C.B.Ü. Fen Bilimleri Dergisi ISSN 1305-1385 C.B.U. Journal of Science
9.2 (2013) 45 – 53 9.2 (2013) 45 –53

GENETiK - LBG ALGORİTMASI İLE SAYISAL GÖRÜNTÜLERİN SIKIŞTIRILMASI

İlker KILIÇ[1*]

[1]Celal Bayar Üniversitesi, Elektrik-Elektronik Mühendisliği Fakültesi, 45140 Manisa, TÜRKİYE

Özet: Bu çalışmada K-Ortalamalar(KO), LBG ve Bulanık C Ortalamalar(BCO) güncel kümeleme algoritmaları yardımı ile bulunan merkezler üzerinden gerçekleştirilen kayıplı görüntü sıkıştırma algoritmalarının performansları, önerilen Genetik LBG Algoritması (GA-LBG) ile iyileştirilmiştir. Önerilen yeni algoritma standart görüntüler üzerinde denenmiş, klasik yöntemlerden hem OKH(Ortalama karesel hata) değerleri, hem de sıkıştırılıp açılan görüntü kalitesi açısından üstün olduğu gözlenmiştir.

Anahtar Kelimeler: *K-Ortalamalar, LBG, Bulanık K-Ortalamalar, Genetik, Kümeleme*

DIGITAL IMAGE COMPRESSION BY GENETIC – LBG ALGORITHM

Abstract: In this study, using cluster centers of the popular clustering algorithms such as K-Means, LBG and Fuzzy C-Means a lossy compression is performed. The performances of these algorithms are improved by the proposed Genetic LBG algorithm. The new algorithm is applied on the standard images and seen that it is better than the classical methods according to both MSE values and visual assessments.

Keywords: *K-Means, LBG, Fuzzy C-Means, Genetic, Clustering*

[*]**İlker KILIÇ**
ilker.kilic@cbu.edu.tr

1. GİRİŞ

Sayısal video veya resim çekip onları saklamak günümüz teknolojisinde en çok başvurduğumuz araçlardandır. Çekilen resmin kalitesi hafızada kapladığı alan ile doğru orantılıdır. Kalite attırıldığında hafızada kapladığı alan da artar. Bu durum çok fazla resim depolamak istediğimizde sorun oluşturmaktadır. Literatürde, Dalgacık, Fourier ve Kosinüs Dönüşüm tabanlı dönüşümler ile görüntüyü frekans uzayına taşıyarak veya doğrudan görüntü üzerinde entropi tanımını kullanarak ya da görüntüyü alt kümelere ayırarak birçok kayıplı ya da kayıpsız sıkıştırma algoritması geliştirilmiştir. Bunlardan Hufman veya Aritmetik Kodlama gibi kayıpsız olanlar görüntü kalitesinden ödün vermezler fakat çok ciddi oranda sıkıştırma gerçekleştiremezler. Bunların yerine görüntüde bir miktar kayıp ile yüksek oranda sıkıştırma daha çok tercih edilmektedir. Bu tekniklerden en popüler olanları ise K-Ortalamalar (KO), Bulanık K-Ortalamalar, LBG ile Vektör Nicemleme ve Genetik Algoritma (GA) dır. Bu çalışmada popüler olan iki kümeleme algoritması LBG ile GA birleştirilmiş aynı sıkıştırma oranlarında diğer algoritmalardan daha kaliteli sonuçlar elde edilmiştir.

2. METOTLAR

2.1 K- Ortalamalar Algoritması

K- Ortalamalar algoritması (KO), veri setini istenen sayıda alt kümeye ayırabilen bir yöntemdir. Kümeleme işlemini, orijinal set içerisindeki verileri *K* adet küme merkezinden Euclidean uzaklığı en yakın olanlarına atayarak gerçekleştirir. Bu algoritmanın temel amacı k adet küme ve bu kümelere ait k adet merkez belirlemektir [1]. Bunu gerçekleştirmek için ilk önce veri seti içerisinden rastgele seçilmiş k adet veriyi merkez olarak belirler. Daha sonra her bir verinin *k* adet merkeze olan Euclidean uzaklığını hesaplayıp en yakın olan merkeze ait kümeye atar. Bir başka deyişle her bir merkez kendisine Euclidean uzaklığı ile en yakın verileri içinde toplayan bir küme oluşturur. Daha sonra küme merkezleri, kümeye ait verilerin ortalaması alınarak güncellenir. Güncellenen yeni küme merkezleri için tüm veri setinden yeni küme elemanları belirlenir. Bu işlem herhangi bir küme merkezinde ve o kümeye ait verilerde herhangi bir değişiklik meydana gelmeyinceye kadar devam eder. KO, (1) nolu denklem ile modellenebilir:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2 \quad (1)$$

Burada *n* veri sayısını, *k* merkez sayısını, $\left\| x_i^j - c_j \right\|^2$ formülü ise x_i^j verisinin c_j küme merkezine, yani *i*. verinin *j*. küme merkezine olan Euclidean uzaklığını belirtir.

Standart KO, [2-4] referansları ile belirtilen çalışmalarda geliştirilmiştir. [2] nolu çalışmada yazarlar tüm veri seti için bir kd-ağaç yapısı oluşturmuşlardır. Bu yapı algoritma çalışmaya başladığında bir kere hesaplanıp daha sonra değişmediğinden algoritmanın toplam çalışma süresini kısaltmış ve algoritma içerisindeki toplam matematiksel hesaplama adedini azaltmıştır. [3] nolu çalışmada yazarlar yeni bir global KO geliştirmişlerdir. Bu algoritmada merkez sayıları birer birer artırılmakta ve her yeni bulunacak merkez için tüm veri seti üzerinde global istatiksel hesaplamalar yapılmakta ve yeni merkezin yeri belirlenmektedir. [4] nolu çalışmada ise yazarlar KO'nın local minimum merkezlere takılmaktan kurtaracak yeni bir yöntem geliştirmişlerdir. Buna göre algoritmanın başında veri seti içerisinden rasgele belirlenen merkezler yerine, yardımcı fonksiyonlar kullanarak veri setinin değişik yapılara sahip bölgelerini belirleyip o bölgelere merkez atayarak algoritmayı daha istikrarlı bir yapıya kavuşturmuşlardır.

2.2 Bulanık C-Ortalamalar Algoritması

Literatürde, Bulanık C-Ortalamalar algoritması (BCO), ilk kez Bezdek tarafından önerilen bir kümeleme algoritmasıdır [5]. BCO'da her bir veri birden fazla kümeye ait olabilir. Üyelik katsayısı, verilerin her bir kümeye ne oranda ait olduğunu gösterip veri ile küme merkezleri arasındaki uzaklık kullanılarak hesaplanır. Orijinal data küme merkezine yaklaştıkça üyelik katsayısı da o oranda artış gösterir. Orijinal herbir verinin küme sayısı kadar, 0 ile 1 arasında bir değerde üyelik katsayısı olup toplamları bire eşittir. Küme merkezleri, o kümeye ait tüm verilerin ortalaması alınarak bulunur[6]. BCO iteratif bir yöntem olup her bir iterasyonda tüm verilere ait üyelik katsayıları ve küme merkezleri güncellenir. BCO'da amaç, aşağıda (2) nolu ifade ile verilen hedef fonksiyonu minimize etmektir.

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left\|x_i - c_j\right\|, \quad 1 \le m < \infty \qquad (2)$$

Burada m; birden büyük gerçek bir sayı, u_{ij}; *j*. sınıfa ait x_i elemanının üyelik katsayısı, x_i; *i*. veri, c_j; *j*. kümenin merkezi, ||*|| ise herhangi bir veri ile kümenin merkezi arasındaki uzaklığı ifade eder. Bulanık kümeleme algoritması iteratif olup küme merkezleri ile üyelik fonksiyonların ardı ardına güncellenmesi ile (2) nolu hedef fonksiyonunun minimize edilmesi esasına dayanır. Üyelik değerleri (u_{ij}) ve küme merkezleri (c_j), (3) nolu fonksiyonlar ile bulunmaktadır.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|x_i - c_j\right\|}{\left\|x_i - c_k\right\|}\right)^{\frac{2}{m-1}}}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} . x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \qquad (3)$$

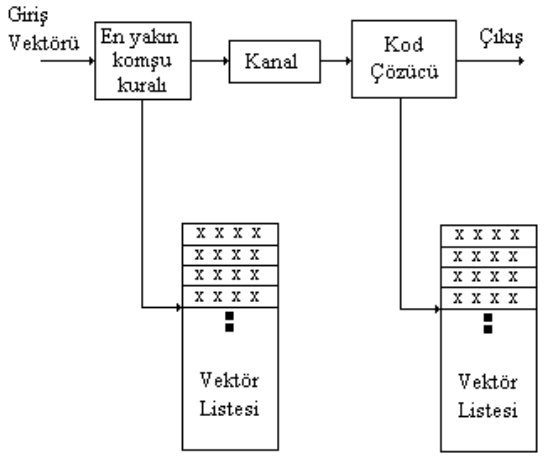
BCO algoritması, 0<ε<1 ve *k* iterasyon sayısı olmak üzere, $\max_{ij}\{|u_{ij}^{k+1} - u_{ij}^{k}|\} < \varepsilon$ kriteri sağlandığında durmaktadır.

Standart BCO algoritması [7,8] nolu çalışmalarda geliştirilmiştir. [7] nolu çalışmada komşu verilerin farkları Gauss fonksiyonu ile temsil edilmiş, bu sayede küme merkezlerinin üyelikleri ayarlanmıştır. [8] nolu çalışmada ise klasik BCO algoritmasında, uzaysal verilerin dikkate alınmamasından kaynaklanan gürültü hassasiyeti problemi çözülmüştür. Bunun için BCO algoritmasına ait hedef fonksiyonu uzaysal bilgiyi içerecek şekilde geliştirilmiştir.

2.3 Vectör Nicemleme ve LBG Algoritması

Vektör Nicemleme(VN) algoritmasında [9], sayısal görüntü alt bloklara, diğer adı ile vektörlere ayrılır ve LBG adı ile bilinen algoritma [10] yardımı ile resmi en iyi temsil eden vektör listesi oluşturulur. Kodlayıcı, resimden gelen bir bloğu alır ve Euclide Uzaklığı hata kriterine bağlı kalarak bloğa en yakın vektörün adresini, LBG algoritması eğitimi sonunda oluşturduğu vektör listesinden bularak alıcıya gönderir. Alıcı, bu adrese karşılık gelen vektörü aynı vektör listesinden bularak bilgisayar ekranına yansıtır. Bu algoritma çalıştırıldığında bir sıkıştırma işlemi kendiliğinden oluşur (Şekil 1). Çünkü vektörün kendisini göndermek yerine çok daha az bit kullanılarak adresi gönderilmiştir.

VN algoritması ile yapılan bir sıkıştırma işleminin performansı, LBG eğitimi sonunda oluşturulan vektör listesine ve bu listenin büyüklüğüne bağlıdır. Bu yüzden LBG algoritması VN için önemlidir. İlk olarak Linde, Buzo ve Gray tarafından bulunan LBG algoritması sadece lokal optimizasyon sağlamaktadır. LBG algoritması ile oluşturulan vektör listesinin boyutları VN'nin



Şekil 1. Vektör Nicemleme ile görüntü sıkıştırma

genel performansı ile doğru, sıkıştırma oranı ile ters orantılıdır. Vektör listesinin boyutu artırıldıkça görüntü kalitesi artmakta fakat sıkıştırma miktarı azalmaktadır. Bu yüzden genellikle vektör liste boyutunda bir optimizasyona gidilir. Bu çalışmada klasik algoritmalar ile önerilen Genetik-LBG algoritması için 256x256 pixel boyutlu resimler ve LBG için 4x4 pixel boyutlu vektörler kullanılmıştır.

Bir VN işleminde *k* boyutlu bir öklit uzayı sonlu bir Y serisine dönüştürülür.

$$Q:R^k \rightarrow Y \qquad (4)$$

Burada $Y=\left(\hat{x}_{ij}:i=1,2,3,...,N\right)$, görüntüyü yeniden oluşturmada kullanılacak vektör serisi, *N* ise bu seri içerisindeki vektör sayısıdır. Yapılan bu işlem iki aşamadan oluşur. İlk aşamada LBG algoritması ile oluşturulan vektör listesi içerisinden uygun *x* vektörü bulunup, bu vektörün adresi *Q(x)* ile ifade edilir. İkinci aşamada kod çözücü kendisine gelen adres bilgisi ile vektör listesinden $\hat{x}$ vektörüne ulaşıp bu vektörü tekrar oluşturur. Özgün vektör ile bu vektörün sıkıştırılıp tekrar açılmasından elde edilecek vektör arasındaki karesel hata *d(x,* $\hat{x}$*)* değeri ne kadar küçük ise sıkıştırılıp tekrar açılmış görüntü o kadar kaliteli olur. LBG ve benzer algoritmalar belli bir veri dizisi kullanarak bu hata kriterinin minimizasyonu için çalışırlar. Karesel hata şu şekilde ifade edilir;

$$d(x,\hat{x})=\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{ij}-\hat{x}_{ij}\right)^2 \qquad (5)$$

Burada *M* resimdeki vektör sayısı, *N* vektör içerisindeki piksel sayısı, x_{ij} orijinal pixel değeri, $\hat{x}_{ij}$ yeniden oluşturulan vektördür. Bu çalışmada 4x4 piksel boyutlu vektörler seçildiği için *N*=16 alınmıştır. Bir vektör nicemleyicinin amacı, *N* adet tekrar üretilebilen vektör dizisi oluşturup bu sayede hatayı azaltmaktır. Belli bir eğitme seti için LBG algoritması aşağıdaki adımlar yardımı ile gerçekleştirilebilir;

1. İlk olarak eğitme seti x_j için *j=0,1,2,...,n-1* tanımlayıp, *N* boyutlu anlık vektör listesi bulunur $\left(\hat{Y}_0\right)$. İterasyon sayısı sıfırlanır. Burada *N*, elde edilmek istenen vektör listesinin büyüklüğü, ε ise hata eşik değeridir (sıfırdan büyük seçilir).

2. Eğitme seti içerisindeki herbir vektörün (256x256 boyutlu imgede 4x4 piksel boyutunda), anlık vektör listesindeki $\hat{Y}_M\left(y_i;i=1,2,3,...,N\right)$ tüm vektörlere olan uzaklığı hesaplanır. Eğer x_j vektörü y_i vektörüne en yakın vektör ise $d\left(x_j,y_i\right)\le d\left(x_j,y_l\right)_{[l\neq i,l=1,2,...N]}$, uzaklık D_k serisi altında, yakın olan vektörlerin sayısı S_k serisi altında toplanır.

$$S_k=\left(S_i;i=1,2,...,N\right)$$
$$D_k=\left[\sum_{j=0}^{n-1}\min_{(y\in Y_m)}x_j\right]/S_k \qquad (6)$$

3. Eğer $\left(D_{k_{m-1}}-D_{k_m}\right)/D_{k_m}\le\varepsilon$ koşulu tüm *k=1,2,...,N* değerleri için sağlanıyorsa iterasyonu durdurup $\hat{Y}_m$ listesi asıl vektör listesi olarak kabul edilir.

4.Yeni optimum vektör listesi bulunur.

$$\hat{x}_j = D_k\left(j, k = 1,2,..., N\right)$$
$$\hat{Y}_{m+1} = \hat{x}_j\left(j = 1,2,..., N\right) \quad (7)$$

İterasyon sayısını bir artırıp ($m \rightarrow m+1$) ikinci adıma gidilir.

Yukarıda verilen LBG algoritmasında $\hat{Y}_0$, iterasyonu başlatan anlık vektör listesi olarak kabul edilmiştir. Anlık vektör listesini oluşturmada kullanılacak birkaç teknik vardır. Bunlardan en basit ve kullanışlı olanında, ilk önce tüm vektörlerin ortalaması alınır, bu eğitmede kullanılan vektörlerin merkezidir. Daha sonra bu merkez vektöre en yakın iki vektör tespit edilir. Bu vektörlere en yakın vektörlerin ortalaması alınır. Ortalama alma işlemi bu iki vektör elemanları değişmeyinceye kadar devam eder. Sonra bu iki vektöre en yakın dört vektör bulunarak, ortalama alma işlemine tekrar başlanır. Bu işlemler vektör sayısı belli bir büyüklüğe erişinceye kadar devam eder. Sonuçta anlık $\hat{Y}_m$ vektör listesi bulunmuş olur.

Literatürde LBG algoritması [11-14] çalışmaları ile geliştirilmiştir. [11] ve [13] nolu çalışmalarda LBG algoritmasının hızı artırılmış, [12] nolu çalışmada LBG başlangıç koşullarına bağımlılığı azaltılıp performansı iyileştirilmiştir. [14] nolu çalışmada LBG vektör listesi Hadamard dönüşüm ortamında gerçekleştirilerek performansı artırılmıştır.

2.4 Genetik - LBG Algoritması

Genetik algoritma (GA), doğal seçim ilkelerine dayanan bir arama ve optimizasyon yöntemidir. Temel adımları ilk kez John Holland [15] tarafından belirlenmiştir. Daha sonra değişik bilim dallarına ait optimizasyon içeren problemlerinde global sonuca ulaşabilen yöntem olarak kullanılmıştır. Literatürde sayısal verilerin kümelenmesinde genetik algoritma kullanılmıştır. [16] nolu çalışmada GA ile LBG algoritmaları gizli bilgileri şifreleme amaçlı birleştirilmiştir. [17] nolu çalışmada sayısal sesler için genetik algoritma kullanarak bir LBG vektör listesi oluşturup kimlik tespiti yapılmıştır. [18] nolu çalışmada vektör listesi oluşturmak için kullanılan genetik algoritmanın hızı artırılmıştır. Bu makalede ise bunlardan farklı olarak GA ile LBG algoritmaları sayısal görüntülerin sıkıştırılması için birleştirilip yeni bir algoritma elde edilmiştir. Literatürdeki klasik algoritmalar ile değişik sıkıştırma oranlarında karşılaştırılmış, aynı sıkıştırma oranlarında OKH kriterine göre daha üstün olduğu gözlenmiştir. Genetik algoritmanın operatörleri üretilen popülasyon üzerine uygulanan işlemlerdir. Bu işlemlerin amacı daha iyi bir çözüm üretmek ve çözüm arama alanını genişletmektir. Farklı uygulamalarda farklı operatörler kullanılmakla birlikte genetik algoritmada 3 standart operatör kullanılır; Yeni popülasyon üretimi (Generation Reproduction), Çaprazlama (Crossover), Mutasyon(Mutation). Bu işlemler ardaşık olarak uygulandıktan sonra, üretilen yeni popülasyonlardan her birine ait uygunluk fonksiyonları (Objective Function) hesaplanır. Problemin en iyi çözümü(Elite) belirlenip saklanarak tüm adımlar yeniden calıştırılır. Genetik algoritma iteratif olup önceden belirlenen bir uygunluk fonksiyon değerine ulaşıncaya kadar devam ettirilir. İçerisinde rasgele islemler barındırdığı için GA çoğunlukla local minimum değerlere takılmayıp global minimum değere ulaşabilen bir yöntemdir.

Bu çalışmada orijinal görüntü 4x4 vektörlere ayrıldıktan sonra GA kullanılarak sayısal görüntüyü temsil edecek en iyi vektör listesi oluşturup görüntü sıkıştırma gerçekleştirilmiştir. GA parametrelerinden populasyon sayısı artırıldığında algoritmanın çok yavaşladığı ve sonuca getirisinin çok fazla olmadığı, azaltıldığında ise resimde istenen düzelmenin sağlanamadığı gözlenmiştir. Bu yüzden popülasyon sayısı

10 ile sınırlandırılmıştır. Diğer genetik algoritma parametrelerinden mutasyon oranı %3, çaprazlanan vektör oranı %10, her bir vektör listesindeki vektör sayısını 8, 16, 32, 64 olarak belirlenmiştir. Önerilen GA-LBG algoritmasında ilk önce LBG algoritması çalıştırılıp elit popülasyon olarak belirlenir. Kalan 9 popülasyon ise orijinal görüntüden n adet başlangıç vektör listesi seçerek başlar. Bu vektörler, orijinal görüntü vektörlerinden rasgele biri olarak seçilebileceği gibi ikisinin veya üçünün ortalaması olarak da belirlenebilir. Daha sonraki GA-LBG adımlarında ise popülasyon içerisindeki vektörler Genetik algoritmanın parametreleri ile belirlenir. Herbir vektör listesi için (5) nolu formül ile OKH değeri bulunup en düşük hata değerini veren vektör listesi elit vektör listesi ile karşılaştırılır daha düşük OKH değeri veren vektör listesi yeni elit olarak belirlenir. Önerilen GA-LBG görüntü sıkıştırma algoritmasının adımları aşağıdaki şekilde belirlenmiştir ;

Adım 1 (Yeni popülasyon üretimi) : Elit popülasyonu korumak şartı ile popülasyonların %40' ını çaprazlama sonucunda oluşan vektörler ile, % 40'ını mutasyon sonunda oluşan vektörler ile %20 sini ise orijinal görüntü vektörlerinden oluşan kombinasyon ile belirle.

Adım 2 (Çaprazlama): Önceki adımda belirlenen n adet vektör listesinin herbiri içerisindeki vektörlerin %10 unu diğer bir listedeki vektörler veya elit sınıfa ait vektörler ile yer değiştir.

Vektör Listesi A :
[V1A V2A V3A V4A V5A … V32A]
Vektör Listesi B :
[V1B V2B V3B V4B V5B … V32B]
Vektör Listesi C :
[V1C V2C V3C V4C V5C … V32C]

Rasgele yapılan çaprazlama sonucunda yeni A, B, C vektör listeleri aşağıdakine benzer şekilde değiştirilir;

Vektör Listesi A :
[V1A V2A V3A V4B V5A … V32C]
Vektör Listesi B :
[V1B V2C V3B V4A V5B … V32B]
Vektör Listesi C :
[V1C V2B V3C V4C V5C … V32A]

Adım 3 (Mutasyon): Değişiklik miktarı toplamda %3'ü geçmeyecek şekilde herhangi bir vektör listesinin herhangi bir vektörüne ait herhangi bir pixel değerini rasgele ±%30 oranında değiştir.

V1A: [10 20 40 50 30 90 100 200 80 70 0 30 70 5 15 64]

Rasgele yapılan mutasyon sonucunda V1A vektörünün elemanları aşağıdakine benzer şekilde değiştirilir;

V1A: [10 20 40 58 30 90 73 200 80 70 0 30 70 5 15 64]

Adım 4 (Uygunluk fonksiyonu): İlk üç adımda oluşturulan n adet vektör listesinin her biri için aşağıda verilen (8) nolu formül ile OKH(Ortalama karesel hata) değeri hesapla. Küçükten büyüğe doğru sıralayıp en az hatayı veren vektör listesini Elite sınıfın sonucu ile karşılaştır, hata değeri daha düşük ise yeni elit sınıf olarak belirle. İstenen iterasyon sayısına veya OKH değerine ulaşılmadı ise Adım 1'e git.

$$OKH = \frac{1}{MN}\sum_{i=1}^{n}\sum_{ij=1}^{16}(x_{ij} - v_j)^2 \quad (8)$$

Burada M.N görüntü içerisindeki toplam piksel sayısını, *n* görüntü içerisindeki blok sayısı, *v* ise vektör listesi içerisinde x_i'e en yakın vektörü temsil etmektedir.

3. SIKIŞTIRMA SONUÇLARI

Bu çalışmada Genetik algoritma ile LBG birleştirilerek yeni bir kayıplı sıkıştırma algoritması geliştirilmiştir. Standart 256 gri tonlu, 256x256 piksel boyutlu, histogramları birbirinden çok farklı Lena ve Cameraman görüntüleri 4x4 piksel boyutunda 4096 adet alt vektöre ayrılmış literatürde bilinen K-Ortalamalar, LBG ile Vektör Nicemleme, Bulanık C Ortalamalar ve önerilen Genetik-LBG algoritması ile önce kümeleme daha

Şekil 2(a) Orijinal Lena Görüntüsü

Şekil 2(b) KO, OKH=274.3, 0.188bp

Şekil 2(c) LBG, OKH=256.6, 0.188bp

Şekil 2(d) GA-LBG, OKH=245.4, 0.188bp

Çizelge 1. Algoritmaların orijinal LENA görüntüsü için değişik sıkıştırma oranlarında OKH performans karşılaştırması

Kodlama Oranı	KO	BCO	LBG	GA-LBG
0.188 bp	274.3	265.5	256.6	245.4
0.25 bp	195.6	193.5	185.9	177.8
0.3125 bp	150.1	149.3	142.7	137.7
0.375 bp	111.3	109.1	105.9	101.4

sonra kayıplı sıkıştırma işlemi gerçekleştirilmiştir. Orijinal görüntüde bir piksel, $256=2^n$ eşitliğinden n=8 bit = 1 byte ile ifade edilip 4x4 piksel boyutlu bir vektör=4x4x8=128 bit ile ifade edilir. Kümeleme algoritmaları ile resmi en iyi ifade eden, uzunluğu 8, 16, 32 veya 64 olan vektör listesi oluşturulur. Örneğin resim için 16 elemanlı bir vektör listesi hazırlandı ise her bir vektör, listedeki sıra numarası ile ifade edilir. Böylelikle orijinal görüntüdeki her bir blok 128 bit yerine vektörün sıra numarası kullanılarak $16=2^n$ eşitliğinden n=4 bit ile kodlanıp 128/4 = 32, yani 32:1 oranında sıkıştırılmış 8/32=0.25 bit/piksel (bp) oranı ile kodlanmış olur. Dolayısı ile 8,16,32 ve 64 vektör uzunlukları sırası ile 42.6:1, 32:1, 25.6:1, 21.33:1 oranında sıkıştırılmış ve sırası ile 0.188 bp, 0.25 bp, 0.3125 bp, 0.375 bp oranı ile kodlanmıştır. Önerilen GA-LBG

algoritmasının MSE performansının 8, 16, 32 ve 64 vektör listeli sonuçları diğer KO, BCO, LBG algoritmalarından daha iyi olduğu sayısal olarak Çizelge 1 ve Çizelge 2'den görsel olarak da Şekil 2 ve Şekil 3'ten görülmektedir.

Çizelge 2. Algoritmaların orijinal Cameraman görüntüsü için değişik sıkıştırma oranlarında OKH performans karşılaştırması

Kodlama Oranı	KO	BCO	LBG	GA-LBG
0.188 bp	414.8	407.2	398.3	389.6
0.25 bp	312.7	310.4	302.4	294.5
0.3125 bp	243.3	241.1	235.1	230.0
0.375 bp	178.5	176.6	172.1	168.6

Şekil 3(a) Orijinal Cameraman Görüntüsü

Şekil 3(b) KO, OKH=414.8, 0.188bp

Şekil 3(c) LBG, OKH= 398.3, 0.188bp

Şekil 3(d) GA-LBG, OKH=389.6, 0.188bp

4. SONUÇ

Bu çalışmada güncel kümeleme algoritmaları yardımı ile bulunan merkezler üzerinden gerçekleştirilen kayıplı görüntü sıkıştırma algoritmalarının performansları, önerilen Genetik LBG Algoritması ile iyileştirilmiştir. Önerilen yeni algoritma standart Lena ve Cameraman görüntüleri üzerinde denenmiş, klasik yöntemlerden hem PSNR değerleri, hem de sıkıştırılıp açılan görüntü kalitesi açısından yüksek sıkıştırma oranlarında daha üstün olduğu gözlenmiştir.

Kaynaklar

[1] Lloyd, Stuart P., "Least squares quantization in PCM", IEEE Transactions on Information Theory, 28 (2): 129–137 (1982).

[2] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., Wu, A. Y., "An efficient k-means clustering algorithm: Analysis and implementation", IEEE Trans. Pattern Analysis and Machine Intelligence, 24: 881–892 (2002).

[3] Likas A., Vlassis N., Verbeek J.J., "The global *k*-means clustering algorithm", Pattern Recognition, 36(2): 451-461 (2003).

[4] Bagirov, A. M., Ugon, J., Webb, D., "Fast modified global *k*-means algorithm for incremental cluster construction", Pattern Recognition, 44(4): 866-876 (2011).

[5] Bezdek, J.C., Ehrlich, R., Full, W., " FCM: The Fuzzy C-Means clustering algorithm", Computers & Geosciences, 10(2-3): 191-203 (1984).

[6] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", Journal of Cybernetics, 3: 32-57 (1973).

[7] Ya-zhong, L., Gan H., Jin-ku G.U., "Improved FCM algorithm using difference of neighborhood information", Journal of Computer Applications, 31(2): 375-378 (2011).

[8] Kang J., Min L., Luan Q., Li X., Liu J., "Novel modified fuzzy c-means algorithm with applications", Digital Signal Processing, 19(2): 309-319 (2009).

[9] Gray, R.M., "Vector Quantization", IEEE ASSP Magazine, 1(2): 4-29 (1984).

[10] Linde Y., Buzo A, Gray R. M., "An Algorithm for Vector Quantizer Design", IEEE Transactions on Communications, 28: 84-95 (1980).

[11] Lin, Y.C & Tai, S.C.,"A Fast Linde-Buzo-Gray Algorithm in Image Vector Quantization", IEEE Transactions on Circiuts and Systems-II : Analog and Digital Signal Processing, 45: 432-435 (1998).

[12] Patane G., Russo M., "The enhanced LBG algorithm ", Neural Networks , 14 : 1219 – 1237(2001).

[13] Tsai C.W., Lee C.Y., Chiang M.C., Yang C.S.,"A fast VQ codebook generation algorithm via pattern reduction", Pattern Recognition Letters, 30: 653–660 (2009).

[14] Pan Z.B., Yu G.H., Li Y., "Improved fast LBG training algorithm in Hadamard domain", Electronics Letters, 47(8): 488-489 (2011).

[15] Holland J.H., "Adaptation in Natural and Artificial Systems", 1975.

[16] Wang F.H., Jain L.C., Pan J. S., "VQ-based watermarking scheme with genetic codebook partition", Journal of Network and Computer Applications, 30(1): 4-23 (2007).

[17] Zhang L., Zheng B., Yang Z., "Codebook design using genetic algorithm and its application to speaker identification", Electronics Letters, 41(10): 619-620 (2005).

[18] Franti P., "Genetic algorithm with deterministic crossover for vector quantization", Pattern Recognition Letters, 21: 61-68 (2000).

Geliş Tarihi: 26.08.2013 **Kabul Tarihi: 09.12.2013**