,1)

Cmax = max(R', G', B') Cmin = min(R', G', B')

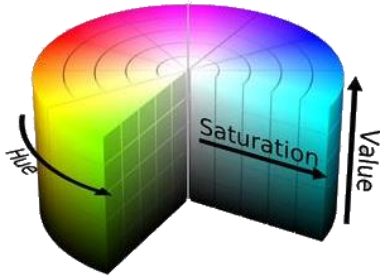
Δ = Cmax – Cmin

$$H = \begin{cases} \text{tanimsiz}, & \text{eger } MAX = MIN \\ 60\frac{G-B}{MAX-MIN} + 0, & \text{eger } MAX = R \text{ ve } G \geq B \\ 60\frac{G-B}{MAX-MIN} + 360, & \text{eger } MAX = R \text{ ve } G < B \\ 60\frac{B-R}{MAX-MIN} + 120, & \text{eger } MAX = G \\ 60\frac{R-G}{MAX-MIN} + 240, & \text{eger } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{eger } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{degilse} \end{cases}$$

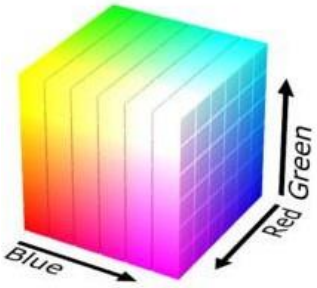
$$V = MAX$$

Yukarıdaki fonksiyon ile RGB resmimiz HSV'ye dönüşmüş olur.



HSV Uzayı

HSV (0-180,0-255,0-255)

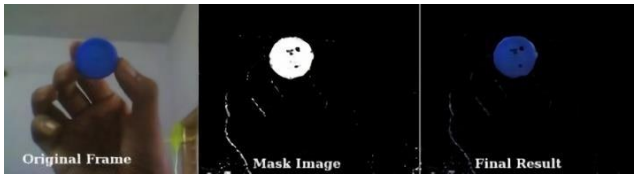


RGB Uzayı

RGB (0-255, 0-255, 0-255)

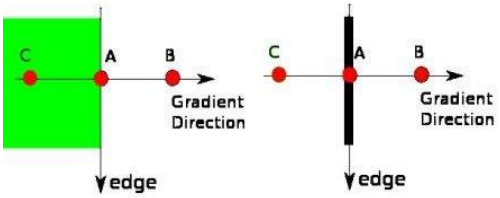
B. InRange ile belirli Renk aralığını bulmak

$$f(x) = \begin{cases} Pikseli\ kopyala, & Eğer\ (if) \\ \text{Alt aralık} < \text{H} < \text{Üst aralık} \\ VE(AND) \\ \text{Alt aralık} < \text{S} < \text{Üst aralık} \\ VE(AND) \\ Alt\ aralık < V < \text{Üst aralık} \\ Pikseli\ kopyalama, & değil\ ise\ (else) \end{cases}$$



Örnek işlem

C. CannyEdge ile kenar noktaları bulmak



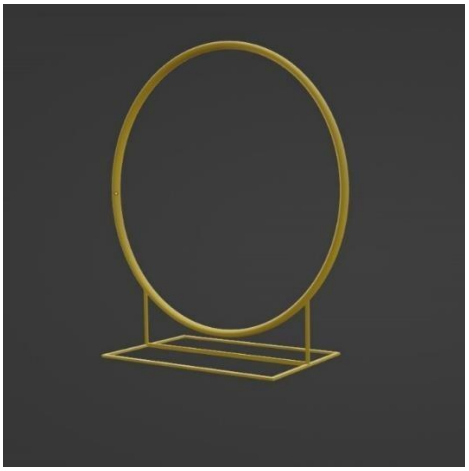
X ve Y ekseninde tarama yapılır. Tarama sonucunda pikseller arasında renk yoğunluk farkına göre o nokta Canny noktası olur. Ve tüm Canny noktaları bir dizide kaydedilir.

Örnek görsel

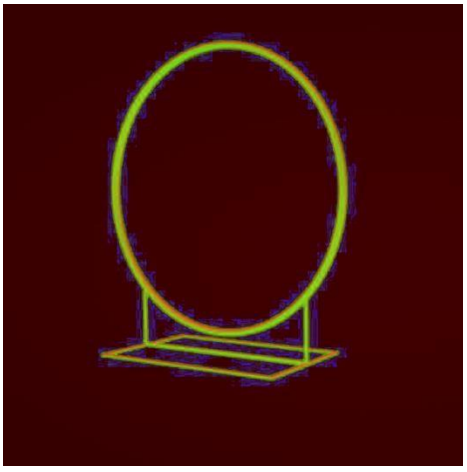
IX. Otonom Araçta Kullanılmasına Dair Bir Örnek

A. Düzgün Kapalı Şekil

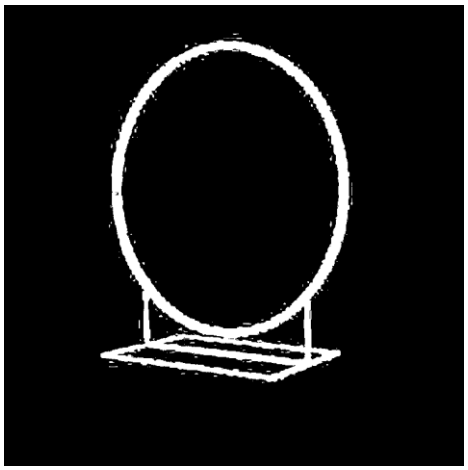
1) Çevre donanım birimleri aracılığı ile elde edilen dijital görüntü alınır.



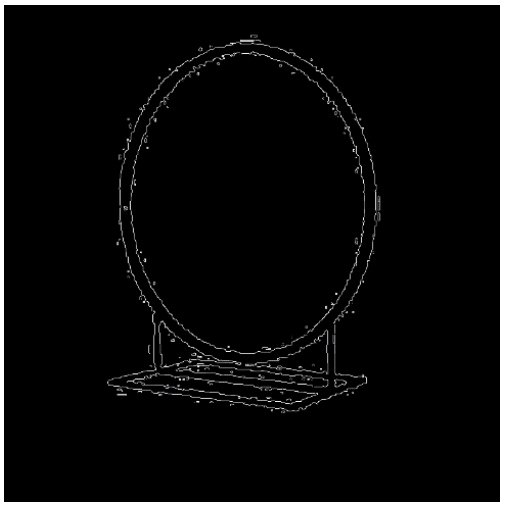
2) Alınan görüntü RGB → HSV'ye çevrilir ve yeni görüntümüz bu şekilde olur.



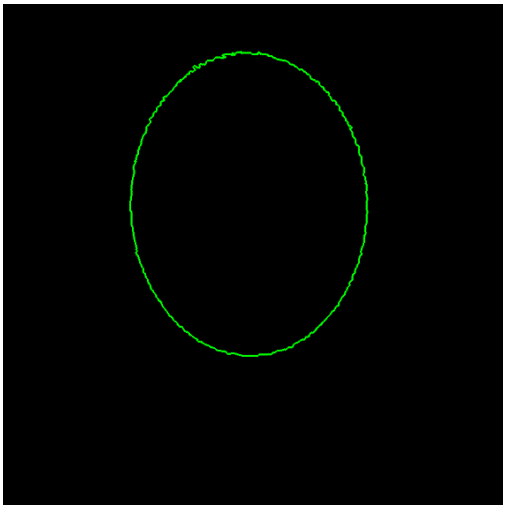
3) Ardından InRange ile istediğimiz renk aralığını seçiyoruz. (bu evrede gaussian filter kullanırsanız parazitler yok olacaktır).



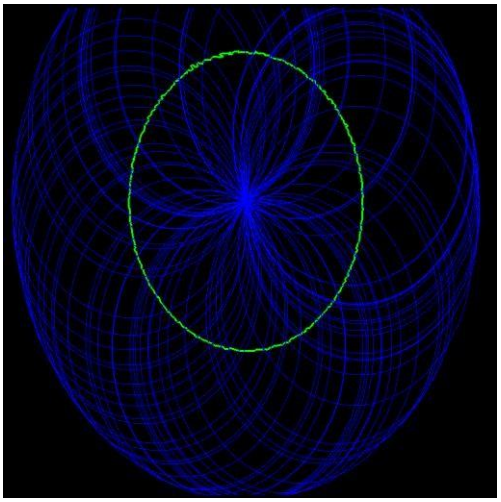
4) Ardından da canny işlemi ile kenarları belirtiyoruz. Buradan FindContours ile her bir kenar çizgisini ayrı bir seri içeriğine kaydediyoruz.



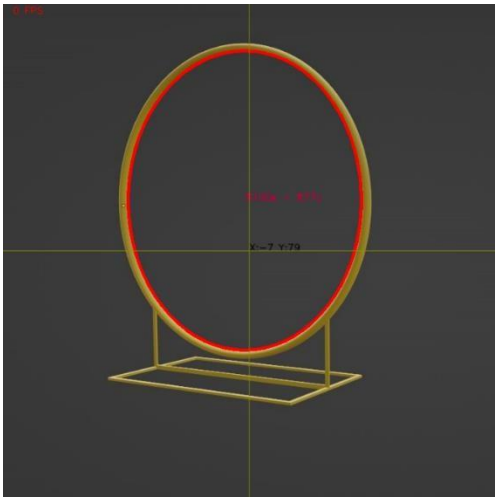
5) Her bir seriye değerleri kaydettikten sonra Hough çember tespiti algoritmasını kullanarak adım adım tarama yapacağımız seride işlemimizi yapıyoruz. Bu örnekte toplam 8 adet iç kontür ve 1 adet ebeveyn kontürü bulunur. Ayrıca bulunan her kontürün yüksekliği, genişliği, açısı, X ve Y konumu da MinAreaRect fonksiyonu ile bulunur. Bu örneğimizde



6) Her şekle yapılan taramanın sonucundan sadece bir tanesinin eliptik bir şekle sahip olduğunu hesaplamalar sonucunda varıyoruz.



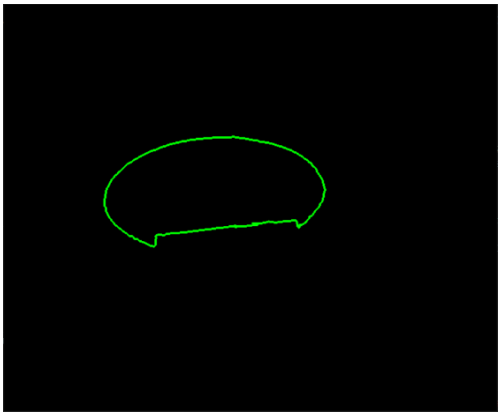
7) Ve son olarak çıktı görüntümüz bu şekilde



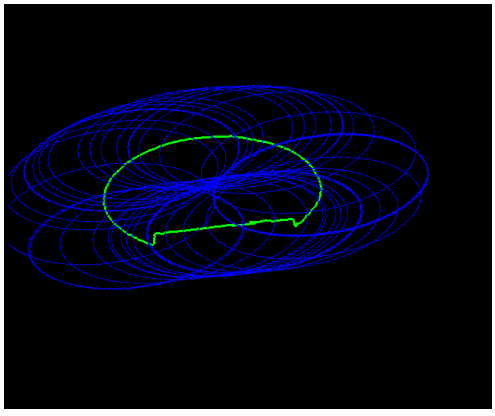
B. *Kesilmiş Elips*

Şimdiye kadarki anlattıklarımız kapalı yani bir tam düzgün elips tespiti için idi. Lakin ortadan 2'ye ayrılmış kapalı olmayan elips bu durumda tespit edilemez. Bundan ötürü Kontür birleştirme eylemini gerçekleştiriyoruz.

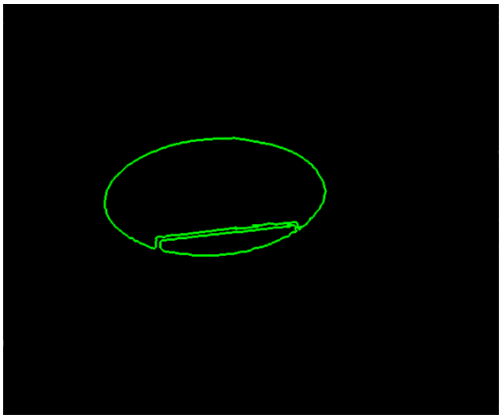
1) İlk önce bir parent konturunun içerisinde en büyük kontürümüzü seçiyoruz



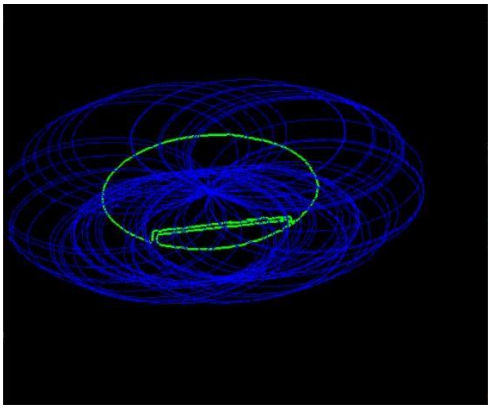
2) Seçilen kontürde tarama yapıyoruz tarama sonucunda doğruluk oranı %55'tir. Aradığımız değerin altındadır.



3) Bu durumda yeni bir algoritma eklenerek doğru sonuç arayışına girilir.
   En büyük iç kontür = A
   A'ya en yakın iç kontür = B
   A + B = C
   C yeni tarama yapacağımız kontür olur ve kontür işleminde hata payımız (düz çizgiler) biraz daha fazla olacağından hata payı toleransımızı arttırırız. Yani normalde en az %85 ise bu yöntemde %63'e kadar toleransımız olur.



4) C kontürüne yaptığımız tarama sonucunda değerimiz %64'tür. Bu da bizim için bir eliptik şekil bulduğumuz anlamına gelir



Referanslar

[1] Richard O. Duda and Peter E. Hart Stanford Research Institute, Menlo Park, California 1970 http://research.vuse.vanderbilt.edu/srdesign/2009/group2/p11-duda.pdf

[2] Rosenfeld, A. Picture Processing by Computer. Academic Press, New York, 1969.

[3] Hough, P.V.C. Method and means for recognizing complex patterns. U.S. Patent 3,069,654 18/11/1962.

[4] Griffith, A.K. Computer recognition of prismatic solids. Ph.D. Th., Dep. of Math., MIT, June 1970

[5] https://www.cis.rit.edu/class/simg782.old/talkHough/HoughLecCircles.html

[6] Sergei Azernikov. Sweeping solids on manifolds. In Symposium on Solid and Physical Modeling, pages 249–255, 2008.

[7] John Canny. A computational approach to edge detection. Pattern Analysis and Machine

[8] Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986.

[9]F. Mai, Y. Hung, H. Zhong, and W. Sze. A hierarchical approach for fast and robust ellipse

[10] extraction. Pattern Recognition, 41(8):2512–2524, August 2008.

[11]Thomas B. Moeslund. Image and Video Processing. August 2008.

[12]https://www.nature.com/articles/s41586-020-2649-2