Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi
Iğdır University Journal of the Institute of Science and Technology

Görüntü Sıkıştırmada Kod Vektör Listesi Üretimi İçin Yeni Bir Bölme Tabanlı LBG Algoritması

Ilker KILIÇ[1], Yücel KOÇYİĞİT[1], Mustafa NİL[1]

ÖZET: Linde-Buzo-Gray (LBG) algoritması, görüntü sıkıştırmada Vektör Nicemleme (VN) tekniği için kullanılan, performansı büyük oranda başlangıç kod vektör listesine bağlı, kararlı, yerel optimum sonuç veren bir tekniktir. Bölme tabanlı LBG algoritmasında görüntüyü oluşturan vektörlerin ortalaması başlangıç olarak seçilir. Merkezler 2^n kez ikiye bölünüp güncellenerek yerel optimum kod listesi oluşturulur. Önerilen yeni teknikte (YLBG), LBG algoritması geliştirilmiş ve algoritma içerisindeki bölme işlemi tüm bölgelere uygulanmayıp sadece hatası en fazla olan bölge tespit edilip o bölgenin merkezi ikiye bölünerek merkez artırması sağlanmıştır. Böylece mevcut standart LBG'den farklı olarak merkezler teker teker artırılıp kod vektör listesinin performansı global olarak artırılmıştır. Bu çalışmada önerilen yeni teknik standart görüntülere uygulanmış, K-Ortalamalar (KO), LBG ve Bulanık C-Ortalamalar (BCO) ile karşılaştırılmış, ortalama karesel hata(OKH) ölçütüne göre üstün olduğu görülmüştür.

Anahtar Kelimeler: Bulanık C–ortalamalar , K-ortalamalar, LBG, vektör nicemleme

A New Spliting LBG Algorithm for Codebook Generation in Image Compression

ABSTRACT: Linde-Buzo-Gray (LBG) algorithm is used in image processing for Vector Quantization (VQ). LBG technique is robust, performs locally best but depends on the initial codebook. In the splitting based VQ, the first center is defined as average of all vectors. The rest of 2^n centres are calculated by splitting and updated procedure. In the proposed new technique (NLBG) the LBG is improved and insted of splitting all centres into two new areas, the worst area that has highest mean square error splitted and updated into to new areas. Therefore, the number of codevectors is increased one by one apart from the classical LBG. Consequently, the performance of the codebook is increased globally. In this paper, the new technique is applied to the standard images, compared to the FCM(Fuzzy C-Means), K-Means (K – Ortalamalar) and LBG. As a result, it is seen that the proposed new technique performs better according to the criteria of MSE.

Keywors: Fuzzy C-means, K-means, LBG, vector quantization

Cilt/*Volume*: 7, Sayı/*Issue*: 1, Sayfa/*pp*: 115-123, 2017
ISSN: 2146-0574, e-ISSN: 2536-4618 DOI: 10.21597/jist.2017127424

[1] CBU, Mühendislik, Elektrik-Elektronik Müh., Manisa, Türkiye
[e]Sorumlu yazar/Corresponding Author: İlker KILIÇ, ilkerkilic71@gmail.com

Geliş tarihi / *Received:* 16.05.2016
Kabul tarihi / *Accepted:* 23.08.2016

GİRİŞ

Video ya da görüntü çekim uygulamaları ile 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 sınırlı depolama alanına sahip uygulamalarda karşımıza sorun olarak çıkmaktadır. 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 LBG ile vektör nicemleme (Linde et al., 1980; Gray, 1984; Lin and Tai, 1998; Patane and Russo, 2001; Tsai et al., 2009; Pan et al., 2011, Ku et al., 2014, Khan et al., 2015), Bulanık C-Ortalamalar (BCO) (Dunn, 1973; Bezdek et al., 1984; Ya-zhong et al., 2011; Kang et al., 2009), K-Ortalamalar (KO) (Lloyd, 1982; Bagirov et al., 2011; Bai et al., 2013; Tzortzis and Likas, 2014) algoritmalarıdır. Klasik LBG algoritması iyi bir başlangıç kod vektör listesi hesaplanıp algoritma başlangıcında kullanılarak geliştirilmiştir (Patane, 2001). Klasik bölme tabanlı LBG algoritmasında her bir küme ikiye bölünerek çoğaltılmata iken önerilen yeni teknikte (YLBG) herbir iterasyonda ortalama karesel hatası en büyük olan küme ikiye bölünmüş ve hatanın sürekli minimize edilmesi amaçlanmıştır. Sonuçta aynı sıkıştırma oranlarında diğer algoritmalardan daha düşük ortalama karesel hata değerlerine sahip sonuçlar elde edilmiştir.

MATERYAL VE YÖNTEM

Vektör nicemleme ve LBG algoritması

Vektör nicemleme kayıplı bir görüntü sıkıştırma algoritmasıdır. Bu algoritmanın temeli Linde-Buzo-Gray tarafından geliştirilen LBG kod vektör listesi üretim mekanizmasına dayanmaktadır. Algoritma daha sonra birçok kez geliştirilmiştir. LBG algoritmasında ilk önce NxN piksel boyutuna sahip dijital bir $Y=\{x_{ij}\}$ görüntüsü mxm boyutlarına sahip alt vektörlere ayrılır.

Böylece tüm görüntü $N_b=\left(\frac{N}{m}x\frac{N}{m}\right)$ adet alt bloklardan oluşan vektörler ile temsil edilmiş olur; $X = \{x_i, i = 1,2,. . . , N_b\}$. L değişkenini mxm boyutunda bir vektör olarak kabul ettiğimizde görüntü içerisindeki her bir x_i vektörü, $\Re$ $\Re^L$ Euclidean uzayına ait olup $x_i \in \Re^L$ şartını sağlamalıdır. Kod vektör listesi içerisinde resmi en iyi temsil eden N_C adet kod vektöründen oluşmaktadır. Orijinal görüntü vektörleri bir satır vektörü ile ifade edilirken, i. kod vektörü şeklinde tanımlanabilir. Vektör nicemlemede her bir görüntü bloğu kod vektör listesi içerisindeki hatası en az olan vektör ile temsil edilir. Böylece orijinal görüntü bloğu yerine onun kod vektör listesindeki numarası ile temsil edilip kayıplı bir sıkıştırma işlemi yapılmış olur. C kod vektör listesinin belirlenmesi için aşağıdaki ortalama karesel hata kriterinin sağlanması gerekir Eşitlik(1-3).

$$\text{OKH(C)} = \text{OKH(C)} = \frac{1}{N_b}\sum_{j=1}^{N_c}\sum_{i=1}^{N_b}\mu_{ij}\left\|x_i - c_j\right\|^2 \quad (1)$$

$$\sum_{j=1}^{N_c}\mu_{ij} = 1\ ,\ i\in\{1,2,\dots,N_b\} \quad (2)$$

$$\mu_{ij} = \begin{cases} 1, & \text{eğer } x_i \text{ j. küme içerisinde ise} \\ 0, & \text{diğer durumlarda} \end{cases} \quad (3)$$

Burada $\left\|x_i - c_j\right\|$ terimi x_i orijinal görüntü bloğu ile c_j kod vektörü arasındaki Euclidean uzaklığı, μ_{ij} kümeye aidiyet katsayısını ifade etmektedir. Yerel optimum kod vektör listesinin oluşturulabilmesi için aşağıdaki kriterlerin sağlanması gerekmektedir.

a) R_j, j=1,2,…,N_c vektör grubu aşağıdaki koşulu sağlamalıdır;

$$R_j \supset \{x \in X: d(x,c_j) < d(x,c_k), \forall k \neq j\} \quad (4)$$

b) R_j vektör grubunun merkezi c_j aşağıdaki şekilde hesaplanır.

$$c_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i, x_i \in R_j \quad (5)$$

Burada N_j, R_j vektör grubuna ait toplam eleman sayısını ifade eder.

Orijinal görüntüye ait vektörler x_i, i = 1,2, …,N_b, Euclidean uzaklığı d ve başlangıç kod vektör listesinin $c_j(0)$, j = 1, 2,…,N_c olduğunu kabul edersek, LBG algoritmasın lokal optimum kod vektör listesini oluşturabilmesi için aşağıdaki üç kuralı sırası ile işletmesi gerekmektedir;

a) Orijinal görüntüye ait tüm vektörlerin ortalaması ilk merkez olarak kabul edilir.

$$c_1 = \frac{\sum_{i=1}^{N_b} x_i}{N_b} \quad (6)$$

b) Görüntüdeki tüm merkezlere ± ε ilave edilerek merkezler ikiye bölünür.

$$c_{ia} = c_i + \varepsilon, \quad i=1,2,\ldots N_c \quad (7)$$

$$c_{ib} = c_i - \varepsilon, \quad i=1,2,\ldots N_c \quad (8)$$

c) c_{ia} ve c_{ib} merkezleri Eşitlik(4,5) yardımı ile merkezlerin değerleri sabit kalıncaya kadar işletilir.

Bulanık C-Ortalamalar Algoritması

Bulanık C-Ortalamalar (BCO) algoritması, ilk olarak (Dunn, 1973) tarafından bulunmuş ve daha sonra (Bezdek et al., 1984) tarafından genelleştirilmiştir. İlerleyen yıllarda algoritma, kümeleme üzerine çalışan değişik araştırmacılar tarafından geliştirilmiştir (Kang et al., 2009; Ya-zhong et al., 2011).

Basit olan bu sınıflama yöntemi, sınır değişkenliği yüksek olan sınıflara ait problemlerin çözümü için etkili bir yöntemdir. Bu yöntemde uzaklık ölçütü karar verme fonksiyonudur ve en küçük olması istenir. Uzaklık hesabı, bir işaret vektörünün o sınıfın merkezine ait vektörden çıkarılarak yapılır.

Klasik sınıflama algoritmaları göz önüne alındığında $X=\{x_1, x_2, \ldots, x_n\}$, bir işaret kümesi; n, öznitelik sayısı; p, öznitelik vektörlerinin boyutu; c sınıf veya öbek sayısı; $V=\{v_1, v_2,\ldots,v_c\}$ sınıf veya öbek merkezleri kümesi; U ise üyelik matrisini göstermek üzere, öbek merkezi ifadesi

$$v_i = \frac{\sum_{k=1}^{n} u_{ik} x_k}{\sum_{k=1}^{n} u_{ik}} \quad ;i = 1,2,...,c \tag{9}$$

şeklindedir.d_{ik} , k ncı vektörün i nolu sınıftan Euclidean uzaklığı

$$d_{ik} = \|x_k - v_i\| = \sqrt{\sum_{j=1}^{p} |x_{kj} - v_{ij}|^2} \tag{10}$$

şeklindedir.

X kümesinin en iyi öbekleştiği dağılımı

$$J_w(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} d_{ik}^2 \tag{11}$$

hedef fonksiyonunu minimum yapan değerdir. Klasik küme kavramı için verilen bu ifadede üyelik değerleri 1 ya da 0 değerini alır. BCO fonksiyonlarında

$$J_m(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m d_{ik}^2 \tag{12}$$

şeklinde ifade edilir. d_{ik} uzaklığı (10) denklemdeki tanıma uygun şekilde ifade edilmektedir. Burada u_{ik} üyelik değeri 0 ile 1 arasında sonsuz sayıda değer alabilmektedir.

Hedef fonksiyonundaki m değeri, 1 ile sonsuz arasında bir değer alan bulanıklığı artıran bir kontrol parametresidir. Literatürde genellikle m=2 değeri seçilmektedir. J_m bulanık karesel hata hedef fonksiyon olmak üzere minimum değerinde, öbek merkezlerinin son değerlerine ulaşılır.

Her bir iterasyonda üyelik değerlerinin hesabı,

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{\frac{2}{m-1}}} \tag{13}$$

ile yapılmaktadır. Bu ifadede u_{ik} , k ncı işaret vektörünün i nci sınıfa ait olma derecesi olan üyelik değerini vermektedir.

Öbek merkezi ifadesi ise keskin küme tanımından farklı olup bu ifadeye bulanık üyelik değerleri girmektedir. Buna göre bulanık öbek merkezi ifadesi ;

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \quad ; i = 1,2,3,...,c \qquad (14)$$

şeklinde yazılabilir.

BCO algoritmasının takip ettiği adımlar aşağıda verilmiştir :

1. adım : Başlangıç değerlerini belirle (üyelik değerleri, c sınıf (öbek) sayısı, ε hata değeri).

2. adım : Bulanık öbek merkezlerini, Eşitlik(14) denklemini kullanarak hesapla

3. adım : Eşitlik(10) denklemini kullanarak her bir işaret vektörü için Euclidean uzaklığını hesapla

4. adım : Yeni üyelik değerlerini , Eşitlik(13) ifadesini kullanarak hesapla.

5. adım : Bulanık öbek merkezlerini güncelle

K-Ortalamalar Algoritması

Algoritma ilk olarak Stuart P. Lloyd tarafından geliştirilmiştir (Lloyd, 1982). Algoritmanın basit ve kullanışlı olmasından dolayı birçok araştırmacı bu algoritma üzerinde çalışmış ve algoritmayı daha iyi sonuç verecek şekilde geliştirmişlerdir (Bagirov ve ark., 2011; Bai ve ark., 2013; Tzortzis ve Likas, 2014). Standart K-Ortalamalar algoritması, n adet sayısal veriyi önceden belirlenen k adet kümeye Euclidean uzaklığını kullanarak böler. Bu bölünen kümelere Voronoi hücreleri denir. k, küme sayısı olup algoritmaya dışarıdan girilmektedir. $S=\{S_1, S_2, \ldots, S_k\}$ veri içerisinde oluşan k adet kümeyi temsil edip K-Ortalamalar algoritmasında amaç (15) nolu denklemdeki hedef fonksiyonunu minimize etmektir.

$$J = \arg\min \sum_{i=1}^{k} \sum_{x \in Si}^{Ni} \left\| x_i - \mu_i \right\|^2 \qquad (15)$$

Burada μ_i, i. kümenin ortalaması, N_i , S_i kümesinin eleman sayısıdır. Algoritmanın adımları;

1.Adım: k adet küme merkezi veri setinden rasgele olarak seçilir.

2.Adım: Veri setindeki her bir x_p vektörünün k adet küme merkezlerine olan Euclidean uzaklığı hesaplanır. Veri, hesaplanan uzaklıklar içinde en küçük olan kümeye atanır.

$$S_i = \{x_p : \left\| x_p - m_i \right\|^2 < \left\| x_p - m_j \right\|^2 \ \forall_j \ , \ 1 \leq j \leq k \qquad (16)$$

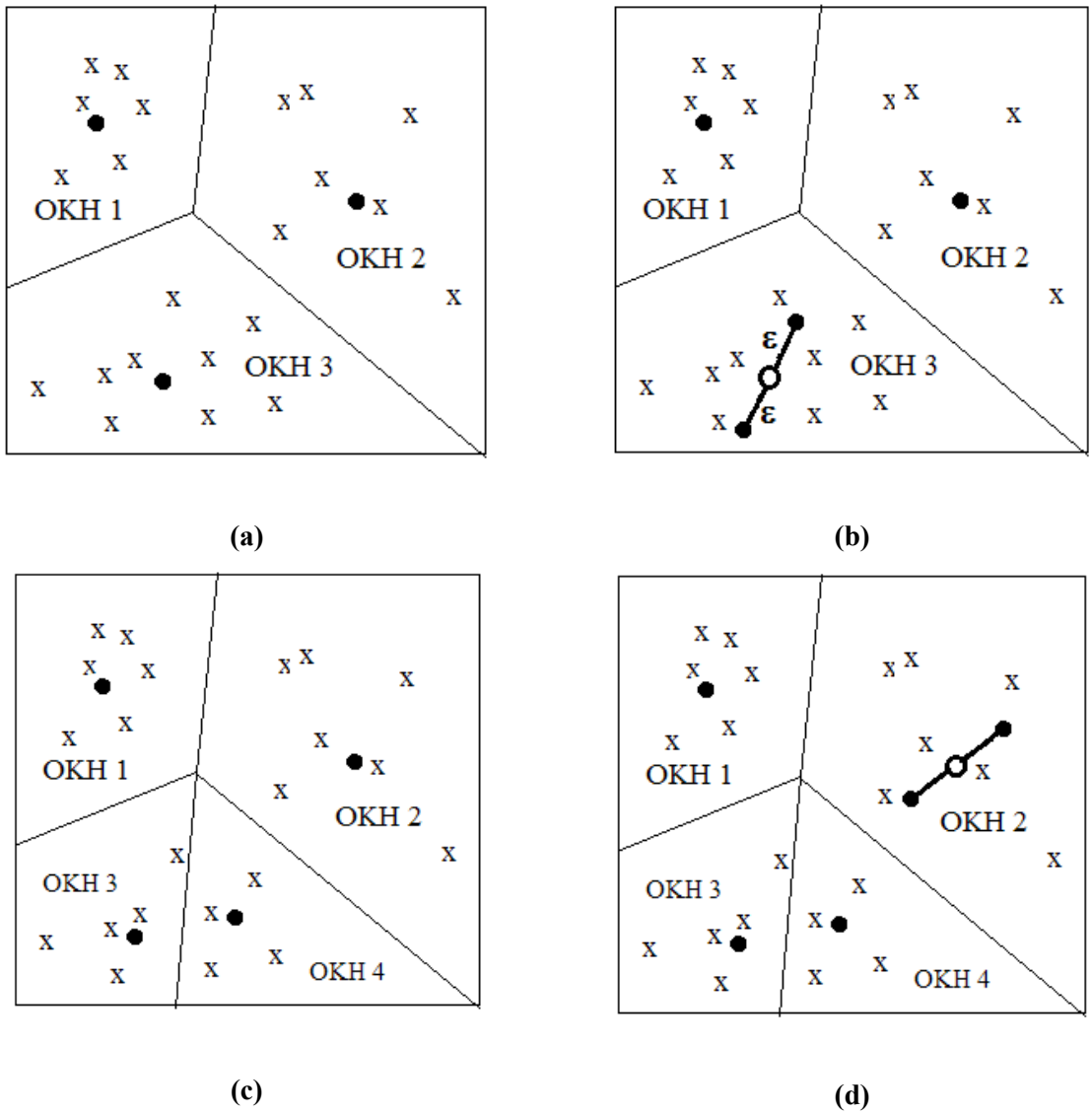
3.Adım:Herbir kümeye ait yeni küme merkezleri Eşitlik(6) ile belirlenir. 4.Adım: Yeni küme merkezleri ile bir önceki iterasyonda hesaplanan küme merkezleri arasındaki mutlak farkların toplamı önceden belirlenen bir ε eşik değerinden büyük ise adım2'ye gidilir aksi durumda algoritma durdurulur.

Yeni Bölme Tabanlı Hata Kontrollü Geliştirilmiş LBG Algoritması

Bölme tabanlı LBG algoritmasında merkez sayısını iki katına çıkaran bölme işlemi bölgelerin hata miktarlarını dikkate almadan standart uygulanan bir işlemdir.

Herhengi bir bölgenin OKH değeri az ya da çok olmasına bakılmaksızın bölgeye yeni iki merkez atanmaktadır.

Bu da bazen bölgenin ihtiyacı olmamasına rağmen fazladan yeni merkez atanmasına neden olmaktadır.



Şekil 1. a) YLBG algoritmasında belirlenmiş üç bölge **b)** OKH 3 en büyük hata değeri olarak belirlenmiş ve eski merkeze ε uzaklığında iki yeni merkez oluşturulmuştur. **c)** Dört bölge güncellenerek son haline getirilmiştir. **d)** İkinci bölgenin en yüksek OKH değerine sahip olduğu tespit edilmiş ve merkez bölünerek çoğaltılmıştır

Önerilen YLBG tekniğinde ise verilere yeni bir merkez atanacağı zaman tüm bölgelerin ayrı ayrı OKH değerleri hesaplanmakta ve OKH değeri en yüksek olan bölgede eski merkez bölünerek yeni merkez oluşturulmaktadır. Bu yöntem sayesinde resmin tüm bölgelerine ait OKH değerleri oldukça düşük kalmaktadır. Şekil 1'de YLBG algoritmasına ait bu işlem gösterilmiştir.Önerilen YLBG algoritmasının adımları aşağıda sıralanmıştır.

Adım 1: Görüntüye ait tüm vektörlerin ortalaması ilk merkez olarak kabul edilir.

$$c_1 = \frac{\sum_{i=1}^{N_b} x_i}{N_b} \qquad (17)$$

Adım 2: Görüntüde hatası en yüksek merkez tespit edilip ± ε ilavesi ile bölgeye ait iki yeni merkez oluşturulur. Bölünen eski merkez yok edilir.

$$c_{ia} = c_i + \varepsilon , \quad i=1,2,\ldots N_c \qquad (18)$$

$$c_{ib} = c_i - \varepsilon , \quad i=1,2,\ldots N_c \qquad (19)$$

Adım 3: Tüm merkezler Euclidean uzaklığı kullanılarak sabitleninceye kadar güncellenir. Yeni bir merkez bulunacaksa Adım 4'e gidilir aksi durumda algoritma sonlandırılır.

Adım 4: Tüm merkezlere ait OKH değerleri hesaplanır, Adım 2'ye gidilir.

BULGULAR VE TARTIŞMA

Önerilen YLBG algoritması 256x256 piksel boyutuna sahip standart Lena, Peppers, Baboon, sayısal görüntüleri üzerinde uygulanmıştır. Sayısal görüntü 4x4 pixel boyutuna sahip 4096 adet vektöre bölünerek analiz yapılmıştır. Herbir algoritma için 8, 16, 32, 64, 128 adet kod vektör listesi oluşturulup OKH değerleri hesaplanmıştır. Simülasyonlarda YLBG algoritması standart K-Ortalamalar (KO), LBG ve Bulanık C-Ortalamalar (BCO) ile karşılaştırılmış, otalama karesel hata(OKH) performans ölçütüne göre üstün olduğu görülmüştür. Algoritmaların zaman açısından performansları incelendiğinde KO'nın en hızlı, daha sonra sırası ile LBG, YLBG ve BCO algoritmalarının geldiği görülmektedir. YLBG algoritmasının LBG algoritmasına göre daha iyi sonuçları makul sayılablecek bir gecikme ile elde ettiği görülmüştür. Bu gecikmenin nedeni ise YLBG algoritmasında fazladan tüm bölgelerin OKH değerlerinin hesaplanmasından kaynaklı zaman gecikmesidir. Standart görüntülerden Lena, Peppers, Babboon görüntülerine ait sonuçlar sırası ile Çizelge 1-3'de gösterilmiştir. BCO, LBG ve YLBG algoritmalarına ait görsel karşılaştırmalı sonuçlar ise Şekil 2'de gösterilmiştir.

Çizelge 1. Standart 256x256 Lena görüntüsüne ait değişik boyutlardaki kod vektör listesi için OKH / Zaman (sn) performansları

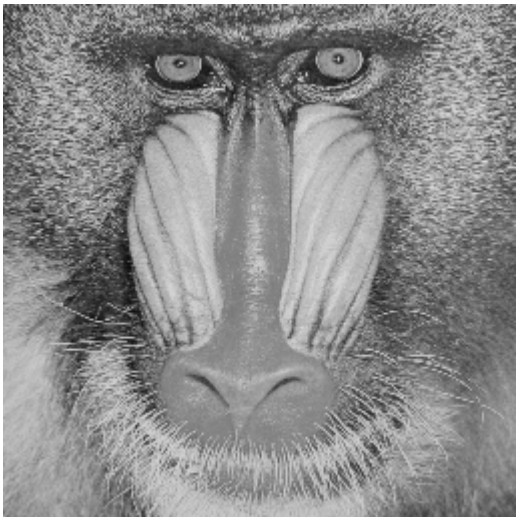
Kod vektör listesi boyut/ Algoritmalar	OKH/zaman BCO	OKH/zaman K-Ortalamalar	OKH/zaman LBG	OKH/zaman YLBG
8	328.76 / 5.1	329.81 / 1.0	321.56 / 1.0	319.77 /2.2
16	248.73/11.3	238.14 / 1.7	234.02 / 2.1	231.41/4.4
32	190.29/23.2	181.59/ 2.3	179.09 / 4.7	174.72/ 8.1
64	153.02/51.5	139.64 /5.1	136.03 / 8.6	131.63/17
128	136.92/102.2	106.54/14.8	100.72 / 16.2	97.41/34.1

Çizelge 2. Standart 256x256 Peppers görüntüsüne ait değişik boyutlardaki kod vektör listesi için OKH / Zaman (sn) performansları

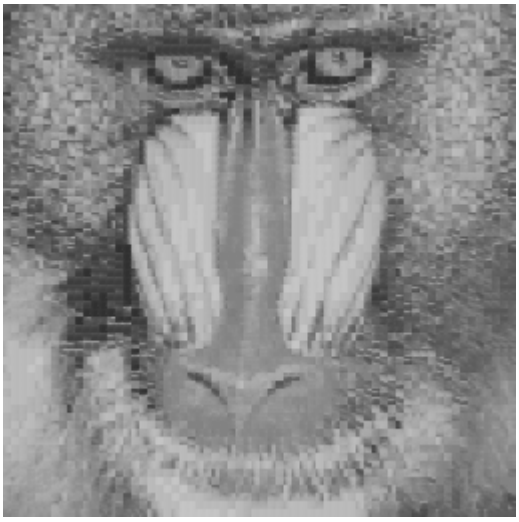
Kod vektör listesi boyutu/ Algoritmalar	BCO	K-Ortalamalar	LBG	YLBG
8	388.82/6.9	372.23/1.1	366.62/1.0	351.14/2.1
16	265.99/12.4	247.97/1.9	241.57/2.2	239.47/4.3
32	222.81/27.1	181.16/2.6	176.84/5.0	171.58/8.8
64	146.61/55.1	135.86/5.2	129.21/8.8	126.15/18.3
128	100.06/104.9	102.17/15.0	98.03/19.1	95.39/36.6

Çizelge 3. Standart 256x256 Baboon görüntüsüne ait değişik boyutlardaki kod vektör listesi için OKH / Zaman (sn) performansları

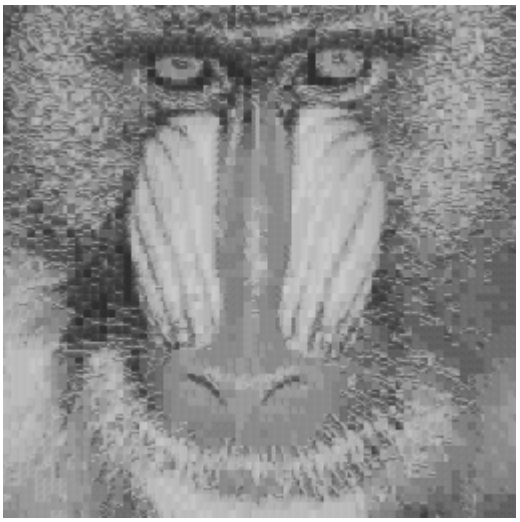
Kod vektör listesi boyutu/ Algoritmalar	BCO	K-Ortalamalar	LBG	YLBG
8	652.53/5.5	642.59/1.1	641.81/1.1	641.47/2.2
16	596.33/11.9	579.737/1.7	577.92/2.1	574.72/4.2
32	544.36/25.5	519.40/2.5	517.06/4.9	511.85/8.5
64	502.63/53.3	464.19/5.5	456.21/8.7	451.54
128	443.02/103.0	404.34/14.8	398.21/17.2	390.88/35.1



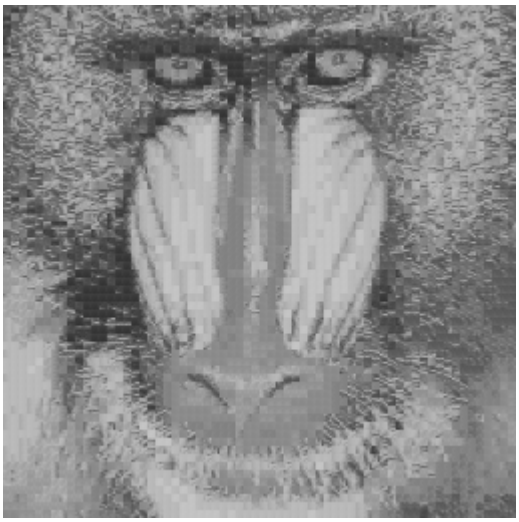
(a) Original 256x256 Baboon



(a) Bulanık C-Ortalamalar, OKH=502.63



(a) LBG, OKH = 456.21



(a) YLBG, OKH = 451.54

Şekil 2. 64 Kod vektor listesi icin Orijinal, BCO, LBG, YLBG algoritmalarının OKH performanslarının görsel karşılaştırması

SONUÇ

Bu çalışmada standart görüntü kümeleme algoritması LBG geliştirilmiştir. Önerilen yeni teknikte (YLBG), LBG algoritması içerisindeki var olan tüm kümeleri ikiye bölüp yeni kümeler oluşturma işlemi yerine sadece hatası en fazla olan bölge tespit edilmiş ve o bölgenin merkezi ikiye bölünerek merkez artırması sağlanmıştır. Böylece mevcut standart LBG'den farklı olarak merkezler teker teker artırılıp kod vektör listesinin performansı global olarak artırılmıştır. Bu çalışmada önerilen yeni teknik standart görüntülere uygulanmış, K-Ortalamalar (KO), LBG ve Bulanık C-Ortalamalar (BCO) ile karşılaştırılmış, otalama karesel hata(OKH) performans ölçütüne göre üstün olduğu görülmüştür.

KAYNAKLAR

Gray RM, 1984. Vector Quantization. IEEE ASSP Magazine, 1(2): 4-29.

Linde Y, Buzo A, Gray RM, 1980. An Algorithm for Vector Quantizer Design. IEEE Transactions on Communications, 28: 84-95.

Lin YC, Tai SC, 1998. 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.

Patane G, Russo M, 2001. The enhanced LBG algorithm. Neural Networks, 14: 1219 – 1237.

Tsai CW, Lee CY, Chiang MC, Yang CS, 2009. A fast VQ codebook generation algorithm via pattern reduction. Pattern Recognition Letters, 30: 653–660.

Pan ZB, Yu GH, Li Y, 2011. Improved fast LBG training algorithm in Hadamard domain. Electronics Letters, 47(8): 488-489.

Ku NY, Chang SC, Hwang SH, 2014. Binary search vector quantization. AASRI Procedia, 8: 112 – 117.

Khan MAU, Mousa WA, Khan TM, 2015. Entropy-constrained reflected residual vector quantization: A realization of large block vector quantization. Optik, 126: 888–897.

Khah SS, Chouakria AD, Gaussier E, 2016. Generalize d k -means-base d clustering for temporal data under weighted and kernel time warp. Pattern Recognition Letters, 75: 63–69.

Dunn JC, 1973. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics, 3: 32-57.

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

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

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

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

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

Bai L, Liang J, Sui C, Dang C, 2013. Fast global k-means clustering based on local geometrical information. Information Sciences, 245: 168–180.

Tzortzis G, Likas A, 2014. The MinMax k-Means clustering algorithm. Pattern Recognition, 47: 2505–2516.