Düzce Üniversitesi
Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Kabuk Zincir Kod Histogramı Kullanılarak Şekil Tanıma

Haydar TUNA [a,*], Recep DEMİRCİ [b]

[a] *Yönetim Bilgi Sistemleri Bölümü, Bilgi İşlem Daire Başkanlığı, M.E.B, Ankara, TÜRKİYE*
[b] *Bilgisayar Mühendisliği Bölümü, Teknoloji Fakültesi, Gazi Üniversitesi, Ankara, TÜRKİYE*
** Sorumlu yazarın e-posta adresi : haydartuna@meb.gov.tr*

ÖZET

Şekil bir nesnenin doğal sınırlarının insan beyninde oluşturduğu algı olarak tanımlanabilir. Şekil tanıma ise herhangi bir nesnenin ait olduğu sınıfın daha önceden karşılaşılan ve hafızaya kaydedilen algılarla karşılaştırılarak bulunmasıdır. Bilgisayarda şekil tanıma nesnelerin sınır veya bölge tabanlı şekil temsil yöntemleriyle elde edilen özellik vektörlerinin karşılaştırılarak sınıflarının tespitidir. Sınır tabanlı özellik çıkarma metotlarından biri olan zincir kodu sayısal görüntüdeki bir nesnenin sınır noktaları takip edilerek üretilen sembol dizisidir. İlgili sembol kümesinin elemanları her yön için daha önceden belirlenmelidir. Şekil temsilinde kullanılan zincir kodlarının en temel problemi ölçekleme veya döndürme işlemlerine karşı yeterince güçlü olmamalarıdır. Başka bir ifade ile şekiller ölçeklendiğinde ya da döndürüldüğünde zincir kodlarının uzunluklarının ve içeriklerinin değişmesidir. Bu nedenle şekillerin benzerlik karşılaştırılmasında farklı uzunluklardaki sembol dizisinden oluşan zincir kodları yerine, normalize edilmiş zincir kod histogramı tercih edilmektedir. Böylece sınır bilgileri sembol çeşidi ile orantılı olan sabit uzunlukta vektörlere dönüştürülerek benzerlik hesaplaması yapılmaktadır. Bu çalışmada nesnelerin sınır noktalarında bulunan piksellerin kabuk numaraları kullanılarak ölçeklenme ve döndürme işlemlerine karşı dayanıklı yeni bir zincir kod histogramı önerilmiştir. Önerilen yöntemin döndürmeye karşı duyarlılığı Freeman 8 (FR8) zincir kod histogramıyla deneysel olarak karşılaştırılmış ve elde edilen sonuçlar verilmiştir.

Anahtar Kelimeler: *Kabuk Zincir Kodu, Zincir Kodu, FR8*

Shape Recognition Using Histogram of Shell Chain Code

ABSTRACT

Shape can be defined as perception of natural boundaries of an object in human brain. Shape recognition is determination of class to which the object belongs by comparison of the object perceptions which are previously encountered and stored in memory. In the computer based shape recognition, it is determination of class to which the object belongs by comparison of the feature vectors which are obtained by contour or region-based methods. The chain code, which is one of the contour-based feature extraction methods, is a sequence of symbols created by following the boundary of an object. The elements of symbol set must be predefined for each direction. The fundamental issue with chain code used for shape description is that they are not robust enough for scaling and rotation. In other words, the length and content of chain code may change when shapes are scaled or rotated. Therefore, in compassion process of shapes, normalized chain code histograms are preferred rather

Geliş: 30/11/2018, Düzeltme: 28/12/2018, Kabul: 30/12/2018

than chain codes with different lengths. Consequently, similarity calculations are performed by means of feature vectors of which the fixed lengths are proportional with symbol types. In this study, a new chain code which is strong for scaling and rotation has been proposed. The shell numbers where the object boundary pixels are located has been used to generate the chain code. The rotational robustness of the proposed chain code has been experimentally compared with outputs of Freeman 8 (FR8) chain code histogram and the obtained results have been given.

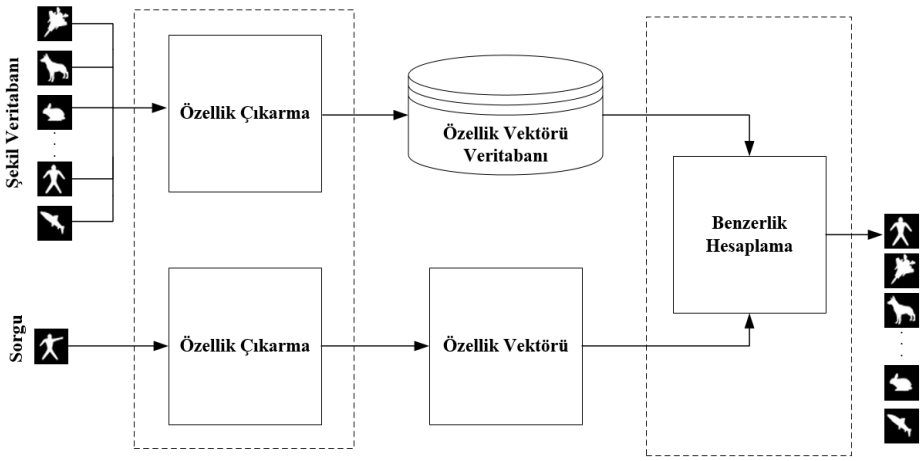
Keywords: *Shell Chain Code, Chain Code, FR8*

I. GİRİŞ

Şekil bir nesnenin doğal sınırları olarak tanımlanabilir. Söz konusu doğal sınırlar insan beyni tarafından yorumlanıp algıya dönüştürülür. İnsan beyni ise ilk defa karşılaştığı bir nesnenin sınıfına daha önce karşılaştığı nesnelere bakarak karar verir. Eğer nesne daha önceden karşılaşılan bir sınıfa benzemiyorsa yeni bir sınıf oluşturulur. Bilgisayarda da insan beynin yaptığına benzer bir süreç vardır. Şekil 1 de görüldüğü gibi tipik bir bilgisayarlı nesne tanıma sisteminde bir adet şekil veri tabanına ihtiyaç vardır. Şekil veri tabanındaki nesnelere ait bilgiler şekil temsil yöntemleri vasıtayla özellik vektörlerine dönüştürülür. Her hangi bir şekil sorgulanmak istendiğinde öncelikle şekil temsil yöntemleriyle özellik vektörü bulunur ve akabinde veri tabanında bulunan diğer nesnelerin özellik vektörleri ile karşılaştırılır. Karşılaştırma işlemi vektörler arasındaki Öklid mesafesi gibi ölçütlere bakılarak yapılabilir. İşlem sonunda sorgulanan özellik vektörüne en yakın nesne sınıfı tespit edilir. Şekil 1 de gösterilen şekil tanıma sistemin en kritik süreci özellik çıkarma safhasıdır. Sorgulanan veya veri tabanına kaydedilen şekillerin doğru temsil edilmesi sistemin şekil tanıma başarısını direkt olarak etkilemektedir. Özellik vektörünün oluşturulmasında birçok yaklaşımlar olmasına rağmen, hesap maliyeti açısından en avantajlı olanı zincir kodudur. Zincir kodu bir nesnenin sınır noktalarındaki pikseller arasındaki yön geçişleri esasına dayanan yöntemdir. Zincir kodu fikri ilk olarak 1961 yılında Freeman tarafından ortaya atılmıştır [1]. Freeman yönteminde yön geçişi olarak bir pikselin 4 ve 8 komşulukları esas almıştır [2]. Müteakip yıllarda, Papert bir pikselin sol ve sağ tarafındaki komşuluklarını kullanan daha basit bir zincir kod tekniğini önermiştir [3]. Cruz ve Dagnino ise pikseller arasındaki ortogonal yön geçişlerini temel alan 3OT (Three Orthogonal) olarak adlandırdıkları farklı bir zincir kod tekniğini geliştirmişlerdir [4]. Nunes ve arkadaşları ise bir önceki piksele göre yön değişimini esas alan ve zincir kodundaki sembolleri azaltan fark zincir kodunu önermişlerdir. Söz konusu zincir kodu R, L, S olmak üzere üç sembolden meydana gelmektedir [5]. Brisbane ise VCC (Vertex Chain Code) olarak adlandırdığı zincir kodu yaklaşımında, piksellerin birbirine temas etmesiyle oluşan köşegen farklılıklarını temel almıştır [6]. Yukarıda verilen çalışmalara ilave olarak, Lui ve Zalik Freeman 8 zincir kod sembollerini bir önceki pikselden geliş yönüne göre 45^0 ve katlarını kullanarak belirlemişlerdir [7].

Genel olarak değerlendirildiğinde, zincir kodları sınır tabanlı yöntemler olduklarından gürültüden etkilenmektedirler. Ayrıca başlangıç noktası değişimi zincir kodunda farklılığa neden olduğundan, başlangıç noktasına da bağımlıdırlar. Diğer taraftan zincir kodu başlangıç noktasına göre normalize edilirse, oluşan farklılık ortadan kaldırılır [8]. Zincir kodlarında karşılan bir diğer önemli sorun ise, döndürme ve ölçeklendirmenin elde edilecek zincir kodlarını değiştirmesidir [9]. Başka bir ifade ile herhangi bir şekil döndürüldüğünde ya da ölçeklendiğinde zincir kod uzunluğunun ya da sırasının değişimidir ki bu da doğrudan karşılaştırma işlemini zorlaştırmaktadır. Ancak herhangi bir şeklin zincir kod histogramı alınırsa, meydana gelen farklılık giderilerek ve sabit uzunlukta vektöre

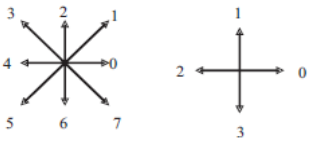
dönüştürülebilir. Sonuçta uzunlukları farklı zincir kodları karşılaştırılabilir hale gelmektedir [10]. Buna rağmen bazı durumlarda zincir kod histogramı döndürme kaynaklı problemlerin çözümünde yetersiz kalabilmektedir. Örneğin Freeman zincir kodu 90^0 ve katlarındaki döndürmelerden etkilenmez iken, diğer açı değerlerinde sonuçlar değişilebilmektedir. Zincir kodları döndürme etkilerinin az olduğu karakter tanıma uygulamalarda daha çok tercih edilmektedir [11]. Zincir kod histogramları zincir kodlarının kullanımındaki problemleri kısmen azalmış ise de, bazı uygulamalarda şekil ayırt edebilme kabiliyetleri yetersiz kalabilmektedir. Ayırt edicilik birbirinden farklı iki nesnenin zincir kod histogram vektörleri arasındaki mesafenin çok düşük olmasıdır ki bu durum doğru olmayan sınıf tasnifine neden olmaktadır. Ayırt edicilik problemi için önerilen bölge tabanlı şekil temsil yöntemlerinden biride kabuk histogramıdır. Söz konusu teknikte bir şekil öncelikle ağırlık merkezinden itibaren kabuk olarak adlandırılan eşit yarıçaplı halkalara ayrılır ve her bir kabuğa sıfırdan itibaren bir etiket veya numara verilir. Sorgulanan şeklin her bir kabuk içerisindeki pikselleri sayılarak histogram oluşturulur [12]. Bu çalışmada ise sorgulanan şeklin her bir kabuk içeresinde kalan sınır pikselleri dikkate alınmıştır. Referans olarak seçilen bir başlangıç noktasından itibaren sınır piksellerinin kabuk numaraları yan yana dizilerek zincir kodu üretilmiştir. Bir sonraki adımda elde edilen kabuk zincir kodunun histogramı (KZKH) kullanılarak şekil tanıma sistemi geliştirilmiştir.



Şekil 1. *Şekil tanıma sistemi*

II. FREEMAN8 ZİNCİR KOD HİSTOGRAMI

Freeman zincir kodunun temel prensibi sınır noktaları arasındaki yön geçişleridir. Yön geçişleri bir referans noktasından başlanarak tekrar aynı noktaya gelinceye kadar devam eder. Tekrar referans noktasına gelindiğinde ise zincir kodu elde edilmiş olur. Freeman pikseller arasındaki yön geçişlerinin Şekil 2 deki gösterildiği gibi 8 veya 4 komşuluğunda olması durumuna göre iki farklı zincir kodu üretim tekniği önermiştir. Üretilecek zincir kodunda 4 piksel komşuluğunun esas alınması durumunda {0, 1, 2, 3} sembolleri (FR4), 8 komşuluğunun esas alınması durumunda {0, 1, 2, 3, 4, 5, 6, 7} sembolleri (FR8) kullanılır [1, 2].



Şekil 2. *Freeman zincir kodu yön sembolleri*

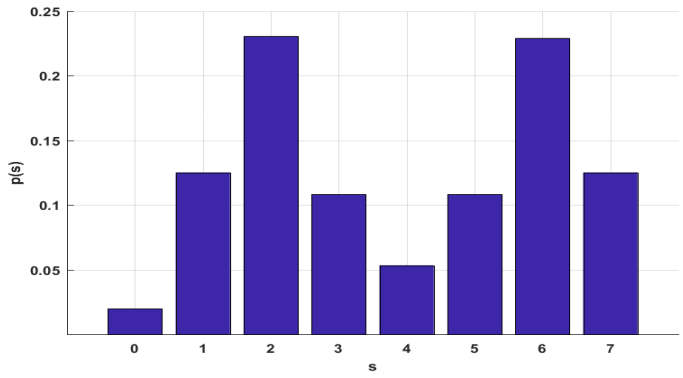
Daha önce belirtildiği gibi herhangi bir şeklin zincir kodunun dizilimi başlangıç veya referans noktasına göre değişmektedir. Bununla birlikte başlangıç noktasının sıfır derece seçilmesi genel eğilimdir. Örneğin olarak Şekil 3 de gösterilen tilki kafası şekli için 631 birim uzunluğunda zincir kodu (FR8) üretilmiştir. Zincir kod üretme işlemi sıfır dereceden başlanarak 8 komşuluk üzerinden saat ibresinin tersi yönünde pikseller arasında geçişler yan yana yazılarak gerçekleştirilmiştir. Her farklı şekil veya aynı şeklin ölçeklenmiş ya da döndürülmüş durumları için farklı zincir kodunun üretileceği açıktır ki buda şekil benzerlik ölçümlerinde bir takım ilave hesap maliyetlerine neden olmaktadır. Şeklin türünden, boyutundan veya konumundan bağımsız özellik vektörüne ihtiyaç duyulur. Bu nedenle herhangi bir şeklin zincir koduna ait bilgiler

$$p(s)=\frac{n_s}{n} \tag{1}$$

ifadesi ile histograma dönüştürülür. Eş. 1 deki n_s *s.* yön sembolünün zincir kodundaki tekrar sayısını, *n* zincir kod uzunluğunu, *p(s)* ise *s.* yön sembolünün zincir kodu içinde bulunma olasılığını göstermektedir. Görüldüğü gibi sembol tekrar sayıları zincir kod uzunluğu ile normalize edildiğinden, elde edilen özellik vektörünün boyutu sabit kalacaktır. Şeklin yapısına ve oryantasyonuna göre sadece özellik vektörü, *p(s)* nin elemanları değişecektir. Şekil 3 gösterilen tilki kafasına ait zincir kodunun histogramı Şekil 4 verilmiştir. İlgili zincir kodunda 8 farklı sembol kullanıldığından yatay eksende 8 farklı seçenek, dikey eksende ise 0-1 aralığına normalize edilmiş dağılımlar görülmektedir. Herhangi bir şekil 90 derece ve katları dışında döndürüldüğünde FR 8 zincir kodu değişeceği için zincir kodu histogramı da değişecektir. Bu özellik Freeman zincir kodunun en belirgin dezavantajını oluşturmaktadır.

3223232232323232332322333333321222122222122222221222222222222222223222223223223232334454454545545545
5554555655123455455454554555554554433433333433434334334567332333433333433433434344344556565656565666656
6666666666666667666666667666667666765555555665565565656565665656656656665666566656666656666666711212121
1211211112656666666666666666666666676666666766701222222222122216676777677677767777777777077077707123232
3232323267767767767777677767777677677677776770011211111211211211112111211112112112656565655656710111
1011101111111112111121121121227666766666666670122122222221222222222222222222222232677776776776767677112
2222222232222232222322232 Uzunluk = 631

Şekil 3. *Tilki kafası ve zincir kodu (631 birim)*



Şekil 4. *Tilki kafası ve FR8 zincir kod histogramı*

III. KABUK ZİNCİR KOD HİSTOGRAMI

Şekil tanıma işlemi için günümüze kadar önerilen zincir kodu yaklaşımlarının en tipik problemi döndürmeye karşı aşırı duyarlı olmalarıdır. Zincir kodları sınır tabanlı yaklaşım olup, sadece nesneyi oluşturan sınır piksel bilgileri dikkate alınmaktadır. Diğer taraftan bölge tabanlı yaklaşımlarda ise nesneyi temsil eden bütün pikseller şekil tanıma süreci içinde değerlendirilmektedir. Örneğin kabuk histogramı yaklaşımında herhangi bir şekil ağırlık merkezi etrafına eşit aralıklarla çizilen çemberler yardımıyla alt bölgelere ayrılır. Her bir bölgeye bir indis ya da etiket atanarak halka içeresinde kalan pikseller yardımıyla histogram oluşturulur. Söz konusu algoritmada kabuk ya da halka sayısı önemli bir parametredir. Bahsedilen halka veya kabuklar bir şeklin ağırlık merkezinden itibaren en büyük yarıçap eşit aralıklara bölünerek elde edilir. Bir şekle ait kabuklar tespit edilir iken yapılacak ilk işlem, nesnenin ağırlık merkezi ile en büyük yarıçap arasındaki Öklid uzaklığının 0-1 aralığına

$$D_i = \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \tag{2}$$

şeklinde normalize edilmesidir. Eş. 2 de d_i nesne üzerindeki i. piksel ile ağırlık merkezi arasındaki Öklid mesafesini, d_{min}, d_{max} ise sırasıyla minimum ve maksimum yarıçapları, D_i ise i. noktanın normalize edilmiş Öklid mesafesini göstermektedir. Bir şeklin normalize edilmiş en büyük yarıçapının kuantalaması ile kabuklar veya halkalar elde edilir. Söz konusu durumda en kritik parametre bir şekli tanımlamak için kaç tane hakla ya da kabuk kullanılmasının gerekliliğidir. Örneğin Şekil 3 de verilen tilki kafası şekli için 8 adet halka veya kabuk seçildiğinde elde edilen etiketler Şekil 5 de gösterilmiştir. Ankerst ve arkadaşları tarafından önerilen yaklaşımda kabuk sayıları için farklı öneriler yapılmıştır [12]. Ankerst ve arkadaşlarının önerdiği kabuk histogramı yaklaşımında her bir hakla içersinde kalan piksel sayıları kullanılarak tanımlama yapılmıştır. Başka bir ifade ile bölge tabanlı bir yaklaşım önerilmiştir.

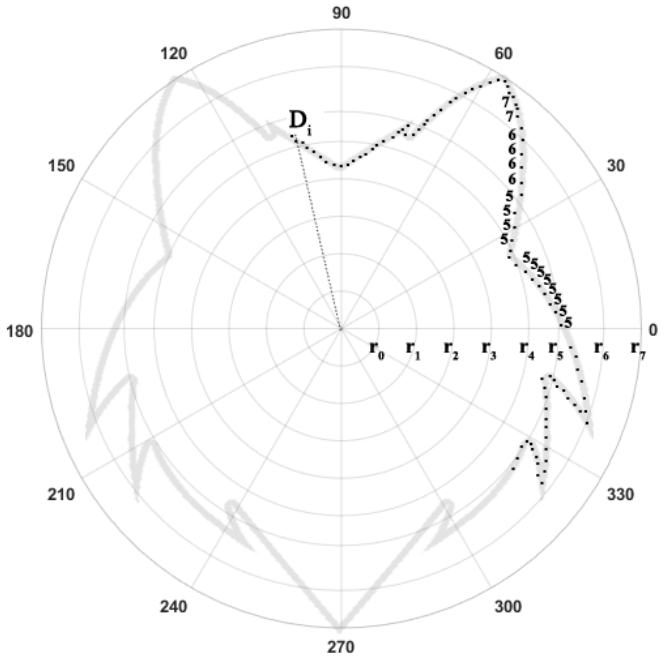
Ankerst ve arkadaşlarının önerdiği yöntem sadece bölge tabanlı iken, bu çalışmada önerilen algoritmada ise her bir hakla içesinde kalan ve sadece nesne sınırlarındaki pikseller dikkate alınmıştır. Bu açıdan bakıldığında hem bölge hemde sınır bilgileri kullanıldığından hibrit bir algoritma elde edilmiştir. Geliştirilen kabuk zincir kodu (KZK) algoritmasında, nesnenin sınır noktalarda yer alan noktaların kabuk etiketleri bir başlangıç noktasından itibaren yan yana yazılarak Şekil 5 deki gibi

kabuk zincir kodu elde edilmiştir. Görüldüğü gibi zincir kodu elde edilirken kullanılacak sembol sayısı kabuk sayısı ile orantılıdır. Freeman zincir kodunda ise sembol sayısı 4 veya 8 farklı sembol ile sınırlıdır. Kabuk sayısı artırılarak KZK zincir kod sembol çeşidi ve dolaysısıyla hassasiyeti artırmak mümkündür. Bu yüzden önerilen algoritma ile zincir kod üretimine esneklik getirilmiştir. Önceki bölümde test edilen tilki kafası şeklinin 8 kabuk kullanılarak elde edilmiş zincir kodu Şekil 6 da verilmiştir. Freeman zincir kodunda olduğu gibi, önerilen yöntemde de zincir kod histogramı üretmek mümkündür. Buna göre nesnelerin sınır piksellerinin *k* nolu kabukta bulunma olasılığı

$$p(k) = \frac{n_k}{n} \tag{3}$$

şeklinde ifade edilebilir. Burada n_k *k.* kabuktaki sembol sayısını ve *n* kabuk zincir kod uzunluğunu göstermektedir. Örneğin tilki kafası şeklinin 8 kabuk kullanılarak elde edilen histogramı Şekil 7 de gösterilmiştir. Daha önce belirtildiği gibi önerilen yöntemde zincir kodu sembol çeşidi esnek olup, kabuk sayısı ile orantılıdır. Kabuk zincir kodunda semboller sıfırdan başlanarak birinci kabuk için 0, ikinci kabuk için 1 sembolü olacak şeklinde verilir. Sembol sayısı 11 ve üzerinde olduğunda ise alt kabuk numaraları ile karışıklığı önlemek için harf kullanılmıştır. Örneğin 16 kabuk kullanıldığında 11. kabuk ve sonrası için {a, b, c, d, e, f} sembolleri kullanılmıştır [14]. Tilki kafası şeklinin 16 kabuk kullanılarak elde edilen zincir kodu ve histogramı ise sırasıyla Şekil 8 ve Şekil 9 de gösterilmiştir. Her iki zincir kodunda kod uzunluğunun değişmemesine rağmen sembol çeşidinin arttığı görülebilir.

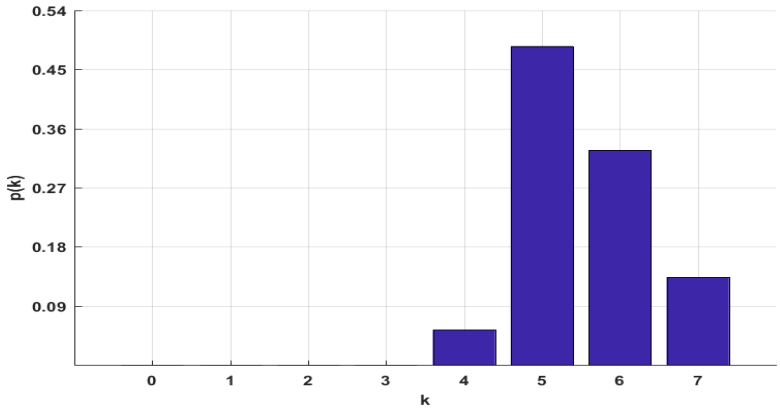
Zincir kodunu temsil eden sembol kümesinin elemanlarını tamamen alfabenin harflerinden seçmekte mümkündür. Böylece zincir kodlarını metin işleme algoritmaları veya sonlu durum makinaları kullanarak değerlendirmekte mümkün olacaktır. Ayrıca KZKH da bir şekil döndürülse bile sınır piksellerde yer alan noktaların kabukları değişmeyeceği için histogramda önemli bir değişiklik olmayacaktır. Şekil ölçeklendiğinde ise noktaların merkez ile arasındaki Öklid uzaklıkları normalize edilerek hesaplandığı için histogram vektörleri de aynı olacaktır.



Şekil 5. *Kabuk zincir kodunun üretilmesi*

555555555555555555544444444444444455555555555555556666666666666666777777777777777777777777776666
666666555555555555555555544444444444444455555555555555556666666666777777777777777777777777777666
666666666666665555555555555554444444444444445555555555555555555555556666666666666666666666666666
655555555555555555555555555555566666666666666666666666655555555555555555555555555556566666666666655
555555555555555555666666666666677777777777777777777777766666666666665555555555555555555566666666
666565555555555555555555555555555566666666666666666666665555555555555555555555555556666666666
66666666666666666665555555 Uzunluk = 631

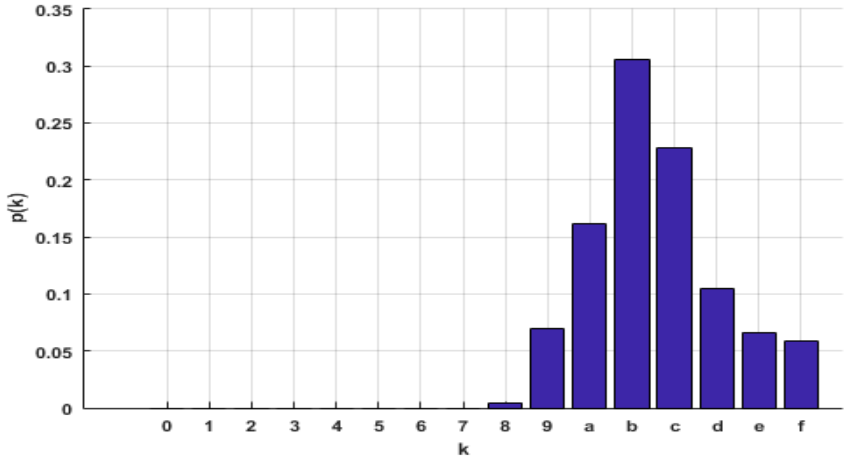
Şekil 6. *Tilki kafası ve kabuk zincir kodu: KZK (8 kabuk)*



Şekil 7. *Tilki kafası ve kabuk zincir kod histogramı: KZKH (8 kabuk)*

bbbbbbaaaaaaaaaaaaaa9999999999999999aaaaaaaaabbbbbbbbccccccccddddddddeeeeeeeeffffffffffffeeeeeeddddddccccccbbbb
bbbbbbbbaaaaaaaa999999888999999aaaaaaaabbbbbbbbbbbbccccddddeeeeeeffffffffffffeeeeeeeeddddddddccccccccbbbbb
bbbaaaaaaaaa9999999999999999aaaaaaaaaaaaaaabbbbbbbbbbbbccccccccccccddddddddccccccccbbbbbaaaaaaaaaaaaaabbbbb
bbbbbbbccccccccccccddddccccccbbbbbbbabbbbbbbbbbbbbbbbbbbbbbbbcbccccccccccbbbbbbbbaaaaabbbbbbbbccccccccddd
dddeeeeeeeffffffffffeeeeeeddddddccccccccbbbbbbbbaaaaabbbbbbbbccccccccccbcbbbbbbbbbbbbbbbbbbbbbbabbbbbbbccc
ccccddddccccccccccccbbbbbbbbbbbbaaaaaaaaaaaaaabbbbbccccccccddddddddccccccccccccbbbbbbb Uzunluk = 631

Şekil 8. *Tilki kafası ve kabuk zincir kodu: KZK (16 kabuk)*



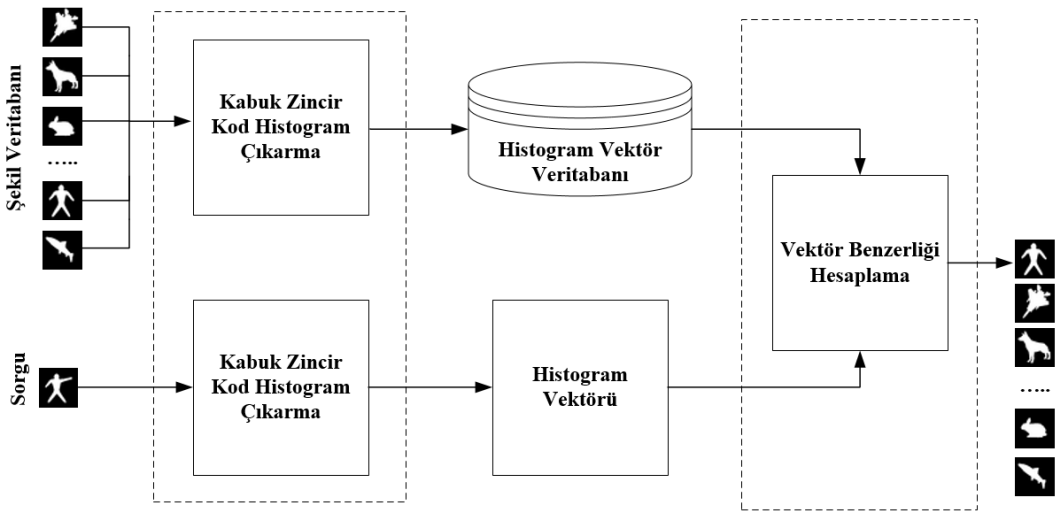
Şekil 9. *Tilki kafası ve kabuk zincir kod histogramı: KZKH (16 kabuk)*

IV. KABUK ZİNCİR KOD ile ŞEKİL TANIMA

Şekil 1 verilen şekil tanıma sisteminin de araştırmacıların değişiklik yaptığı kısımlar genelde özellik çıkarma ve benzerlik hesaplama bloklarında olmuştur. Özellik çıkarma veya şekil tanımlama olarak adlandırılan kısım yetersiz olursa, zaten sistemin genel başarımı da düşük olmaktadır. Özellik çıkarma bloğu yerine bu çalışmada kabuk zincir kod histogramı algoritması konulmuştur. KZKH histogramı ile şekil tanıma işlemi Şekil 10 da görüldüğü gibi iki aşamada gerçekleştirilir. Birinci aşamada şekil veri tabanında bulunan nesnelerin kabuk zincir kodu bulunup histogram vektör veri tabanı oluşturulur. Oluşturulan histogram veri tabanında nesnelere ait özellik vektörleri ve sınıfları birlikte tutulmaktadır. İkinci aşamada ise sorgulanacak nesneye ait kabuk zincir kodu bulunup histogram vektörüne dönüştürülür. Tanımanın en kritik aşaması olan benzerlik hesaplaması için ise farklı çözüm önerileri olmasına karşılık, en yaygın kullanılan yaklaşım özellik vektörleri arasındaki Öklid mesafesinin

$$d\left(\vec{u},\vec{v}\right)=\sqrt{(u_1-v_1)^2+(u_2-v_2)^2+\ldots(u_m-v_m)^2} \tag{4}$$

şeklinde hesaplanmasıdır. Eş. 4 de *u* sorgulanan nesneye ait özellik vektörünü, *v* ise veri tabanında kayıtlı bulunan nesnelerin özellik vektörlerini temsil etmektedir. Bütün vektörlerin boyutu *m* olup, özellik sayısını belirtmektedir. Freeman 8 (FR8) algoritmasında *m* 8 iken, önerilen kabuk zincir kod histogramı yaklaşımında *m* kabuk ya da halka sayısına eşittir. Özellik vektörler arasındaki uzaklık bulunduktan sonra, sorgulanan nesne ile diğer nesneler arasındaki Öklid uzaklıkları küçükten büyüğe doğru sıralanır. Böylece sorgulanan nesneye en çok benzeyen nesne sınıfları sırasıyla tespit edilir.



Şekil 10. *Kabuk zincir kod ile şekil tanıma*

V. DENEYSEL SONUÇLAR ve YORUMLAR

Şekil 1 genel yapısı verilen şekil tanıma sisteminde görüleceği gibi şekil veya şekillere ait özellik veri tabanı olmadan şekil tanıma mümkün değildir. Temelde insanların nesne tanıma süreci de benzerdir. Yani hayatımız hiç görmediğimiz veya ilk defa karşılaştığımız bir nesneyi tanımakta veya tasnif yapmakta hayli zorlanırız. Daha önce görmüş olduğumuz ve bir takım nitelikleri hafızamızda kalan nesnelere benzeterek sınıflandırma veya tasnif yaparız. Dolaysısıyla bilgisayar destekli nesne tanıma yapabilmek içinde daha önceden oluşturulmuş nesne veri tabanına ihtiyaç duyulur. Yeni geliştirilen algoritmaların başarımı referans olarak kabul edilen bir takım şekil veri tabanları kullanılarak test edilmektedir. Bu çalışmada önerilen algoritmayı test etmek için orijinali Şekil 11 de verilen Kimia 216 şekil veri tabanı kullanılmıştır. Ayrıca algoritmanın döndürme durumundaki başarımını test edebilmek için Kimia 216 veri tabanındaki şekiller rastgele döndürmeye tabii tutularak Şekil 12 de gösterilen ikinci bir veri tabanı elde edilmiştir. Kimia 216 veri tabanında 18 farklı nesne sınıfı ve her bir sınıfta 12 nesne mevcuttur [13]. Test için aynı sınıftan 12 nesne şekli kullanılması başlangıçta anlamsız gelse bile, gerçekte çalışan veya uygulanabilen sistem tasarımı için önemlidir. İki boyutlu görüntü kullanılarak şekil tanıması yapılmaktadır. Etrafımızdaki nesneler ise üç boyutludur. Üç boyutlu bir nesnesinin fotoğrafı çekildiğinde matematiksel olarak üç boyutlu uzaydan iki boyutlu uzaya iz düşüm yapıldığından ve kameranın nesneye bakış açısından dolayı bozulma, kopma veya bilgi kaybı kaçınılmazdır. Ayrıca yukarıda bahsedilen nedenlerden dolayı hiçbir bilgi kaybolmasa bile tanınacak nesnedeki küçük değişiklikler onun sınıfını değiştirmeyecektir. Örneğin fil şeklini dikkate alalım. Nesne tanıma algoritmaları filin bütün organları tam iken tanıma yapabiliyorsa, filin bir ayağı, kuyruğu ya da herhangi bir dişi eksik olduğunda da yanı sınıfa atama yapabilmelidir. Bir kolunu kaybetmiş bir insan, insan sınıfından çıkmayacağı gibi bir dişini kaybetmiş filde, fil sınıfından çıkmayacaktır. Uygulamaya dönük bir örnek ise askeri hedef tanıma sistemleri verilebilir. Her zaman üsten ya da yandan fotoğrafı çekilmiş tank veya gemi hedeflerini bulmak mümkün değildir. Hedef tanıma yazılımlarının her türlü şartlarda çekilmiş fotoğraflardan sonuç çıkarabilmesi hayati öneme sahiptir. Diğer taraftan beynimizin algı sistemi bu ayrımı yapabilecek yetenektedir. Bu durum insan algısına yakın kabiliyetleri olan şekil tanıma sistem ve yazılımları geliştirmek için dikkate almamız gereken önemli bir husustur. Dolaysısıyla Kimia 216 veri tabanına şekil temsil metotlarının

başarımını test edebilmek için kopma, bükülme veya döndürme gibi çeşitli bozulmalar eklenmiştir. Söz konusu gürültüler nesnelerin tanınmasını güçleştirip gerçek ortama yakın bir test ortamı sunmaktadır.

Kimia 216 veri tabanında görüldüğü gibi 18 farklı sınıf ve 216 farklı şekil mevcuttur. Sorgulama ise her seferinde bir adet şekil aracılığı ile yapılmaktadır. Her sınıfta 12 adet şekil olduğu için her sınıf için 12 adet sorgulama yapılmış ve sorgulan şekil hariç en çok benzeyen ilk şeklin sınıfı kaydedilmiştir. İdeal şartlarda bütün sorgulanan nesnelerin aynı sınıfta olması beklenir. Ancak söz konusu durum pratikte pek mümkün olmadığından, başarımın nümerik olarak değerlendirilmesi için ilave ölçütlere ihtiyaç duyulur. Şekil tanıma süreci çok sınıflı bir sınıflandırma problemidir. Çok sınıflı sınıflandırıcıların performansını değerlendirmek için Şekil 13 deki gösterilen genelleştirilmiş karışıklık matrisi en çok tercih edilen metottur. Karışıklık matrisi GxG boyutunda bir kare matristir ve satır indisleri sorgulanan nesnelerin gerçekteki sınıflarını, sütun indisleri ise sorgulama sonucunda elde edilen sınıfları göstermektedir. Matrisin köşegeninde yer alan hücreler doğru olarak sınıflandırılan nesne adetlerini gösterirken, köşegen dışında kalan tüm hücreler yanlış sınıflandırılan nesne sayısını ifade eder [15]. Deney setindeki toplam örnek sayısı matristeki tüm hücrelerin toplamı alınarak

$$N = \sum_{g=1}^{G} \sum_{k=1}^{G} s_{gk} \tag{5}$$

şeklinde bulunur [15]. Eş. 5 de G sınıf sayısını, g gerçekteki nesne sınıflarını, k tahmin edilen nesne sınıflarını ve s_{gk} matristeki hücreleri göstermektedir. Ayrıca g numaralı nesne sınıfının toplam örnek sayısı

$$n_g = \sum_{k=1}^{G} s_{gk} \tag{6}$$

gibi hesaplanır[15]. Eş.6 daki s_{gk} gerçekte g sınıfında iken sorgu sonucu k sınıfında tahmin edilen örnek sayısını göstermektedir. Sorgu sonucunda g. sınıfta tahmin edilen örnek sayısı ise sınıfın bulunduğu sütundaki matris elemanları toplanarak

$$n'_g = \sum_{k=1}^{G} s_{kg} \tag{7}$$

şeklinde bulunur[15]. Eş. 7 deki s_{kg} ise gerçekte k sınıfında iken, g sınıfına atanan nesne sayısını ifade etmektedir. Karışıklık matrisi ile sınıf temelli performans değerlendirilmesi yapılabildiği gibi algoritmanın global düzeyde performansının değerlendirmesi de yapılabilir. Sınıf düzeyli değerlendirme için bir sınıfın kesinlik (Precision:P) ve hassasiyet (Recall:R) ölçütleri sırasıyla

$$P_g = \frac{s_{gg}}{n'_g} \tag{8}$$

$$R_g = \frac{s_{gg}}{n_g} \tag{9}$$

şeklinde hesaplanır [15]. Eş. 8 de P_g g. sınıfın kesinlik metriğini, s_{gg} g. sınıfta doğru olarak tahmin edilen örnek sayısını ve n'_g g. sınıfta tahmin edilen toplam örnek sayısını ifade etmektedir. Eş. 9 daki R_g , g. sınıfın hassasiyetini ve n_g gerçekte g. sınıfta yer alan örnek sayısını göstermektedir. Algoritmanın bütün sınıfları da içeren genel başarısını veya doğruluğunu (Accuracy:A_G)

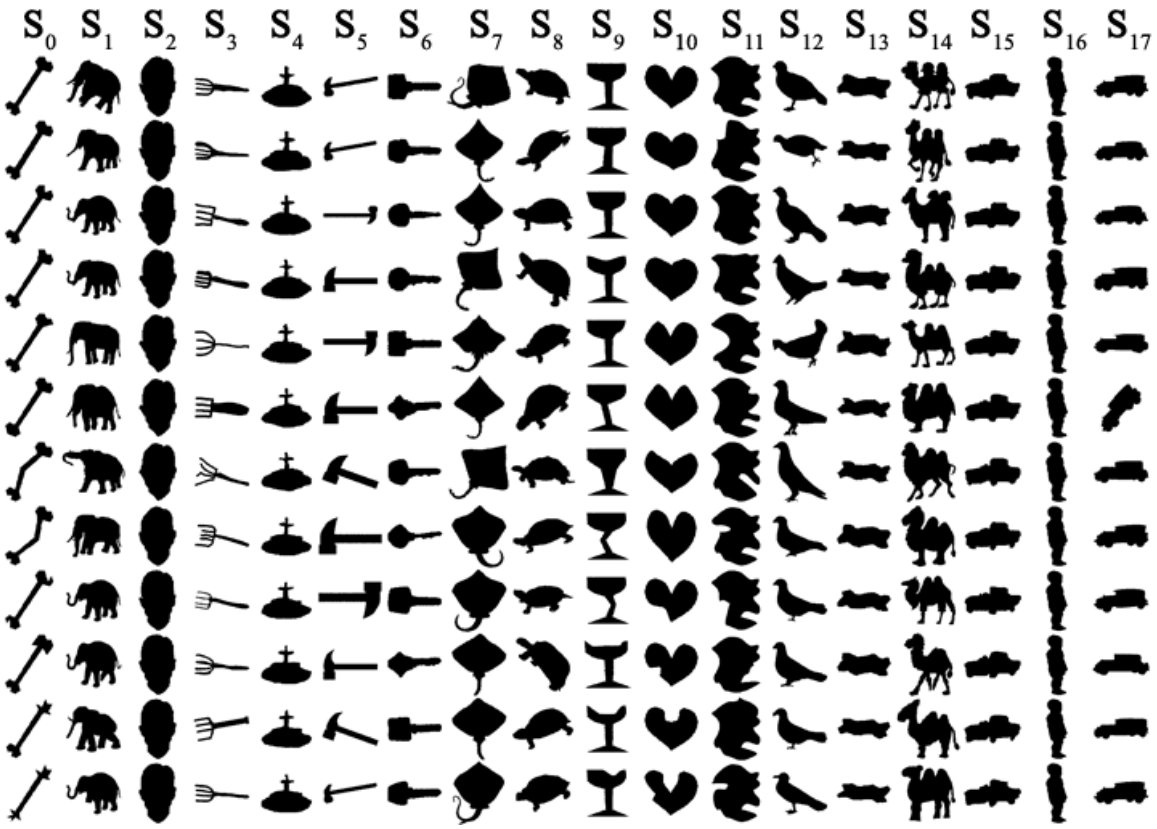
$$A_G = \frac{\sum_{g=1}^{G} s_{gg}}{N} \tag{10}$$

şeklinde hesaplamak mümkündür[15]. Eş. 10 da s_{gg} doğru olarak sınıflandırılan örnek sayısını temsil etmektedir. Çok sınıflı karışıklık matrisinde her bir sınıfın kesinlik ve hassasiyeti her zaman algoritmanın yeteneği hakkında doğru sonuç vermeyebilir. Bu yüzden bütün sınıflardaki başarımın ortalaması alınarak sırasıyla,

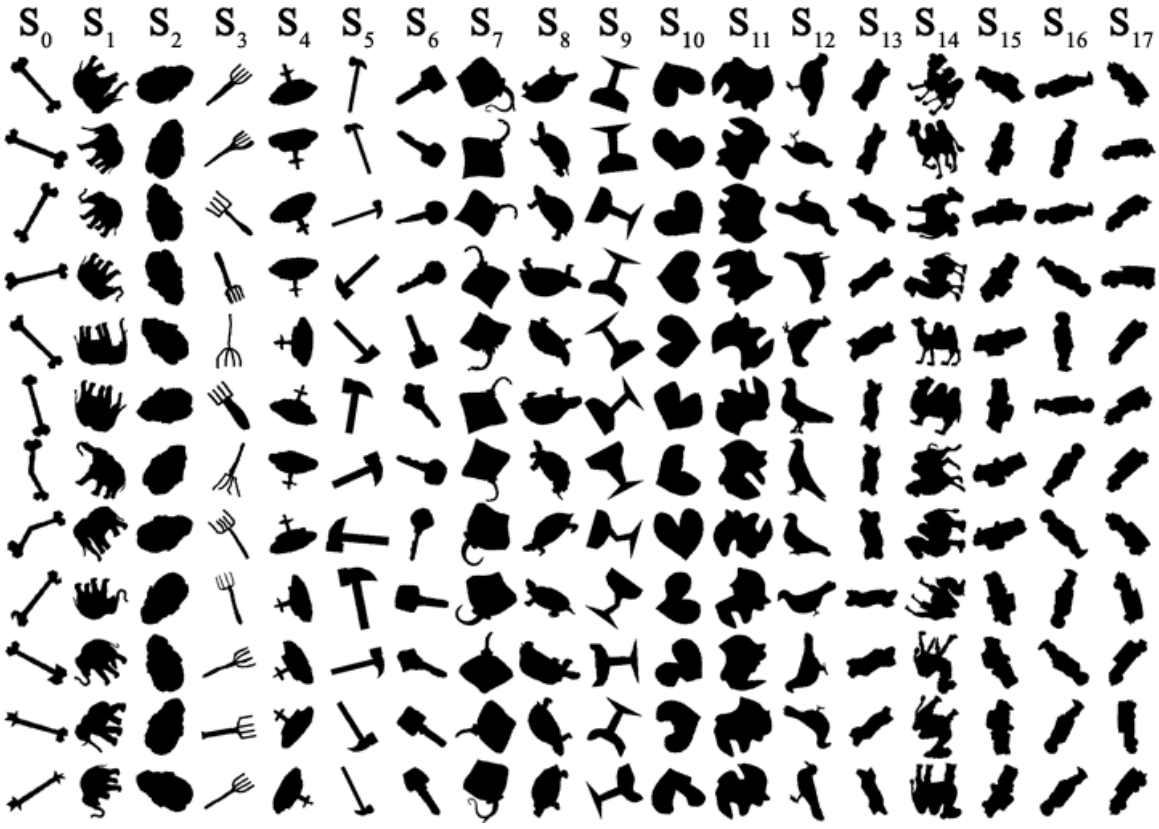
$$P_M = \frac{\sum_{g=1}^{G} P_g}{G} \tag{11}$$

$$R_M = \frac{\sum_{g=1}^{G} R_g}{G} \tag{12}$$

makro kesinlik ve makro hassasiyet hesaplanması tercih edilen yaklaşımdır [15-16]. Bu çalışmada Freeman 8 zincir kod histogramı ve 8 kabuklu KZKH histogramı test edilmiştir. Ayrıca orijinal ve döndürülen Kimia 216 şekil veri tabanı kullanıldığından 4 farklı karışıklık matrisi elde edilmiştir. Karışıklık matrisinin içeriği elde edilir iken, sorgulanan nesne hariç en yakın nesne sınıfı sayılmıştır. Orijinal Kimia 216 veri tabanı, FR8 zincir kod histogramı ile test edildiğinde alınan sonuçlar Tablo 1 de, 8 kabuklu KZKH histogramı ile test edildiğinde ise Tablo 2 gösterilen sonuçlar alınmıştır. Aynı deneyler döndürülmüş Kimia 216 veri tabanı ve FR8 zincir kod histogramı ile yapıldığında Tablo 3 deki sonuçlar, 8 kabuklu KZKH histogramı ile yapıldığında ise Tablo 4 gösterilen sonuçlar elde edilmiştir.



Şekil 11. *Kimia 216 veritabanı*



Şekil 12. *Döndürülmüş Kimia 216 veritabanı*

		Tahmin Edilen Sınıflar							
		1	2	3	…	…	G	n_g	R_g
Gerçek Sınıflar	1	s_{11}	s_{12}	s_{13}	…	…	s_{1g}	n_1	R_1
	2	s_{21}	s_{22}	s_{23}	…	…	s_{2g}	n_2	R_2
	3	s_{31}	s_{32}	s_{33}	…	…	s_{3g}	n_3	R_3
	…	…	…	…	…	…	…	…	…
	…	…	…	…	…	…	…	…	…
	G	s_{G1}	s_{G2}	s_{G4}	…	…	s_{GG}	n_G	R_G
	n'_g	n'_1	n'_2	n'_3	…	…	n'_G		
	P_g	P_1	P_2	P_3	…	…	P_G		

Şekil 13. *Genelleştirilmiş karışıklık matrisi*

Tablo 1. *Orijinal Kimia 216 ve FR8 zincir kod histogramı*

		Tahmin Edilen Sınıf																			
		S_0	**S_1**	**S_2**	**S_3**	**S_4**	**S_5**	**S_6**	**S_7**	**S_8**	**S_9**	**S_{10}**	**S_{11}**	**S_{12}**	**S_{13}**	**S_{14}**	**S_{15}**	**S_{16}**	**S_{17}**	**n_g**	**R_g**
Gerçekteki Sınıf	**S_0**	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_2	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	12	0,750
	S_3	0	0	0	8	0	0	0	1	0	1	0	0	0	1	0	0	0	1	12	0,667
	S_4	0	0	0	2	7	0	0	0	0	0	0	0	0	0	0	0	2	1	12	0,583
	S_5	0	0	0	0	0	6	0	1	0	0	0	0	1	1	0	3	0	0	12	0,500
	S_6	0	0	0	0	0	0	7	0	0	0	5	0	0	0	0	0	0	0	12	0,583
	S_7	0	0	0	1	0	0	0	8	0	1	0	0	0	2	0	0	0	0	12	0,667
	S_8	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	12	1,000
	S_9	1	0	0	0	0	1	0	1	0	7	0	0	0	1	0	1	0	0	12	0,583
	S_{10}	1	0	0	0	0	0	3	0	0	0	8	0	0	0	0	0	0	0	12	0,667
	S_{11}	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	12	1,000
	S_{12}	0	0	0	0	0	1	0	0	0	0	0	0	9	0	2	0	0	0	12	0,750
	S_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	2	12	0,833
	S_{14}	0	0	0	0	0	1	0	1	0	2	0	0	1	0	5	2	0	0	12	0,417
	S_{15}	0	1	1	0	0	4	0	0	0	1	0	0	0	1	1	2	0	1	12	0,167
	S_{16}	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	8	0	12	0,667
	S_{17}	0	0	0	2	1	0	0	0	0	1	0	0	0	2	0	0	0	6	12	0,500
	n'_g	14	13	13	13	8	13	10	12	12	13	13	12	11	19	8	9	12	11		
	P_g	0,857	0,923	0,692	0,615	0,875	0,462	0,700	0,667	1,000	0,538	0,615	1,000	0,818	0,526	0,625	0,222	0,667	0,545		

Tablo 2. *Orijinal Kimia 216 ve KZKH (8 kabuk)*

		Tahmin Edilen Sınıf																			
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	n_g	R_g
Gerçekteki Sınıf	S_0	11	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	12	0,917
	S_1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_2	0	0	10	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	12	0,833
	S_3	0	0	2	9	0	0	1	0	0	0	0	0	0	0	0	0	0	0	12	0,750
	S_4	0	0	0	0	8	0	0	1	0	0	0	0	1	1	0	0	0	1	12	0,667
	S_5	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_6	0	0	0	3	1	0	4	0	0	0	2	0	0	0	0	0	0	2	12	0,333
	S_7	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_8	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	12	1,000
	S_9	0	0	0	0	0	1	0	0	0	11	0	0	0	0	0	0	0	0	12	0,917
	S_{10}	0	0	0	0	0	0	2	0	0	0	10	0	0	0	0	0	0	0	12	0,833
	S_{11}	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	12	1,000
	S_{12}	0	2	0	0	1	0	0	0	0	0	0	0	9	0	0	0	0	0	12	0,750
	S_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	1	12	0,917
	S_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	12	1,000
	S_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	11	0	0	12	0,917
	S_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	12	1,000
	S_{17}	0	0	0	0	2	0	2	0	0	0	0	0	0	2	0	0	0	6	12	0,500
	n'_g	11	14	12	13	12	14	9	13	12	11	13	12	10	14	13	11	12	10		
	P_g	1,000	0,857	0,833	0,692	0,667	0,857	0,444	0,923	1,000	1,000	0,769	1,000	0,900	0,786	0,923	1,000	1,000	0,600		

Tablo 3. *Döndürülmüş Kimia 216 ve FR8 zincir kod histogramı*

		Tahmin Edilen Sınıf																			
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	n_g	R_g
Gerçekteki Sınıf	S_0	4	0	0	1	2	1	0	0	1	0	0	1	0	1	0	1	0	0	12	0,333
	S_1	0	2	0	2	1	0	1	1	0	2	0	0	0	2	0	1	0	0	12	0,167
	S_2	0	1	2	1	1	0	1	0	0	0	0	0	0	0	0	0	6	0	12	0,167
	S_3	0	2	0	4	0	0	1	1	0	0	1	1	0	1	0	0	0	1	12	0,333
	S_4	1	1	1	0	1	1	2	2	0	0	0	0	0	0	0	3	0	0	12	0,083
	S_5	1	0	0	0	0	1	1	1	1	1	2	0	0	1	1	0	0	2	12	0,083
	S_6	0	2	0	0	0	0	0	2	0	0	5	1	0	0	0	1	0	1	12	0,000
	S_7	0	0	0	1	0	0	1	0	0	1	1	0	0	5	0	1	0	2	12	0,000
	S_8	2	0	0	0	0	4	0	0	1	1	1	0	0	0	0	0	0	3	12	0,083
	S_9	0	3	0	0	1	0	0	0	2	5	0	0	0	1	0	0	0	0	12	0,417
	S_{10}	0	1	0	0	0	1	4	0	1	0	1	0	0	2	0	0	1	1	12	0,083
	S_{11}	2	0	0	2	0	0	3	1	0	0	0	1	0	2	0	0	0	1	12	0,083
	S_{12}	1	0	0	0	0	1	0	1	1	0	0	0	5	0	3	0	0	0	12	0,417
	S_{13}	0	1	0	0	0	1	1	4	0	0	1	0	0	0	0	1	0	3	12	0,000
	S_{14}	0	0	0	0	0	2	0	2	2	0	1	0	1	0	0	3	0	1	12	0,000
	S_{15}	1	2	0	0	2	0	2	1	0	0	1	1	0	0	1	1	0	0	12	0,083
	S_{16}	0	0	4	1	1	0	0	0	0	0	0	0	0	0	0	0	6	0	12	0,500
	S_{17}	0	1	0	1	1	1	1	0	2	0	0	2	0	2	0	1	0	0	12	0,000
	n'_g	12	16	7	13	10	13	18	16	11	10	14	7	6	17	5	13	13	15		
	P_g	0,333	0,125	0,286	0,308	0,100	0,077	0,000	0,000	0,091	0,500	0,071	0,143	0,833	0,000	0,000	0,077	0,462	0,000		

Tablo 4. *Döndürülmüş Kimia 216 ve KZKH (8 kabuk)*

		Tahmin Edilen Sınıf																			
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	n_g	R_g
Gerçekteki Sınıf	S_0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	12	0,833
	S_1	0	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	12	0,917
	S_2	0	0	7	3	0	0	1	0	0	0	1	0	0	0	0	0	0	0	12	0,583
	S_3	0	0	3	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0,750
	S_4	1	0	0	0	9	0	0	1	0	0	0	0	0	1	0	0	0	0	12	0,750
	S_5	0	0	0	0	0	11	0	1	0	0	0	0	0	0	0	0	0	0	12	0,917
	S_6	0	0	1	1	1	0	7	0	0	0	0	0	0	1	0	0	0	1	12	0,583
	S_7	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	12	1,000
	S_8	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	12	1,000
	S_9	0	0	0	0	0	1	0	0	0	11	0	0	0	0	0	0	0	0	12	0,917
	S_{10}	0	0	0	0	0	0	2	0	0	0	8	0	0	0	0	0	0	2	12	0,667
	S_{11}	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	12	1,000
	S_{12}	0	3	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	12	0,750
	S_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	12	1,000
	S_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	12	1,000
	S_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	12	1,000
	S_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	12	1,000
	S_{17}	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	9	12	0,750
	n'_g	11	14	11	13	10	12	11	14	12	11	10	12	10	15	13	12	12	13		
	P_g	0,909	0,786	0,636	0,692	0,900	0,917	0,636	0,857	1,000	1,000	0,800	1,000	0,900	0,800	0,923	1,000	1,000	0,692		

Tablo 1 ve Tablo 2 de görüleceği gibi önerilen KZKH algoritmasının genel olarak hassasiyet (R_g) ve kesinlik (P_g) metrikleri FR8 zincir kod histogramının çıktılarından yüksektir. Başka bir deyişle KZKH algoritması ayırt edicilik yeteneği açısından FR8 den daha başarı olduğu söylenebilir. FR8 zincir kod histogramı ile en kötü ayırt edicilik S_{15} nesne sınıfında gözlenmiş olup hassasiyeti 0,167 olarak bulunmuştur. Diğer taraftan aynı nesne sınıfının KZKH ile hassasiyeti ise 0,917 olarak hesaplanmıştır. KZKH da ise en kötü ayırt edicilik sonucu S_6 sınıfında gerçekleşmiştir. FR8 zincir kod histogramı ile döndürme etkilerinin diğer sınıflara göre daha az olduğu S_0, S_8 ve S_{11} nesne sınıflarında ayırt edicilik sorunuyla karşılaşılmamıştır. KZKH yöntemi ile S_7 nesne sınıfında tüm nesneler doğru bir şekilde tanınmıştır.

Tablo 3 ve Tablo 4 de verilen döndürülmüş veri tabanı sonuçlarına göre, sınıf bazında hassasiyet ve kesinlik metrikleri açısından KZKH algoritmasının çıktılarında pek az değişiklik olmuştur. En önemli değişim ağırlık merkezi etkilerinin gözlemlendiği S_{14} nesne sınıfında yaşanmıştır. Buda önerilen yöntemin döndürülmeye karşı dayanıklı olduğunun göstergesidir. Diğer taraftan döndürülmüş veri tabanı ile yapılan deneylerde FR8 zincir kodunun performansını oldukça düştüğü açıktır. Bunun nedeni sınır noktalarda meydana gelen yön değişiklikleridir. Örneğin S_6, S_7, , S_{13}, S_{14}, S_{17} nesne sınıfında hiçbir şekil doğru tahmin yapılamamıştır. Diğer nesne sınıflarında ise orijinal Kimia 216 şekil veri tabanındaki sonuçlara göre hem hassasiyet hem de kesinlik ölçütlerinde düşüş meydana gelmiştir.

Yapılan 4 farklı deneyin karışıklık matrisleri kullanılarak hesaplanan makro çıktıları Tablo 5 gösterilmiştir. Tablo 5 de görüleceği gibi orijinal Kimia 216 veri tabanı ile yapılan deneylerde alınan doğruluk (A_G) sonucuna göre KZKH algoritması %84,73 ve FR 8 zincir kod histogramı %68,60

başarılı olmuştur. Döndürme etkileri az olmasına rağmen sınır noktalarda meydana gelen küçük değişikler FR 8 zincir kod histogramını önerilen metoda göre daha çok etkilemiştir. Döndürülmüş Kmia 216 veri tabanı ile yapılan deneylerde ise KZKH metodunda makro kesinlik ve hassasiyette önemli bir değişiklik olmaz iken, FR8 zincir kod histogramında makro hassasiyette ve makro kesinlikte çok ciddi düşüş meydana gelmiştir. Makro ölçeklerde oluşan düşüşler FR8 zincir kod histogramının döndürmeye karşı duyarlılığını göstermektedir. Söz konusu veri tabanı kullanıldığında elde edilen doğruluk (A_G) sonucuna göre KZKH algoritması %85,19 ve FR 8 zincir kod histogramı %18,92 başarılı olmuştur. Ayrıca her iki veri tabanıyla test edilen yöntemlerde doğruluk metriği nesne sınıfları eşit miktarda örneklem içerdiği için makro hassasiyet metriğiyle aynı çıkmıştır.

Tablo 5. *Kmia 216 ve global değerlendirme*

Şekil Veri Tabanı	Metot	P_M	R_M	A_G
Orijinal Kimia 216	FR8 Histogramı	0,6860	0,6852	0,6852
Döndürülmüş Kimia 216	FR8 Histogramı	0,1892	0,1574	0,1574
Orijinal Kimia 216	KZKH (8 kabuk)	0,8473	0,8519	0,8519
Döndürülmüş Kimia 216	KZKH (8 kabuk)	0,8583	0,8565	0,8565

VI. SONUÇ

Bu çalışmada hem bölge hem de sınır bilgisinin kullanıldığı yeni bir zincir kod histogramı önerilmiştir. Geliştirilen metot ve FR8 yaklaşımı döndürmeye karşı test edilmiştir. Döndürme etkilerinin fazla olduğu nesnelerde, önerilen metodun makro doğruluk ve makro kesinlik gibi başarı ölçütlerinde önemli bir değişiklik olmaz iken, FR8 zincir kod histogramı yönteminde makro performanslar oldukça düşük kalmıştır. Böylece KZKH tekniğinin döndürmeye karşı dayanıklı olduğu gözlemlenmiştir. Günümüze kadar önerilen zincir kodlarında sembol sayısı sabittir. Diğer taraftan geliştirilen kabuk zincir kodundaki sembol sayısı kabuk sayısı ile orantılıdır. Ağırlık merkezi değişimlerinden etkilenmesi önerilen yöntemin en belirgin dezavantajıdır.

V. KAYNAKLAR

[1] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers,* vol. EC-10, no. 2, pp. 260-268, 1961.

[2] H. Freeman, "Computer processing of line drawing images," *ACM Computing Surveys (CSUR),* vol. 6, no. 1, pp. 57-97, 1974.

[3] S. Papert, "Uses of technology to enhance education," AI lab MIT, U.S.A, Technical Report 298, 1973.

[4] S. H. Cruz and R. M. R.Dagnino, "Compressing bi-level images by means of a 3-bit chain code," *SPIE Opt. Eng.*, vol. 44, no. 9, pp. 1-8, 2005.

[5] P. Nunes, F. Pereira, F. Marqués, "Multi-grid chain coding of binary shapes," ICIP'97 Proceedings of the 1997 International Conference on Image Processing, DC USA, vol.3, pp. 114-117, 1997.

[6] E. Bribiesca, "A new chain code," *Pattern Recognition*, vol. 32, no. 2, pp. 235-251, 1999.

[7] Y.K. Lui ve B. Zalik, "An efficient chain code with Huffman coding," *Pattern Recognition*, vol. 38, no. 4, pp. 553-557, 2005.

[8] B. Zalik, D. Mongus, Y. Liu, N. Lukač, "Unsigned Manhattan chain code," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 186-194, 2016.

[9] D. Zhang ve G. Lu, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, no. 1, pp. 1-19, 2004.

[10] J. Iivarinen ve A. Visa, "Shape recognition of irregular objects," *Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling*, vol. 2904, pp. 25-32, 1996.

[11] S. K. Pradhan, S. Sarker, S. K. Das, "A Character Recognition Approach using Freeman Chain Code and Approximate String Matching," *International Journal of Computer Applications*, vol. 84, no. 11, pp. 38-42, 2013.

[12] M. Ankerst, G. Kastenmüller, H.P. Kriegel, T. Seidl, "Nearest neighbor classification in 3D protein databases," Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology, vol. 99, pp. 34-43, 1999.

[13] T.B. Sebastian, P.N. Klein, B.B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 550–571, 2004.

[14] H.Tuna ve R. Demirci, "Shell Chain Code," ISMSIT 2018 Proceedings of the 2018 Symposium on Multidisciplinary Studies and Innovative Technologies, Kızılcahamam Turkey, 2018.

[15] D. Ballabio, F. Grisoni, R. Todeschini, "Multivariate comparison of classification performance measures," *Chemometrics and Intelligent Laboratory Systems*, vol. 174, pp. 33-44, 2018.

[16] M. Sokolova ve G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427-437, 2009.